

Connecteur UXP 2, version 1.1

Guide de l'utilisateur

UXP-UG-CONNECTOR2

Table des matières

Notes de mise à jour de la dernière version de Connecteur UXP 2	1
1. Introduction	2
1.1. Public cible	2
1.2. Concepts	2
1.3. Connecteur UXP 2	2
1.3.1. Idée générale	2
1.3.2. Mettre en œuvre les API REST	3
2. Principes de fonctionnement	4
2.1. Flux de travail	4
2.2. Mappage des entrées et sorties	6
2.2.1. Une ligne	6
2.2.2. Plusieurs lignes	7
2.2.3. Éléments répétitifs imbriqués	8
3. Connexion à la base de données	11
3.1. Configurer la connexion à la base de données	11
3.2. Propriétés de la connexion à la base de données	14
3.3. Renouveler la clé de cryptage de la base de données	15
4. Sécurité des connexions	16
4.1. Mode de sécurité	16
4.1.1. Importance d'une connexion sécurisée	16
4.1.2. Authentification mutuelle TLS	16
4.1.3. Visualisation du mode de sécurité de la connexion	17
4.1.4. Désactivation de l'authentification client	17
4.2. Ajouter des certificats de confiance	18
4.2.1. Obtention du certificat	18
4.2.2. Certificats de confiance	18
4.2.3. Télécharger un certificat de confiance	19
4.2.4. Générer un magasin de clés pour le système client	19
4.3. Certificat TLS de Connecteur	20
5. Mettre en œuvre les API REST	22
5.1. Création d'un nouveau service	22

5.2. Créer un nouveau point de terminaison	24
5.3. Configurer les opérations de point de terminaison	26
5.3.1. Spécification des détails des opérations des points de terminaison.....	26
5.3.2. Spécification des paramètres d'entrée de l'opération de point de terminaison.....	27
5.3.3. Spécification des paramètres de sortie de l'opération du point de terminaison	29
5.3.4. Tester le fonctionnement du point de terminaison à l'aide de Connecteur	30
5.3.5. Tester le fonctionnement du point de terminaison à l'aide de Postman	31
5.3.6. Supprimer une opération	33
5.4. Exporter et importer des points de terminaison	33
5.5. Copier des points de terminaison	33
6. Enregistrer des API REST dans Serveur de sécurité UXP	35
7. Informations complémentaires	36
7.1. Spécification du mappage des paramètres.....	36
7.2. Types de données	38
8. Gérer les comptes d'utilisateurs	40
8.1. Casdoor	40
9. Dépannage	43
9.1. Surveillance	43
9.1.1. Obtenir les informations d'état	44
9.2. Journaux	44
9.3. Erreur de mémoire insuffisante	45
Annexe A: Notes de mise à jour Connecteur UXP 2	47

Notes de mise à jour de la dernière version de Connecteur UXP 2

1.1.0 (12.2025)

- Prise en charge d'Ubuntu 24, optimisations des performances, prise en charge de la base de données Oracle.

Pour toutes les notes de mise à jour de Connecteur UXP 2, voir l'[Annexe](#).

1. Introduction

1.1. Public cible

Ce guide de l'utilisateur s'adresse aux développeurs chargés de créer et de gérer les API REST partagées sur UXP.

Ce document s'adresse à des lecteurs ayant des connaissances de base en matière de services Web, de JSON et de bases de données.

1.2. Concepts

Membre UXP est une personne morale (ou physique) qui a rejoint UXP pour fournir et/ou consommer des services.

Sous-système est une représentation du système d'information du membre UXP qui fournit et/ou consomme des services. Un membre UXP peut avoir plusieurs sous-systèmes.

API REST est une interface de service Web qui permet aux clients du service d'accéder à des données et de les échanger. Dans Connecteur UXP 2, la mise en œuvre d'une API REST implique la configuration du service et de ses points de terminaison associés. Au sein d'UXP, une API REST est appelée service UXP.

Service est une entité au sein de Connecteur 2 qui fait partie de l'API REST. Pour permettre aux clients du service d'accéder à l'API REST, au moins un point de terminaison doit être configuré pour le service.

Point de terminaison est une entité au sein de Connecteur 2 qui fait partie de l'API REST. Chaque point de terminaison doit être lié à un service afin que les clients du service puissent interagir avec l'API REST.

Méthode d'opération fait référence à la méthode HTTP utilisée pour interagir avec le point de terminaison. Connecteur prend en charge les méthodes d'opération suivantes :— GET, HEAD, POST, PUT, PATCH et DELETE.

Description OpenAPI est une spécification normalisée qui définit la structure de l'API REST, en détaillant les points de terminaison, les méthodes, les paramètres et les réponses disponibles dans un format lisible par la machine.

1.3. Connecteur UXP 2

1.3.1. Idée générale

Connecteur UXP 2 est un outil permettant de mettre en œuvre des services Web compatibles avec le protocole UXP.

Connecteur UXP 2 :

- convertit les demandes REST entrantes en instructions SQL ;
- exécute les instructions sur les bases de données SQL spécifiées ;
- et convertit les ensembles de résultats de la base de données en réponses REST sortantes.

Avec Connecteur UXP 2, le développeur peut rapidement mettre en œuvre les services UXP en écrivant des demandes de base de données ou des instructions d'insertion/mise à jour. Pour les cas plus complexes, il est possible d'implémenter la logique applicative sous forme de procédure stockée pouvant être invoquée par le Connecteur UXP 2.

1.3.2. Mettre en œuvre les API REST

Pour mettre en œuvre une API REST dans Connecteur UXP 2 (ci-après dénommé Connecteur), vous devez fournir les informations suivantes :

- Informations générales sur le service et le point de terminaison—Informations sur le service et le point de terminaison (code de service, chemin d'accès au point de terminaison) nécessaires pour accéder à l'API REST par le client du service, et description informative de l'API REST.
- Connexion à la base de données pour le point de terminaison—connexion à la base de données SQL utilisée pour exécuter l'instruction SQL.
- Instruction SQL pour le point de terminaison—instruction que le Connecteur exécutera sur la base de données.
- Mappage des entrées pour le point de terminaison—Description de la manière de mapper les paramètres de la demande REST aux paramètres d'entrée de l'instruction SQL.
- Mappage de sortie pour le point de terminaison—Description de la manière de mapper le résultat renvoyé par la base de données après l'exécution de l'instruction SQL aux éléments d'un message de réponse REST.

2. Principes de fonctionnement

2.1. Flux de travail

Connecteur sert d'intermédiaire entre le Serveur de sécurité UXP et la base de données SQL. La figure 1 présente les composants et les protocoles de communication correspondants.

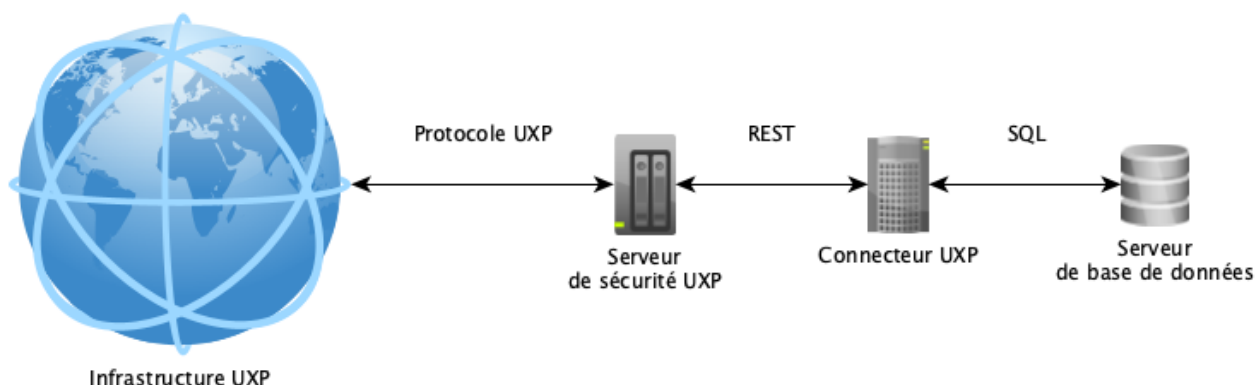


Figure 1. Diagramme de contexte du composant Connecteur

Les descriptions de services dans le Connecteur sont construites autour d'une instruction SQL paramétrée. En plus d'une instruction SQL, la configuration du point de terminaison contient des mappages d'entrée et de sortie qui convertissent les données SQL, respectivement, depuis et vers JSON. Lorsque le Connecteur reçoit une demande REST, les paramètres d'entrée de l'instruction SQL sont remplis avec les données de la demande et l'instruction est exécutée sur la base de données désignée. L'ensemble de résultats de sortie de l'instruction est ensuite utilisé pour élaborer la réponse REST. La figure 2 illustre ce processus de manière plus détaillée.

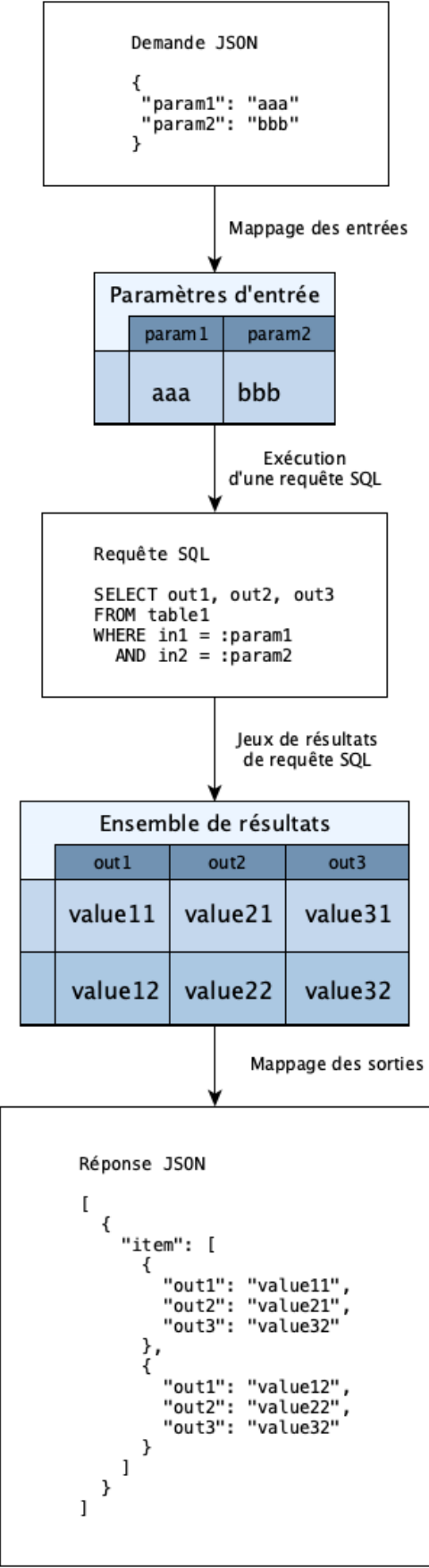


Figure 2. Traitement des messages dans Connecteur

En plus d'une instruction SQL, la configuration du point de terminaison contient des mappages d'entrée et de sortie qui convertissent les données SQL, respectivement, depuis et vers JSON. Le mappage d'entrée convertit la demande en paramètres d'entrée pour l'instruction SQL. Dans un cas élémentaire, l'entrée contient un ensemble de paramètres (représentés par un tableau d'une ligne).

Dans un cas plus complexe, l'entrée peut être représentée par une structure de données dans laquelle certains éléments peuvent apparaître plusieurs fois. Dans ce cas, le fichier JSON d'entrée est transformé en un tableau comportant plusieurs lignes — une pour chaque ensemble de paramètres d'entrée uniques. Pour plus de détails sur les mappages d'entrée et de sortie, voir [Mappage des entrées et sorties](#).

Dans l'étape suivante, l'instruction SQL est exécutée une fois pour chaque ligne du tableau d'entrée. Toutes les invocations liées à une demande REST unique sont exécutées dans une transaction unique, ce qui garantit l'atomicité du traitement de la demande. En général, l'ensemble de résultats contient plusieurs colonnes et plusieurs lignes.

Enfin, l'ensemble des résultats est transformé en une réponse REST. La transformation est pilotée par le mappage de sortie qui décrit la conversion de données tabulaires en JSON structuré.

2.2. Mappage des entrées et sorties

Dans le Connecteur, le mappage des entrées et des sorties convertit les données entre le format JSON et le format tabulaire. Les deux transformations sont l'inverse l'une de l'autre, la principale différence étant que les paramètres d'entrée sont identifiés par leur nom alors que les colonnes de sortie sont identifiées par des numéros de colonne. Par conséquent, par souci de concision, cette section se concentre uniquement sur le mappage de sortie qui convertit l'ensemble des résultats SQL au format JSON.

2.2.1. Une ligne

Dans le cas le plus simple, l'ensemble de résultats ne contient qu'une seule ligne (voir le tableau 1 par exemple). Dans ce cas, le mappage direct convertit chaque valeur en un élément de la réponse. Le mappage, présenté dans la figure 3, correspond à la structure du document JSON produit. Les attributs Optional et Repeat sont expliqués ci-dessous dans la section [Spécification des paramètres de sortie de l'opération du point de terminaison](#). Dans l'exemple de l'entrée, la réponse se présente comme suit :

```
[
  {
    "firstname": "Mary",
    "lastname": "Mercury"
  }
]
```

Tableau 1. Ensemble de résultats

1 (firstName)

2 (lastName)

Mary

Mercury

Details

Input Parameters

Output Parameters

Test Endpoint

READ FROM TEST QUERY

↑

↓

+

🗑️

☰

Col. No.	Name	Data Type	Optional	Repeat
1	firstname	string	<input type="checkbox"/>	<input type="checkbox"/>
2	lastname	string	<input type="checkbox"/>	<input type="checkbox"/>

SAVE

Figure 3. Mappage de sortie dans le cas le plus simple

2.2.2. Plusieurs lignes

Un autre cas typique est celui où la requête renvoie plusieurs lignes. Dans ce cas, les valeurs de sortie ne peuvent pas être directement affichées sous l'élément racine. Il faut donc ajouter un élément intermédiaire *person* au mappage de sortie. Comme cet élément peut apparaître plusieurs fois dans le document, l'attribut Repeat doit être sélectionné. La figure 4 montre le mappage qui convertit l'exemple d'entrée du tableau 2 en JSON :

```
[
  {
    "person": [
      {
        "lastname": "Smith",
        "firstname": "John"
      },
      {
        "lastname": "Murray",
        "firstname": "Rick"
      },
      {
        "lastname": "Redmond",
        "firstname": "Ben"
      }
    ]
  }
]
```

Tableau 2. Ensemble de résultats

1 (firstName)	2 (lastName)
John	Smith
Rick	Murray
Ben	Redmond

Details

Input Parameters

Output Parameters

Test Endpoint

READ FROM TEST QUERY

Col. No.	Name	Data Type	Optional	Repeat
	person		<input type="checkbox"/>	<input checked="" type="checkbox"/>
2	lastname	<div>string</div>	<input type="checkbox"/>	<input type="checkbox"/>
1	firstname	<div>string</div>	<input type="checkbox"/>	<input type="checkbox"/>

SAVE

Figure 4. Mapping avec répétition de l'élément racine

2.2.3. Éléments répétitifs imbriqués

Dans un cas encore plus complexe, l'ensemble de résultats relie plusieurs entités à l'aide de relations un-à-plusieurs. Supposons que la requête relie les tableaux « person » et « vehicle » de manière que chaque personne puisse posséder plusieurs véhicules. Le tableau 3 présente un exemple de résultat de requête sous forme de tableau. Dans la cartographie de sortie (voir figure 5), deux éléments intermédiaires sont introduits. Tout d'abord, *person* nous permet d'afficher plusieurs personnes. Deuxièmement, l'élément *vehicle* permet à chaque élément personne de disposer de plusieurs véhicules.

Tableau 3. Ensemble de résultats

1 (firstName)	2 (lastName)	3 (licenceplate)	4 (make)
John	Smith	324EAF	Toyota
John	Smith	876AFK	Kia
Cathy	Carr	166TAN	Volvo

i Details
 ↓ Input Parameters
 ↑ **Output Parameters**

📄 Test Endpoint

READ FROM TEST QUERY
 ↑ ↓ + *🗑️* *☰*

Col. No.	Name	Data Type	Optional	Repeat
	person		<input type="checkbox"/>	<input checked="" type="checkbox"/>
1	firstname	string ▼	<input type="checkbox"/>	<input type="checkbox"/>
2	lastname	string ▼	<input type="checkbox"/>	<input type="checkbox"/>
	vehicle		<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	licenceplate	string ▼	<input type="checkbox"/>	<input type="checkbox"/>
4	make	string ▼	<input type="checkbox"/>	<input type="checkbox"/>

SAVE

Figure 5. Mappage avec élément répétitif imbriqué

Lors de la génération de JSON à partir de données tabulaires, le Connecteur itère sur les lignes et génère un nouvel élément à chaque fois que les champs correspondant à l'élément changent. Dans l'exemple du jeu de données, la première ligne produite entraîne la génération d'un nouvel élément parent *person* et d'un nouvel élément enfant *vehicle*. Dans la deuxième ligne, les valeurs des champs *firstname* et *lastname* ne sont pas modifiées, l'élément *person* reste donc le même.

Toutefois, *licenceplate* et *make* ont été modifiés, de sorte qu'un nouvel élément *vehicle* est créé sous l'élément *person* existant. Dans la troisième ligne, les champs *firstname* et *lastname* sont modifiés et une nouvelle *person* est créée avec un nouveau *vehicle*. Le résultat final est le JSON suivant :

```

{
  "person": [
    {
      "firstname": "John",
      "lastname": "Smith",
      "vehicle": [
        {
          "licenceplate": "324EAF",
          "make": "Toyota"
        },
        {

```

```
        "licenceplate": "876AFK",
        "make": "Kia"
    }
]
},
{
    "firstname": "Cathy",
    "lastname": "Carr",
    "vehicle": [
        {
            "licenceplate": "166TAN",
            "make": "Volvo"
        }
    ]
}
]
```

Ainsi, le Connecteur suit une règle de base : chaque ligne est comparée à la précédente. Pour chaque élément de sortie, si l'un des champs appartenant à l'élément est modifié, un nouvel élément de sortie est créé.



L'algorithme suppose que l'ensemble des résultats est ordonné de manière que les lignes appartenant à chaque élément soient regroupées. Dans l'exemple, la requête SQL doit comporter une clause ORDER BY qui spécifie au moins les champs *firstname* et *lastname*. Si l'ensemble des résultats n'est pas correctement ordonné, des éléments en double peuvent apparaître dans le résultat.

3. Connexion à la base de données

Connecteur nécessite des pilotes JDBC pour établir des connexions aux bases de données SQL.

Les pilotes pour **PostgreSQL**, **Microsoft SQL Server**, **MariaDB** et **OracleDB** sont inclus dans Connecteur UXP 2.

Les pilotes JDBC inclus sont des logiciels tiers distribués sous différentes licences.

Si vous avez besoin d'aide pour d'autres bases de données, veuillez contacter le service clientèle.

Tableau 4. Pilotes JDBC inclus dans Connecteur

Base de données	Pilote JDBC	Licence
PostgreSQL	PostgreSQL JDBC v42.7.5	Licence BSD à deux clauses
Microsoft SQL Server	Pilote Microsoft JDBC pour SQL Server v12.8.1	Licence MIT
MariaDB	MariaDB Connector/J v3.5.2	GNU LGPL
OracleDB	Pilote Oracle JDBC v23.26.0.0.0	Oracle FUTC

3.1. Configurer la connexion à la base de données

Avant de pouvoir mettre en œuvre une API REST, la connexion à la base de données que le point de terminaison du service utilisera doit être configurée dans Connecteur. Cette opération peut être effectuée sur la page **Connexions de la base de données**.

Le bouton **Nouvelle connexion** ouvre la fenêtre **Détails de la connexion à la base de données** :

Database Connection Details

Database Name: *

URL: *

Username:

Password:

SAVE

TEST CONNECTION

CANCEL

Figure 6. Fenêtre Détails de la connexion à la base de données

Les champs de la fenêtre **Détails de la connexion à la base de données** ont les significations suivantes :

- Nom de la base de données — nom unique qui sera utilisé pour identifier la connexion.
- URL — URL de connexion JDBC qui sera utilisée pour se connecter à la base de données. Reportez-vous au tableau [Formats d'URL JDBC pour les bases de données](#) pour connaître le format d'URL des pilotes de base de données pris en charge par Connecteur.
- Nom d'utilisateur — nom d'utilisateur qui sera utilisé lors de la connexion à la base de données.
- Mot de passe — mot de passe facultatif, s'il est nécessaire pour se connecter à la base de données.

Vous pouvez tester la connexion à la base de données avant et après avoir enregistré la nouvelle connexion à la base de données.

Tableau 5. Formats d'URL JDBC pour les bases de données

Base de données	Format de l'URL
PostgreSQL	<code>jdbc:postgresql://hostName[:portNumber][/databaseName]</code>
Microsoft SQL Server	<code>jdbc:sqlserver://hostName[:portNumber][/databaseName];sendTimeAsDatetime=false;</code>
MariaDB	<code>jdbc:mariadb://hostName[:portNumber][/databaseName]</code>

Base de données**Format de l'URL**

OracleDB	jdbc:oracle:thin:@//hostName[:portNumber][/databaseName]
----------	--



Microsoft SQL Server exige que la propriété `sendTimeAsDatetime=false` soit définie dans l'URL JDBC. Sinon, le type de données `Time` sera inutilisable avec MSSQL car les données seront toujours envoyées sous la forme `Datetime`.

Toutes les connexions de base de données configurées sont affichées dans la page **Connexions à la base de données** :

Figure 7. Page Connexions à la base de données

Dans la page **Connexions à la base de données**, chaque connexion à la base de données existante possède les attributs et commandes suivants :

- Nom de la base de données — nom unique qui sera utilisé pour identifier la connexion.
- Nombre de points de terminaison — nombre de points de terminaison qui utilisent la connexion à la base de données.
- Le bouton **Afficher les points de terminaison** — ouvre la fenêtre **Afficher les points de terminaison**, dans laquelle s'affichent les points de terminaison utilisant la connexion à la base de données.
 - Les points de terminaison activés et désactivés sont affichés.
 - Pour chaque point de terminaison, le chemin d'accès, l'identifiant et la connexion sont affichés.
 - Vous pouvez rechercher un point de terminaison par son chemin d'accès et son identifiant en saisissant sa valeur partielle ou complète.
- Le bouton **Modifier** — ouvre la fenêtre **Détails de la connexion à la base de données**, dans laquelle la connexion à la base de données peut être modifiée.
 - Vous pouvez modifier tous les attributs d'une connexion à une base de données (nom de la base de données, URL, nom d'utilisateur et mot de passe).
 - Vous pouvez vérifier si la connexion à la base de données a été établie avec succès.

- Le bouton **Supprimer**—permet de supprimer une connexion à une base de données existante.
 - Les points de terminaison activés qui utilisent la connexion à la base de données sont désactivés si la connexion à la base de données est supprimée.

Sur la page **Connexions à la base de données**, vous pouvez rechercher une connexion à la base de données par son nom en saisissant sa valeur partielle ou complète.



L'utilisateur de la base de données doit avoir les privilèges nécessaires pour exécuter des instructions sur les tables de la base de données qui seront affectées. Par exemple, si le service utilise l'instruction *SELECT* sur la table *person*, l'utilisateur doit disposer du privilège d'utiliser l'instruction *SELECT* sur la table *person*.

3.2. Propriétés de la connexion à la base de données

Temps d'inactivité maximal de la connexion correspond au nombre de secondes pendant lesquelles une connexion peut rester en pool mais inutilisée avant d'être supprimée. Dans Connecteur, la valeur par défaut est de 3 600 secondes ou 1 heure.

Période de test de connexion inactive contrôle la fréquence, en secondes, à laquelle les connexions seront testées. Dans Connecteur, la valeur par défaut est de 300 secondes ou 5 minutes.

Cache de sources de données stocke les configurations et les connexions pour différentes sources de données, ce qui permet au système de les récupérer et de les réutiliser rapidement au lieu d'en créer de nouvelles à chaque fois :

- Taille maximale (`cache.maxSize`)—la taille maximale du cache est de 100 entrées. Cette limite est fixée pour éviter que le cache ne devienne trop grand et ne consomme trop de mémoire.
- Délai d'expiration (`cache.expirationTimeInMs`)—Les connexions dans le cache ont un délai d'expiration de 60 000 millisecondes (60 secondes). Si une connexion est trop ancienne, elle sera rejetée.
- Taux fixe d'expulsion (`cache.evictionFixedRateMs`)—le cache est vérifié toutes les 300 000 millisecondes (300 secondes ou 5 minutes) afin d'expulser les anciennes connexions. Ce contrôle régulier permet de s'assurer que les connexions obsolètes sont supprimées du cache.

Pour configurer le cache de la source de données, vous devez procéder comme suit :

1. Modifiez les paramètres `cache.maxSize`, `cache.expirationTimeInMs` et `cache.evictionFixedRateMs` dans le fichier approprié avec les privilèges root :
 - `/etc/uxp/connector-v2/connector-config/application.properties` fichier—L'API Connecteur utilise le cache de la source de données pour les fonctionnalités de demande de test et de point de terminaison de test.
 - `/etc/uxp/connector-v2/restsql-config/application.properties`

fichier — L'API REST-SQL utilise le cache de la source de données dans l'échange de messages.

2. Redémarrez le conteneur Docker `uxp-connector` ou `rest-sql`, en fonction du fichier que vous avez édité :

- Consultez la liste des conteneurs Docker :

```
Usage: docker ps [OPTIONS]
```

- Lors du redémarrage du conteneur, il convient de se référer à son nom ou à l'identifiant du conteneur :

```
Usage: docker restart [OPTIONS] CONTAINER [CONTAINER...]
```

3.3. Renouveler la clé de cryptage de la base de données

Lorsqu'une clé de cryptage de base de données est renouvelée, `encryption.key` doit être mis à jour dans les fichiers suivants de configuration de Connecteur :

- `/etc/uxp/connector-v2/connector-config/application.properties`
- `/etc/uxp/connector-v2/restsql-config/application.properties`

Notez que `encryption.key` doit comporter 16 caractères.

Pour que les changements prennent effet, les conteneurs `uxp-connector` et `rest-sql` doivent être redémarrés. En outre, les connexions à la base de données dans Connecteur doivent être réenregistrées.

4. Sécurité des connexions

4.1. Mode de sécurité

4.1.1. Importance d'une connexion sécurisée

Connecteur transmet les données stockées sur une base de données vers le monde extérieur. Il est donc important de contrôler quels systèmes peuvent envoyer des demandes REST à Connecteur.

Selon l'utilisation prévue, un serveur de sécurité sert de point d'accès unique à Connecteur depuis le réseau externe et les points de terminaison de Connecteur sont censés n'être ouverts que sur le réseau local. Cependant, dans certaines situations, il n'est pas possible ou pas assez sûr de cacher Connecteur dans un réseau interne. Par exemple, lorsque Connecteur est hébergé dans un environnement en nuage ou Connecteur doit également être protégé contre les demandes non autorisées provenant du réseau interne.

Pour le contexte, voir l'exemple de diagramme où Connecteur a deux systèmes clients : Le serveur de sécurité UXP pour la communication avec le monde extérieur et Postman pour tester les API REST en interne.

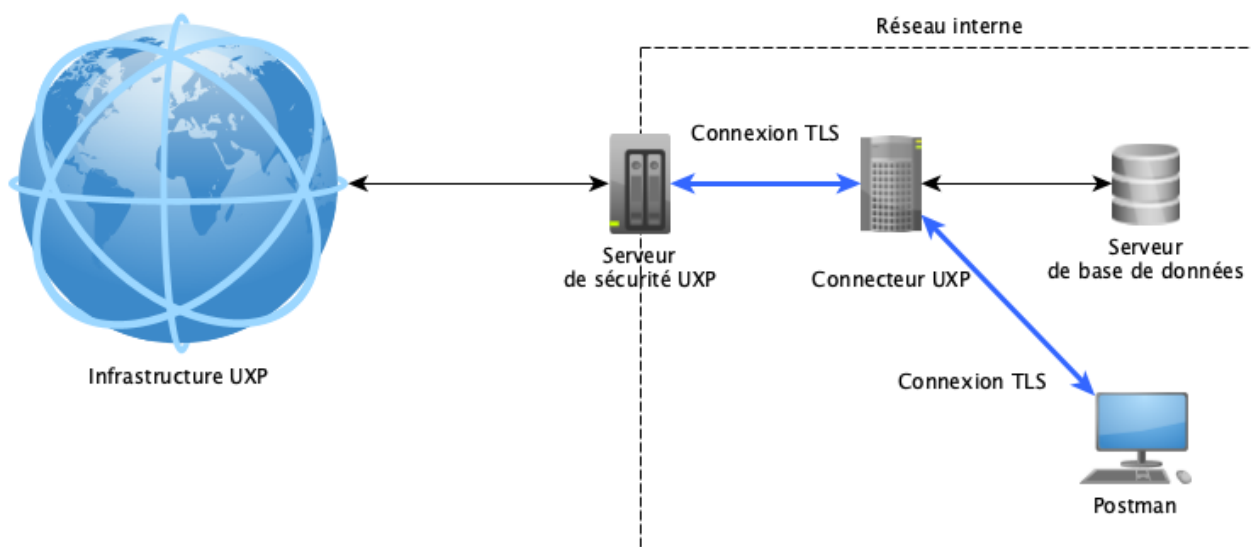


Figure 8. Connecteur avec deux systèmes clients : Serveur de sécurité UXP et Postman

4.1.2. Authentification mutuelle TLS

Connecteur garantit que seuls les systèmes autorisés peuvent adresser des demandes aux API REST en demandant aux systèmes clients de s'authentifier. Connecteur utilise l'authentification mutuelle TLS où les partenaires de communication doivent d'abord échanger leurs certificats numériques, ce qui leur permet d'établir un canal de connexion sécurisé où les partenaires peuvent s'identifier mutuellement.

Pour autoriser la connexion à partir d'un système client, le certificat TLS du système client doit être répertorié comme certificat de confiance dans Connecteur. Voir la section [Ajouter des certificats de confiance](#). S'il est important que le système client authentifie le Connecteur auquel il envoie des demandes, le certificat de Connecteur doit être téléchargé sur le système client. Reportez-vous à la section [Certificat TLS de Connecteur](#).

4.1.3. Visualisation du mode de sécurité de la connexion

Par défaut, l'authentification du client est activée et Connecteur n'autorise les connexions qu'à partir de systèmes qui prouvent qu'ils possèdent un certificat approuvé par Connecteur. Mais le mode de sécurité peut être modifié, voir la section [Désactivation de l'authentification client](#).

Vous pouvez déterminer le mode de sécurité actuel sur la page **Sécurité de la connexion**.

SECURITY MODE

Client authentication is turned ON.

Connector accepts service requests only from systems that present TLS certificates added as trusted certificates.

Figure 9. Mode de sécurité : l'authentification du client est activée

Le mode de sécurité indique si l'authentification du client est activée ou désactivée.

Lorsque l'authentification du client est **activée**, Connecteur n'accepte que les connexions provenant de systèmes présentant un certificat de confiance.

Lorsque l'authentification du client est **désactivée**, Connecteur accepte les demandes sans validation de certificat.

4.1.4. Désactivation de l'authentification client

S'il est nécessaire de désactiver l'authentification du client, vous pouvez le faire en modifiant les fichiers de configuration du Connecteur avec les privilèges de l'administrateur.



N'oubliez pas qu'il est déconseillé de désactiver l'authentification du client. Assurez-vous que les bases de données connectées au Connecteur ne contiennent pas de données sensibles et/ou que le Connecteur n'est accessible qu'à partir d'un réseau interne de confiance.

Pour désactiver l'authentification du client, vous devez remplacer la valeur du paramètre `client-authentication` par `false` dans le fichier `/etc/uxp/connector-v2/connector-config/application.properties`.

1. Modifiez la ligne suivante dans le fichier `application.properties` :

```
client-authentication=false
```

2. Redémarrez le conteneur Docker `uxp-connector` :

- Consultez la liste des conteneurs Docker :

```
Usage: docker ps [OPTIONS]
```

- Lorsque vous redémarrez le conteneur `uxp-connector`, référez-vous à son nom ou à son identifiant :

```
Usage: docker restart [OPTIONS] CONTAINER [CONTAINER...]
```

Pour vérifier si la modification a été effectuée avec succès, consultez la section **Mode de sécurité** de la page **Sécurité de la connexion**.

Activation de l'authentification client

Si vous avez désactivé l'authentification client, vous pouvez la réactiver en remplaçant la valeur `client-authentication` par `true` dans le fichier `/etc/uxp/connector-v2/connector-config/application.properties` et en redémarrant le conteneur `uxp-connector` Docker.

4.2. Ajouter des certificats de confiance

4.2.1. Obtention du certificat

Pour établir une connexion sécurisée dans le cas où le système client est un serveur de sécurité, le certificat interne de celui-ci doit être téléchargé à partir du serveur de sécurité et téléchargé en tant que certificat de confiance dans Connecteur.

Pour les autres types de systèmes clients, si le système client dispose déjà d'une clé privée et d'un certificat, il suffit de télécharger le certificat dans Connecteur.

Si vous n'avez pas de magasin de clés avec la clé privée et le certificat, par exemple pour Postman, vous pouvez générer votre magasin de clés dans Connecteur. Voir [Générer un magasin de clés pour le système client](#).

4.2.2. Certificats de confiance

Le tableau **Certificats TLS de confiance** comporte les colonnes et commandes suivantes :

- **Sujet**—système propriétaire du certificat, nom commun du sujet du certificat. Pour le serveur de sécurité, le sujet est le nom d'hôte du serveur de sécurité.
- **Hash (SHA-256)**—identifiant unique du certificat.
- **État**—Les certificats peuvent être désactivés pour interdire temporairement l'accès à ce système.
- **Bouton Supprimer**—supprime définitivement le certificat. Le système client perd l'accès aux API REST.

- Bouton Désactiver — désactive le certificat. Permet d'interdire temporairement l'accès à ce système sans qu'il soit nécessaire de supprimer et de recharger le certificat ultérieurement.

TRUSTED TLS CERTIFICATE			GENERATE KEYSTORE	ADD CERTIFICATE
Subject	Hash (SHA-256)	Status		
CN=postman	15:0F:5C:86:A5:31:0E:9D:A5:38:9C:5C:91:96:95:C8:DE:DE:26:2A:9D:06:87:84:AE:A3:02:56:D5:EA:BE:EC	Enabled	<input type="checkbox"/>	
CN=uxp-ssl.cloud.example.com	54:62:69:57:0F:EF:3D:2F:2E:0F:CE:36:F6:B9:19:4B:DC:7E:47:B9:F4:5D:E3:69:D0:BF:24:73:8F:1C:93:01	Enabled	<input type="checkbox"/>	

Figure 10. Certificats de confiance

4.2.3. Télécharger un certificat de confiance

Sur la page **Sécurité de la connexion**, sélectionnez **Ajouter un certificat** et téléchargez un fichier de certificat (les formats PEM et DER sont tous deux acceptés).

Vous pouvez choisir d'activer ou de désactiver le certificat avant de l'enregistrer.

Après avoir téléchargé un certificat, celui-ci apparaît dans le tableau des **certificats TLS de confiance**.

4.2.4. Générer un magasin de clés pour le système client

Si, à des fins de test ou de développement, il est nécessaire de faire des demandes d'API REST à partir d'un système qui n'est pas un serveur de sécurité, par exemple l'outil de test Postman, vous pouvez générer le magasin de clés pour ce système dans Connecteur.



Le serveur de sécurité génère lui-même son magasin de clés et son certificat. Téléchargez le certificat à partir du serveur de sécurité et téléchargez-le dans le Connecteur.

Sur la page **Sécurité de la connexion**, sélectionnez **Générer un magasin de clés**.

1. Saisissez le nom du sujet pour identifier le système et le certificat dans la liste des certificats TLS approuvés.
2. Appuyez sur **Générer**.
Connecteur génère un magasin de clés et enregistre le certificat correspondant.
3. Copiez et enregistrez le mot de passe du magasin de clés.
4. Téléchargez et enregistrez le magasin de clés.
5. Fermez la boîte de dialogue.

Dans un système client (pour Postman, voir les instructions détaillées ci-dessous) :

6. Téléchargez le magasin de clés.
7. Saisissez le mot de passe copié pour activer le magasin de clés.

Téléchargez le magasin de clés vers Postman

1. Dans Postman, sélectionnez l'icône des paramètres dans l'en-tête et sélectionnez **Settings**.
2. Sélectionnez l'onglet **Certificates**.
3. Sélectionnez **Add Certificate**.
4. Entrez le domaine **Host** pour le certificat (n'incluez pas le protocole).
5. Entrez le **numéro de port personnalisé (4400)** à associer au domaine. Si vous ne spécifiez pas de port, Postman utilise le port HTTPS par défaut (443).
6. Sélectionnez **PFX file** pour votre certificat.
7. Saisissez le **Passphrase**.
8. Sélectionnez **Add**.

Postman utilisera le magasin de clés pour toutes les demandes effectuées par votre installation Postman.

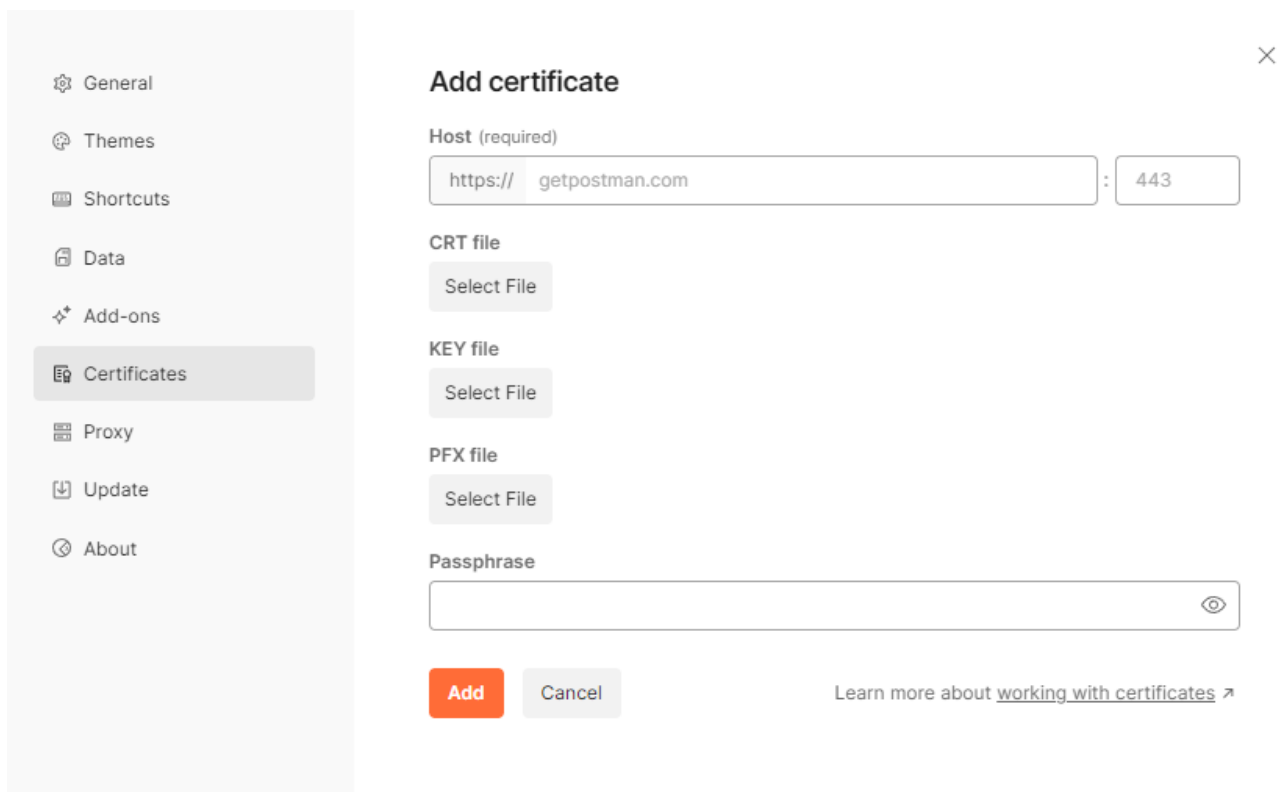


Figure 11. Certificats Postman

4.3. Certificat TLS de Connecteur

Si un système client doit vérifier qu'il envoie des demandes REST à un Connecteur de confiance, le certificat du Connecteur doit être téléchargé sur le système client.

Sur le serveur de sécurité, l'exigence de vérification de l'identité de Connecteur est configurée par API REST.

Si le type de connexion HTTPS est sélectionné sur le serveur de sécurité, le serveur de sécurité doit disposer du certificat de Connecteur fournissant ce service pour authentifier celui-ci.

CONNECTOR TLS CERTIFICATE

DOWNLOAD

Subject

uxp-connector.cloud.example.com

Hash (SHA-256)

53:9A:AA:B0:E6:A4:A5:DC:25:8C:17:9B:68:57:97:27:B7:51:46:DF:BF:07:88:B2:CD:E2:66:17:7A:C7:79:71

Figure 12. Certificat TLS de Connecteur

Télécharger le certificat de Connecteur vers un serveur de sécurité

1. Sur la page **Sécurité de la connexion** de Connecteur, téléchargez le certificat de Connecteur.
2. Sur le serveur de sécurité, choisissez le client du serveur de sécurité qui sera le fournisseur des services créés dans Connecteur.
3. Naviguez jusqu'à l'onglet **Systèmes d'information** de ce client du serveur de sécurité.
4. Téléchargez le certificat de Connecteur dans la section **Certificats TLS internes des systèmes d'information**.

Télécharger le certificat de Connecteur vers un autre système

Téléchargez le certificat vers tout autre système ou outil de développement en suivant les instructions de l'outil en question.

Il n'est pas nécessaire de télécharger le certificat de Connecteur dans Postman.

5. Mettre en œuvre les API REST

5.1. Création d'un nouveau service

Un nouveau service peut être créé sur la page **Services**.

Le bouton **Nouveau service** ouvre la vue **Nouveau service** :

The screenshot shows the 'NEW SERVICE' form. On the left is a sidebar with 'CONFIGURATION' and a list of items: 'Services' (highlighted), 'Database Connections', 'Endpoints', 'Connection Security', and 'License'. Below this is the 'UXP by CYBERNETICA' logo. The main area is titled 'NEW SERVICE' and has a 'username' user indicator. At the top left of the main area is a '← BACK TO SERVICES' button. The form is divided into sections: 'SERVICE INFO' with four input fields ('Service Title: *', 'Service Code: *', 'Service Version:', 'Summary:'), 'ENDPOINTS' with 'No endpoints added' text and an '+ ADD ENDPOINTS' button, and a bottom section with a 'SAVE' button.

Figure 13. Vue nouveau service

La vue **Nouveau service** contient les champs suivants :

- Titre du service — titre du service dans OpenAPI. Ce champ est obligatoire.
- Code de service et version du service — ces champs identifient le service et leur combinaison doit être unique. Un service identifié par un certain code peut avoir plusieurs versions. Le champ du code de service est obligatoire, mais celui de la version du service est facultatif.
- Résumé — résumé du service dans OpenAPI. Ce champ est facultatif. La longueur maximale du champ est de 2 000 caractères.

Le bouton **Ajouter des points de terminaison** dans la vue **Nouveau service** ouvre la fenêtre **Ajouter des points de terminaison** :

Add Endpoints

Endpoint Path

Endpoint Identifier

Search...

Search...

	Endpoint Path	Endpoint Identifier	Connection
<input type="checkbox"/>	path1	endpoint1	connection1
<input type="checkbox"/>	path2	endpoint2	connection2
<input type="checkbox"/>	path3	endpoint3	connection3
<input type="checkbox"/>	path4	endpoint4	connection4

Per page

5

1 of 1

<

>

ADD SELECTED

CANCEL

Figure 14. Fenêtre Ajouter des points de terminaison

Dans la fenêtre **Ajouter des points de terminaison**, les points de terminaison activés et désactivés peuvent être ajoutés au service. Il n'est pas possible d'ajouter plusieurs points de terminaison avec le même chemin d'accès au même service.

Dans la section **Points de terminaison** de la vue **Nouveau service**, le bouton **Supprimer** situé à la fin d'un point de terminaison vous permet de supprimer ce point de terminaison du service.

Tous les services créés sont affichés dans la page **Services** :

CONFIGURATION

Services

Database Connections

Endpoints

Connection Security

License

UXP by CYBERNETICA

SERVICES

NEW SERVICE

username

Service Code

Service Version

Title

Search...

Search...

Search...

URL

Search...

Service Code	Service Version	Title	URL	Added/Refreshed	
service1		service example 1	https://host:port/service1/	2024-07-11 05:26:14 (UTC)	<div><div></div><div></div></div>
service2	version1	service example 2	https://host:port/service2/version1/	2024-07-18 09:30:19 (UTC)	<div><div></div><div></div></div>
service3		service example 3	https://host:port/service3/	2024-06-11 08:02:19 (UTC)	<div><div></div><div></div></div>

Figure 15. Page Service

Sur la page **Services**, chaque service existant dispose des attributs et commandes suivants :

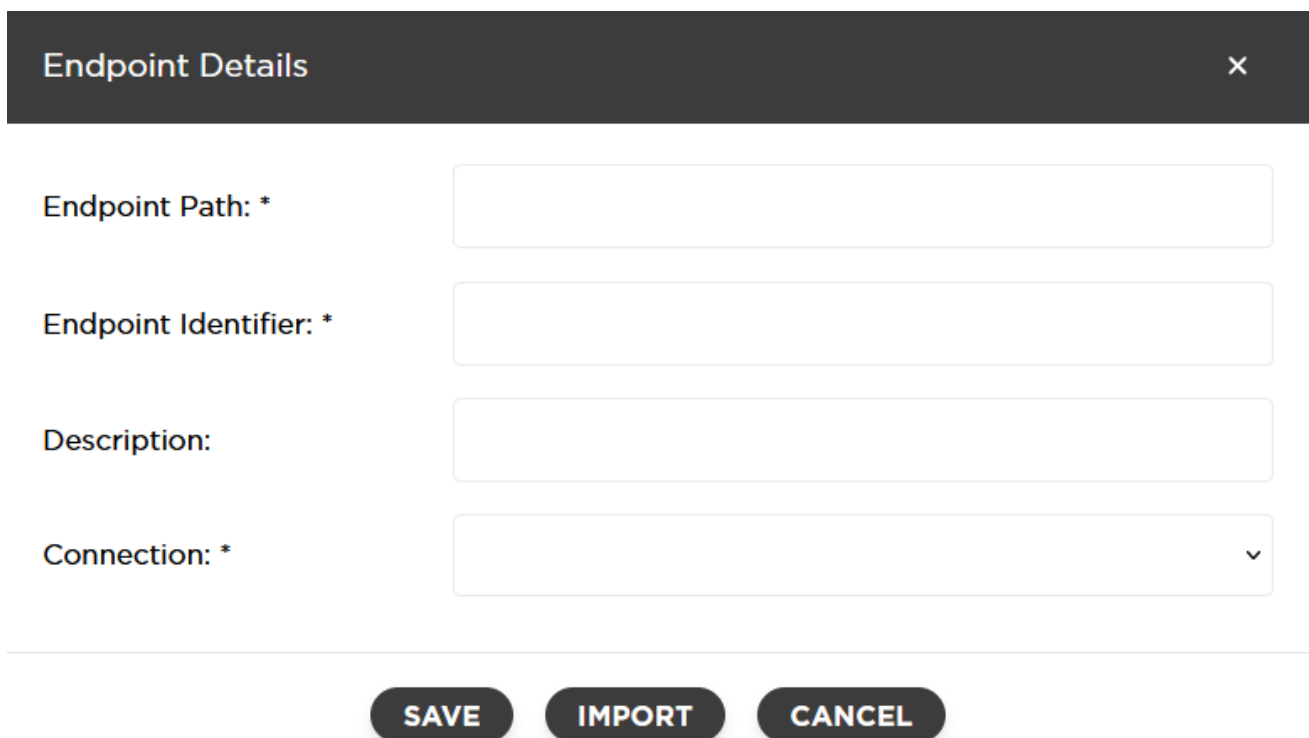
- Code de service et version de service — Combinaison unique utilisée pour identifier le service.
- Titre — titre du service dans OpenAPI.
- URL du service — l'URL du service.
- Ajouté / Actualisé — horodatage indiquant la date d'ajout ou de dernière mise à jour du service.
- Le bouton **Modifier** — ouvre la vue détaillée du service, où celui-ci peut être modifié :
 - Vous pouvez modifier tous les attributs du service (titre du service, code du service, version du service et résumé).
 - Vous pouvez ajouter des points de terminaison au service et en supprimer.
 - Vous pouvez consulter la description du service et l'URL OpenAPI.
- Le bouton **Supprimer** — vous permet de supprimer un service existant.

Sur la page **Services**, vous pouvez rechercher un service par son code, sa version, son titre et son URL en saisissant une partie ou la totalité de ces informations.

5.2. Créer un nouveau point de terminaison

Un nouveau point de terminaison peut être créé sur la page **Points de terminaison**.

Le bouton **Nouveau point de terminaison** ouvre la fenêtre **Détails du point de terminaison** :



The image shows a dialog box titled "Endpoint Details" with a close button (X) in the top right corner. The dialog contains four input fields:

- Endpoint Path: *** (text input)
- Endpoint Identifier: *** (text input)
- Description:** (text input)
- Connection: *** (dropdown menu with a downward arrow)

At the bottom of the dialog, there are three buttons: **SAVE**, **IMPORT**, and **CANCEL**.

Figure 16. Fenêtre Détails du point de terminaison

La fenêtre **Détails du point de terminaison** contient les champs suivants :

- **Chemin d'accès au point de terminaison**—Le chemin d'accès au point de terminaison, associé à l'URL du service, permet aux clients d'accéder à l'API REST. Ce champ est obligatoire.
- **Identifiant du point de terminaison**—identifiant du point de terminaison dans OpenAPI. Ce champ est obligatoire et doit être unique.
- **Description**—description du point de terminaison dans OpenAPI. Ce champ est facultatif. La longueur maximale du champ est de 2 000 caractères.
- **Connexion**—connexion à la base de données utilisée par le point de terminaison. Vous pouvez choisir parmi les connexions à la base de données qui ont été configurées dans Connecteur.

Dans la fenêtre **Détails du point de terminaison**, vous pouvez également importer un point de terminaison. Reportez-vous à la section [Exporter et importer des points de terminaison](#).

Tous les points de terminaison créés sont affichés sur la page **Points de terminaison** :

Endpoint Path	Endpoint Identifier	Connection	Operation	Modified At
/path1	endpoint1	connection1	GET HEAD POST PUT PATCH DELETE	2024-06-20 05:39:06 (UTC)
/path2	endpoint2	connection2	GET HEAD POST PUT PATCH DELETE	2024-06-07 04:31:13 (UTC)
/path3	endpoint3	connection3	GET HEAD POST PUT PATCH DELETE	2024-06-07 10:03:55 (UTC)
/path4	endpoint4	connection4	GET HEAD POST PUT PATCH DELETE	2024-06-07 10:04:31 (UTC)

Figure 17. Page Points de terminaison

Sur la page **Points de terminaison**, chaque point de terminaison existant possède les attributs et commandes suivants :

- **Le chemin d'accès au point de terminaison**—associé à l'URL du service, permet aux clients d'accéder à l'API REST.
- **Identifiant du point de terminaison**—identifiant unique du point de terminaison dans OpenAPI.
- **Connexion**—connexion à la base de données utilisée par le point de terminaison.
- **Modifié à**—horodatage de l'ajout ou de la dernière mise à jour du point de terminaison.
- **Opérations sur les points d'extrémité (GET, HEAD, POST, PUT, PATCH, DELETE)** :
 - Pour configurer une opération de point de terminaison, cliquez sur l'opération. Reportez-vous à la section [Configurer les opérations de point de terminaison](#).
 - Les opérations de point de terminaison qui n'ont pas été configurées s'affichent en noir. Les opérations configurées des points de terminaison activés sont affichées en vert. Les opérations configurées des points de terminaison désactivés sont affichées en rouge.
- **Activer/désactiver le point de terminaison**—permet d'activer ou de désactiver le point de terminaison.

- Les points de terminaison désactivés ne peuvent pas être utilisés par les clients du service.
- Les points de terminaison désactivés sont affichés en rouge.
- Le bouton **Modifier** — ouvre la fenêtre **Détails du point de terminaison**, dans laquelle le point de terminaison peut être modifié :
 - Vous pouvez modifier tous les attributs du point de terminaison (chemin d'accès, identifiant, description, connexion).
- Le bouton **Supprimer** — vous permet de supprimer un point de terminaison existant.
- Le bouton **Copier** — vous permet de copier le point de terminaison. Reportez-vous à la section [Copier des points de terminaison](#).
- Le bouton **Exporter** — vous permet d'exporter le point de terminaison. Reportez-vous à la section [Exporter et importer des points de terminaison](#).

Sur la page **Points de terminaison**, vous pouvez rechercher un point de terminaison à l'aide de son chemin d'accès et de son identifiant en saisissant une partie ou la totalité de leur valeur.

5.3. Configurer les opérations de point de terminaison

Vous pouvez configurer une nouvelle opération de point de terminaison ou modifier une opération existante en sélectionnant la méthode d'opération de point de terminaison dans la vue **Points de terminaison**. Les méthodes d'opération sélectionnables sont GET, HEAD, POST, PUT, PATCH et DELETE.

La vue permettant de configurer l'opération de point de terminaison contient les onglets **Détails**, **Paramètres d'entrée**, **Paramètres de sortie** et **Tester le point de terminaison**, ainsi que le bouton **Supprimer le point de terminaison**. L'objectif des quatre onglets est décrit dans les sections suivantes.

5.3.1. Spécification des détails des opérations des points de terminaison

/POST EXAMPLEIDENTIFIER

← BACK TO ENDPOINTS

Details
Input Parameters
Output Parameters
Test Endpoint

Endpoint Path:

Summary:

Query: *

SAVE

Figure 18. Onglet Détails de la vue Opération du point de terminaison

L'onglet **Détails** contient les champs suivants :

- Chemin d'accès au point de terminaison — chemin d'accès au point de terminaison qui a été configuré dans la fenêtre **Détails du point de terminaison**. Vous ne pouvez pas modifier ce champ lors de la configuration de l'opération du point de terminaison.
- Résumé — description de l'opération du point de terminaison dans OpenAPI. Ce champ est facultatif. La longueur maximale du champ est de 2 000 caractères.
- Demande — Instruction SQL que le point de terminaison exécute sur la base de données lorsqu'il est utilisé. Ce champ est obligatoire pour l'interrogation de la base de données.

Pour distinguer les paramètres d'entrée dans l'instruction SQL, utilisez les deux points (:) au début du nom du paramètre. Par exemple :country. Ces paramètres seront utilisés ultérieurement lors de la création du mappage d'entrée pour le point de terminaison.

5.3.2. Spécification des paramètres d'entrée de l'opération de point de terminaison

L'onglet **Paramètres d'entrée** permet à l'utilisateur de modifier le mappage des paramètres d'entrée utilisé lors de la conversion entre le message REST entrant et la demande de base de données. Le processus de création d'un mappage de paramètres d'entrée commence par l'analyse des paramètres nommés de la demande configurée pour le point de terminaison.

/POST EXAMPLEIDENTIFIER

← BACK TO ENDPOINTS

Details
↓ Input Parameters
↑ Output Parameters
Test Endpoint

READ FROM QUERY
↑
↓
+
🗑️
☰

Name	Parameter Type	Data Type	Optional	Repeat
No records found				

SAVE

Figure 19. Onglet Paramètres d'entrée de la vue Opération de point de terminaison

L'onglet **Paramètres d'entrée** contient les contrôles suivants pour construire le mappage d'entrée :

- Le bouton **Lire à partir de la demande**—analyse la demande de base de données configurée dans l'onglet **Détails** et insère un champ de mappage pour chaque paramètre nommé trouvé dans la demande.
- Commandes du mappage—pour plus de détails, voir [Spécification du mappage des paramètres](#).

Un élément du mappage d'entrée (tel qu'illustré à la figure 19) possède les attributs suivants :

- Nom — nom du paramètre que cet élément représente tel qu'il apparaît dans le JSON.
- Type de paramètre — type de paramètre représenté par cet élément, indiquant si le paramètre d'entrée est inclus dans l'URL (Query Param) ou dans le corps JSON (Request Body) de la demande REST. Pour les méthodes d'opération GET, HEAD et DELETE, le type de paramètre par défaut est Query Param. Pour les méthodes d'opération POST, PUT et PATCH, le type de paramètre par défaut est Request Body. Pour les méthodes d'opération GET et HEAD, il n'est pas possible de changer le type de paramètre en Request Body.
- Type de données — Type de données du paramètre représenté par cet élément, sélectionnable uniquement pour les paramètres nommés analysés à partir de la demande de base de données. Voir la section [Types de données](#) pour les types pris en charge par Connecteur.
- Facultatif — spécifie que le paramètre ne doit pas nécessairement être présent dans la demande REST entrante. Si le message ne contient pas de paramètre de la demande de la base de données, « null » est utilisé.
- Répétition — spécifie si le paramètre représenté par cet élément peut apparaître plusieurs fois dans le message REST entrant.

5.3.3. Spécification des paramètres de sortie de l'opération du point de terminaison

L'onglet **Paramètres de sortie** permet à l'utilisateur de modifier le mappage des paramètres de sortie utilisé lors de la conversion entre la table des résultats de la demande de base de données et le message REST de réponse. Le processus de création d'un mappage de paramètres de sortie commence par l'analyse des paramètres nommés à partir de la table de résultats d'une demande de test.

/POST EXAMPLEIDENTIFIER

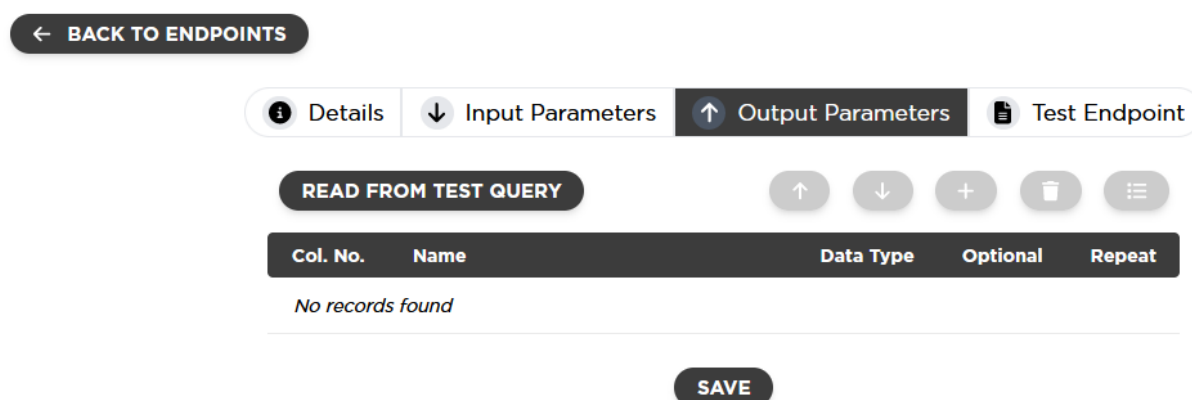


Figure 20. Onglet Paramètres de sortie de la vue Opération de point de terminaison

Le mappage de sortie peut être créé à l'aide de :

- Le bouton **Lire à partir de la demande de test** — ouvre la fenêtre **Exécuter la demande de test** qui permet à l'utilisateur de spécifier les valeurs des paramètres d'entrée pour une demande de test et lit les paramètres de sortie à partir du tableau résultant.

Par défaut, Connecteur annule les demandes de test, ce qui signifie que les modifications apportées à la base de données par les demandes ne sont pas conservées. Vous pouvez également vous engager à conserver les modifications.

- Commandes du mappage — pour plus de détails, voir [Spécification du mappage des paramètres](#).

Un élément du mappage de sortie (tel qu'illustré à la figure 20) possède les attributs suivants :

- Col. No. — index de la colonne dans la table des résultats de la demande de base de données à partir de laquelle le paramètre doit obtenir sa valeur.
- Nom — nom du paramètre que cet élément représente tel qu'il apparaît dans le JSON.
- Type de données — Type de données du paramètre représenté par cet élément, sélectionnable uniquement pour les paramètres nommés analysés à partir de la table des résultats de la demande de base de données. Voir la section [Types de données](#) pour les types pris en charge par Connecteur.

- **Facultatif** — spécifie que la valeur du paramètre ne doit pas nécessairement être présente dans la table des résultats de la demande de base de données. Si la valeur du paramètre est *null*, aucun élément n'est créé dans le message REST de réponse.
- **Répétition** — indique si le paramètre représenté par cet élément est susceptible d'apparaître sur plusieurs lignes dans la table des résultats de la demande de base de données.

Si le mappage des paramètres de sortie n'est pas créé, le point de terminaison renverra toujours un {} vide comme réponse. L'opération HEAD renvoie toujours un corps REST vide.

5.3.4. Tester le fonctionnement du point de terminaison à l'aide de Connecteur

L'onglet **Tester le point de terminaison** permet à l'utilisateur de vérifier si le point de terminaison créé se comporte conformément aux attentes. Le panneau **Paramètres d'entrée** (figure 21) contient des champs de saisie pour tous les paramètres nommés de la demande de service.

Si un ou plusieurs paramètres sont des pièces jointes, le bouton **Ajouter une pièce jointe** s'affiche. Cliquez sur le bouton pour sélectionner le fichier à télécharger. Ensuite, Connecteur vous invite à saisir un identifiant pour la pièce jointe téléchargée.

Pour chaque pièce jointe téléchargée, les paramètres suivants sont affichés :

- **Identifiant** — identifiant unique (parmi les méthodes du point de terminaison) de la pièce jointe téléchargée.
- **Nom du fichier** — nom du fichier téléchargé.
- **Type** — type du fichier téléchargé.

Pour utiliser le fichier téléchargé comme valeur de paramètre, sélectionnez l'identifiant souhaité dans la liste.

En remplissant les paramètres et en appuyant sur le bouton **Exécuter la demande**, vous obtiendrez un aperçu du corps de la requête REST, puis la demande de service sera exécutée sur la connexion de base de données sélectionnée avec les valeurs de paramètres spécifiées.

Le résultat de cette demande sera ensuite converti en JSON et affiché dans le panneau **Réponse**. Les données de la pièce jointe seront énumérées sous la demande ou la réponse JSON. Si la demande échoue, un message d'erreur s'affiche.

Par défaut, Connecteur annule les demandes de test, ce qui signifie que les modifications apportées à la base de données par les demandes ne sont pas conservées. Si vous souhaitez vérifier que la demande de test apporte les modifications souhaitées à la base de données, cochez la case **Valider pour conserver les modifications** avant d'exécuter la demande.

/POST EXAMPLEIDENTIFIER

← BACK TO ENDPOINTS

ⓘ Details
↓ Input Parameters
↑ Output Parameters
📄 Test Endpoint

INPUT PARAMETERS

ParameterName: *

Commit to keep changes ?

☐

RUN QUERY

REQUEST

RESPONSE

SAVE

Figure 21. Onglet Tester le point de terminaison de la vue Opération de point de terminaison

5.3.5. Tester le fonctionnement du point de terminaison à l'aide de Postman

En plus d'utiliser l'onglet **Tester le point de terminaison** pour tester le fonctionnement de celui-ci avec Connecteur, vous pouvez utiliser d'autres applications à des fins de test.

Pour tester un point de terminaison avec Postman :

1. Téléchargez et installez l'outil de test Postman. <https://www.postman.com/>
2. Si l'authentification du client est activée dans Connecteur, générez un magasin de clés dans Connecteur pour établir une connexion sécurisée entre Connecteur et Postman. Téléchargez le magasin de clés vers Postman. Reportez-vous à la section [Générer un](#)

magasin de clés pour le système client.

3. Sélectionnez + dans l'atelier pour ouvrir un nouvel onglet.
4. Sélectionnez la méthode d'opération et saisissez l'URL de la demande. Le cas échéant, ajoutez les paramètres de la demande et leurs valeurs. Fournissez le corps de la demande si elle est configurée.
 - Si vous effectuez une demande directement sur Connecteur UXP 2, entrez https://hostname:4400/{service}/{service_version}/{endpoint_path} pour l'URL de la demande, où *nom d'hôte* et *port* font référence au nom d'hôte et au port du connecteur. Ajoutez la clé *uxp-service* et sa valeur sous l'onglet Headers. La valeur de la clé *uxp-service* doit être au format `{uxp_instance}/{member_class}/{member_code}/{service_code}/{service_version}`, où l'identifiant UXP est tiré de la licence Connecteur.
 - La partie *service_version* n'est obligatoire pour l'URL et l'en-tête *uxp-service* que si elle a été configurée pour le service.
 - Si vous effectuez la demande vers le Connecteur via le serveur de sécurité, entrez https://hostname/restapi/{endpoint_path} pour l'URL de la demande, où *nom d'hôte* fait référence au nom d'hôte du serveur de sécurité. Ajoutez les clés *uxp-client* et *uxp-service* ainsi que leurs valeurs sous l'onglet Headers. La valeur de la clé *uxp-client* doit être au format `{uxp_instance}/{member_class}/{member_code}/{subsystem_code}`. La valeur de la clé *uxp-service* doit être au format `{uxp_instance}/{member_class}/{member_code}/{subsystem_code}/{service_code}/{service_version}`. Pour ces deux valeurs, l'identifiant UXP est extrait de la vue client sur le Serveur de sécurité UXP.
 - La partie *service_version* n'est obligatoire que si elle a été configurée pour le service.
 - Pour fournir le corps de la demande :
 - Si la demande ne comporte pas de pièces jointes :
 - sélectionnez l'option *raw* sous l'onglet *Body* et choisissez le type *JSON* dans le menu déroulant ;
 - saisissez la charge utile JSON dans la zone de texte prévue à cet effet ;
 - sous l'onglet *Headers*, assurez-vous que le Content-Type de la demande principale est *application/json*.
 - Si la demande comporte des pièces jointes :
 - sélectionnez l'option *form-data* sous l'onglet *Body* ;
 - ajoutez une clé nommée *body* et définissez son type sur *Text* ;
 - entrez la charge utile JSON comme valeur pour cette clé et définissez le Content-Type sur *application/json* ;
 - pour fournir une pièce jointe, ajoutez une clé en tant que type *File* : nommez la clé d'après la valeur du paramètre du fichier (nom du fichier) et téléchargez la pièce jointe en cliquant sur *Select Files* ;

- sous l'onglet **Headers**, assurez-vous que le `Content-Type` de la demande principale est `multipart/form-data`.
- Pour fournir des paramètres de requête :
 - Sélectionnez l'onglet **Params**. Ajoutez une clé, en utilisant le nom du paramètre, et ajoutez la valeur souhaitée.
 - Une autre option consiste à ajouter les paramètres de requête directement à l'URL de la demande au format `{parameter_name}={parameter_value}`, en séparant les paramètres par `&`.

5. Soumettez la demande.

5.3.6. Supprimer une opération

Pour supprimer une opération décrite, cliquez sur le bouton **Supprimer** situé en haut de la vue **Détails de l'opération**.

5.4. Exporter et importer des points de terminaison

Les points de terminaison enregistrés dans Connecteur peuvent être exportés sous forme de fichiers JSON et stockés séparément sur le support de données souhaité. Un point de terminaison généré par Connecteur et stocké précédemment peut également être réimporté dans Connecteur et sauvegardé en tant que point de terminaison indépendant.

Pour exporter un point de terminaison :

1. Ouvrez la fenêtre **Exporter un point de terminaison** en cliquant sur le bouton **Exporter** situé à la fin du point de terminaison dans la vue **Points de terminaison**.
2. Sélectionnez zéro ou plusieurs opérations de point de terminaison que vous souhaitez exporter.
3. Connecteur demande un fichier JSON à télécharger.

Pour importer un point de terminaison :

1. Ouvrez la fenêtre **Détails du point de terminaison** en cliquant sur le bouton **Nouveau point de terminaison** dans la vue **Points de terminaison**.
2. Cliquez sur **Importer**.
3. Téléchargez un fichier JSON depuis votre système de fichiers local.
4. Spécifiez les détails du point de terminaison (chemin d'accès, identifiant, description et connexion) et sélectionnez les opérations que vous souhaitez importer avant d'enregistrer le nouveau point de terminaison dans Connecteur.

5.5. Copier des points de terminaison

Un moyen rapide de créer une nouvelle version ou une modifiée d'un point de terminaison est de copier celui-ci :

1. Ouvrez la fenêtre **Copier un point de terminaison** en cliquant sur le bouton **Copier** situé à la fin du point de terminaison dans la vue **Points de terminaison**.
2. Spécifiez les détails du point de terminaison (chemin d'accès, identifiant, description et connexion) et sélectionnez les opérations que vous souhaitez copier avant d'enregistrer le nouveau point de terminaison dans Connecteur.

6. Enregistrer des API REST dans Serveur de sécurité UXP

Pour rendre une API REST créée dans Connecteur disponible via UXP, vous devez publier l'API REST sur un serveur de sécurité.

Si vous avez accès à un serveur de sécurité, vous pouvez publier vous-même l'API REST. Si vous n'avez pas accès à un serveur de sécurité, envoyez les informations nécessaires à l'administrateur du serveur de sécurité de votre organisation.

Contactez l'administrateur du serveur de sécurité de votre organisation et fournissez-lui les informations suivantes :

1. **URL de description OpenAPI** décrivant l'API REST.

Connecteur accepte les demandes de service à <https://hostname:4400/api/services/{serviceCode}/openapi> (ou <https://hostname:4400/api/services/{serviceCode}/openapi?version={serviceVersion}> s'il existe une version de service), où *<nom d'hôte>* est l'adresse de la machine exécutant le serveur d'applications. Dans Connecteur, l'URL de description OpenAPI est affichée dans la vue détaillée du service sous **URL Description OpenAPI**.

2. **Liste des membres UXP et de leurs sous-systèmes** qui doivent avoir accès à l'API REST.

3. **Certificat TLS du Connecteur** pour une connexion sécurisée entre le serveur de sécurité et le Connecteur.

Le certificat de Connecteur peut être téléchargé à partir de la page **Sécurité de la connexion**.

Une fois que l'administrateur du serveur de sécurité a enregistré le service, celui-ci peut être utilisé via UXP en créant un message REST formaté selon la description OpenAPI. Ce message doit ensuite être envoyé à un serveur de sécurité dans l'instance UXP, qui le transmettra au serveur de sécurité où le service a été enregistré. Le message est envoyé à Connecteur à partir de ce serveur de sécurité.

Sur le Serveur de sécurité :

1. L'API REST doit être ajoutée à l'aide de l'**URL Description OpenAPI**. Pour garantir la cohérence, **lorsque vous saisissez le code et la version du service de l'API REST, utilisez les mêmes valeurs que celles spécifiées dans Connecteur**.
2. Les **droits d'accès** doivent être configurés. Le droit d'accès peut être configuré séparément pour chaque point de terminaison.
3. Le **certificat TLS du Connecteur** doit être ajouté au serveur de sécurité. Reportez-vous à [Certificat TLS de Connecteur](#).

7. Informations complémentaires

7.1. Spécification du mappage des paramètres

Les éléments de mappage des entrées/sorties possèdent un certain nombre d'attributs modifiables. Les commandes qui permettent à l'utilisateur d'ajuster le mappage sont décrites dans cette section.



Figure 22. Commandes du mappage

Les onglets **Paramètres d'entrée** et **Paramètres de sortie** disposent des commandes suivantes :

- Déplacer le paramètre sélectionné vers le haut (flèche vers le haut) — réorganise les éléments, en échangeant l'élément sélectionné et l'élément situé au-dessus de celui-ci au même niveau.
- Déplacer le paramètre sélectionné vers le bas — réorganise les éléments, en échangeant l'élément sélectionné et l'élément situé en dessous au même niveau.
- Le paramètre (ou les paramètres) sélectionné(s) par le groupe (+) — ajoute un nouvel élément structurel au-dessus de l'élément (ou des éléments) sélectionné(s), reléguant l'élément (ou les éléments) sélectionné(s) au rang d'enfant de l'élément structurel nouvellement créé.



Lors de la création de mappages de paramètres plus complexes, commencez par l'élément parent le plus élevé, puis passez aux niveaux inférieurs de la hiérarchie dans l'ordre.



Les paramètres dont le type de paramètre est défini sur Query Param ne peuvent pas être regroupés.

- Supprimer le groupe — supprime l'élément structurel du mappage, promouvant tous les éléments imbriqués.
- Afficher les détails du paramètre sélectionné — ouvre la fenêtre des détails de l'élément, qui contient les champs suivants :
 - Nom — nom de l'élément.
 - Description — description de l'élément.

Si le type d'élément est *string*, il comporte des champs supplémentaires.

- Longueur maximale — nombre maximal de caractères que la chaîne doit contenir.
- Le mode — spécifie si la chaîne doit être envoyée en tant que pièce jointe texte ou en

tant que valeur de chaîne normale.

Les pièces jointes de type chaîne utilisent la valeur *Content-Type* `text/plain` et le jeu de caractères par défaut *charset* `UTF-8`.

Si le type d'élément est *decimal*, il comporte des champs supplémentaires.

- Valeur minimale — spécifie la valeur minimale du nombre décimal.
- Valeur maximale — spécifie la valeur maximale du nombre décimal.
- Échelle — spécifie le nombre de chiffres à droite de la virgule décimale dans le nombre décimal. Par exemple, `1` (sans décimales) correspond à `scale = 0`, tandis que `2,45` (deux décimales) correspond à `scale = 2`.

Si le type d'élément est *binary*, il comporte des champs supplémentaires.

- Mode — spécifie si le binaire doit être envoyé en pièce jointe ou sous forme de valeur encodée (base64 ou hexadécimale). Si le type de paramètre est Query Param, il est uniquement possible de choisir d'envoyer le code binaire sous la forme d'une valeur codée (base64 ou hexadécimale).
- Type de contenu — spécifie le type de média de la pièce jointe. Ce champ ne peut être utilisé que pour les paramètres de sortie si le mode de pièce jointe a été sélectionné. Les pièces jointes binaires utilisent la valeur *Content-Type* `application/octet-stream` sauf indication contraire dans le champ Type de contenu. Par exemple, si la base de données renvoie une image JPG, entrez le type de média `image/jpeg`. Voir [la liste complète des types de médias officiels](#).



Dans la version actuelle, Connecteur n'utilise pas les champs supplémentaires de longueur maximale, de valeur minimale, de valeur maximale et d'échelle pour le mappage de sortie.

Il existe des noms de paramètres réservés qui représentent les valeurs contenues dans l'en-tête REST de la demande. Ces noms de paramètres peuvent être utilisés dans la requête (ils seront remplacés par la valeur dans l'en-tête REST) mais les paramètres avec leurs noms ne doivent pas être présents dans le mappage. Le type de tous les paramètres réservés est une *string*. Les noms des paramètres réservés sont les suivants :

- *uxp-client* — spécifie le client du service (instance UXP, classe membre, code membre, code sous-système).
- *uxp-service* — spécifie le service invoqué par le client du service (instance UXP, classe membre, code membre, code sous-système, code service, version service).
- *uxp-userid* — identifiant unique de l'utilisateur dont l'action est à l'origine de la demande.
- *uxp-queryid* — identifiant de demande du message.
- *uxp-transaction-id* — identifiant de transaction du message.
- *uxp-consent-ref* — référence de consentement fournie avec le message.
- *uxp-issue* — identifie l'application, la question ou le document reçu qui est à l'origine de la demande de service.

Le mappage des paramètres d'entrée doit être créé avant que les paramètres de sortie

puissent être déterminés.

7.2. Types de données

Le tableau 6 répertorie tous les types de données pris en charge par Connecteur.

Tableau 6. Types de données

Type dans Connecteur	Type SQL	Type SQL Oracle	Type JSON	Description
integer	INTEGER	NUMBER(10)	integer	Un entier signé de 32 bits. Correspond à Oracle NUMBER avec précision 6-10.
string	CHAR, VARCHAR, LONGVARCHAR, CLOB	VARCHAR2, NVARCHAR2	string	Une séquence de caractères.
boolean	BIT, BOOLEAN	NUMBER(1)	boolean	Type de données logiques. Oracle NUMBER(1) : 0 correspond à False, 1 correspond à True.
byte	TINYINT	NUMBER(3)	integer	Un entier signé de 8 bits. Correspond à Oracle NUMBER avec précision 2-3.
short	SMALLINT	NUMBER(5)	integer	Un entier signé de 16 bits. Correspond à Oracle NUMBER avec précision 4-5.
long	BIGINT	NUMBER(19)	integer	Un entier signé de 64 bits. Correspond à Oracle NUMBER avec précision 11-19.

Type dans Connecteur	Type SQL	Type SQL Oracle	Type JSON	Description
date	DATE	-	string (date)	Stocke les années, les mois et les jours. (Remarque : Oracle DATE est mappé sur Timestamp pour préserver l'heure).
time	TIME	-	string (time)	Stocke les heures, les minutes et les secondes. Oracle n'a pas de type TIME dédié.
timestamp	TIMESTAMP	DATE, TIMESTAMP	string (date-time)	Stocke les années, les mois, les jours, les heures, les minutes et les secondes. Inclut Oracle DATE (qui contient l'heure).
float	REAL	BINARY_FLOAT	number (float)	Un nombre à virgule flottante 32 bits à précision simple.
double	FLOAT, DOUBLE	BINARY_DOUBLE	number (double)	Un nombre à virgule flottante 64 bits à double précision.
decimal	NUMERIC, DECIMAL	NUMBER(38)	string (decimal)	Une valeur décimale à précision fixe. Utilisé pour Oracle NUMBER avec précision > 19 ou échelle > 0.
binary	BINARY, VARBINARY, BLOB, LONGVARBINARY	BLOB, RAW	string (base64encoded, hexencoded or ContentID)	Une séquence d'octets.

8. Gérer les comptes d'utilisateurs

Connecteur utilise **Casdoor** pour gérer les comptes d'utilisateur de l'administrateur Casdoor et de l'administrateur Connecteur. En outre, Connecteur utilise Casdoor pour authentifier les administrateurs de Connecteur lorsqu'ils se connectent à l'interface Web de Connecteur.

8.1. Casdoor

Les comptes Administrateur Casdoor et Administrateur Connecteur sont créés lors de l'installation. Des comptes supplémentaires peuvent être ajoutés manuellement dans Casdoor après l'installation.

Pour créer un nouvel utilisateur

1. Connectez-vous à Casdoor.
2. Dans l'onglet **User Management** du menu principal, sélectionnez le sous-onglet **Users** dans le menu déroulant.
3. Sélectionnez **Add**.
4. Ajoutez les détails sur l'utilisateur :
 - Si vous créez un Administrateur Casdoor, sélectionnez **built-in** dans le champ **Organisation**.
 - Si vous créez un utilisateur Connecteur, sélectionnez l'organisation définie lors de l'installation dans le champ **Organisation**.
5. Sélectionnez **Save & Exit**.



Si vous avez modifié le champ **Name** lors de la création d'un nouvel utilisateur, le mot de passe de l'utilisateur ne peut pas être défini immédiatement et l'utilisateur ne pourra pas se connecter à Connecteur ou à Casdoor. Définissez le mot de passe en modifiant les détails de l'utilisateur **après** sa création.

Si vous souhaitez supprimer un utilisateur de Casdoor, sous le sous-onglet **Users**, recherchez l'utilisateur dans la liste des utilisateurs existants et sélectionnez **Delete** à la fin de sa ligne.

Pour attribuer le rôle **CONNECTOR_ADMINISTRATOR** à l'utilisateur de Connecteur

1. Dans l'onglet **Authorization** du menu principal, sélectionnez le sous-onglet **Roles** dans le menu déroulant.
2. Recherchez le rôle **CONNECTOR_ADMINISTRATOR**, puis cliquez sur son nom ou sélectionnez **Edit** à la fin de sa ligne.
3. Dans le champ **Sub users** de la vue **Edit Role**, ajoutez l'utilisateur à partir d'un menu déroulant.
4. Sélectionnez **Save & Exit**.

Vous n'avez pas besoin d'attribuer de rôles à l'Administrateur Casdoor.

Pour annuler l'attribution d'un rôle à un utilisateur, accédez à la vue **Edit Role**. Dans le champ **Sub users**, localisez l'utilisateur que vous souhaitez supprimer et cliquez sur **x** à côté de son nom d'utilisateur. Sélectionnez **Save & Exit** pour confirmer les modifications.

Si un utilisateur de Connecteur souhaite modifier son mot de passe, il peut utiliser l'option **Forgot password?** sur la page de connexion de Connecteur. Pour activer cette option, l'Administrateur Casdoor doit configurer le fournisseur de messagerie dans la console Casdoor.

Pour ajouter un nouveau fournisseur de messagerie électronique

1. Dans l'onglet **Identity** du menu principal, sélectionnez le sous-onglet **Providers** dans le menu déroulant.
2. Sélectionnez **Add**.
3. Sélectionnez la valeur **Email** pour le paramètre **Category**.
4. Ajoutez les valeurs suivantes pour les paramètres :
 - **Name** — l'identifiant unique que vous souhaitez attribuer à votre fournisseur de messagerie électronique.
 - **Display name** — le nom d'affichage que vous souhaitez donner à votre fournisseur de messagerie électronique.
 - **Organization** — **admin (Shared)**.
 - **Category** — **Email**.
 - **Type** — **Default**.
 - **From address** — Adresse e-mail de l'expéditeur.
 - **From name** — "Connecteur".
 - **Host** — l'adresse du serveur SMTP de votre fournisseur de messagerie électronique.
 - **Port** — le port utilisé pour la communication SMTP.
 - **Email title** — « Code de vérification Casdoor ».
5. Sélectionnez **Save & Exit**.

Pour ajouter le fournisseur de messagerie électronique à l'application Connecteur

1. Dans l'onglet **Identity** du menu principal, sélectionnez le sous-onglet **Applications** dans le menu déroulant.
2. Recherchez l'application **connector** dans la liste des applications existantes, puis cliquez sur le nom de l'application ou sélectionnez **Edit** à la fin de sa ligne.
3. Dans la vue **Edit Application**, ajoutez le fournisseur de messagerie sous le paramètre **Fournisseurs**.
4. Sélectionnez **Save & Exit**.

Pour permettre aux utilisateurs administrateurs Casdoor d'utiliser également l'option **Forgot**

password?, ajoutez également le fournisseur à l'application `app-built-in`.

9. Dépannage

9.1. Surveillance

Connecteur fournit un **service d'information sur l'état** avec deux points de terminaison qui renvoient l'état de santé de Connecteur.

Le **point de terminaison d'échange de messages** renvoie des informations si l'échange de messages via Connecteur fonctionne. Le point de terminaison renvoie également les états individuels des conditions suivantes :

- L'API REST-SQL est en cours d'exécution.
- L'API Connecteur est en cours d'exécution.
- La base de données de Connecteur est en cours d'exécution.
- Redis est en cours d'exécution.
- La licence Connecteur UXP 2 existe et est valide.

Si au moins une condition est fausse, l'échange de messages ne fonctionne pas.

Le **point de terminaison des fonctionnalités administratives** renvoie des informations indiquant si l'exécution des fonctionnalités administratives dans Connecteur se déroule comme prévu. Le point de terminaison renvoie également les états individuels des conditions suivantes qui affectent l'exécution des activités administratives dans Connecteur :

- L'API Connecteur est en cours d'exécution.
- La base de données de Connecteur est en cours d'exécution.
- L'interface utilisateur de Connecteur est en cours d'exécution.
- Redis est en cours d'exécution.
- Casdoor est en cours d'exécution.

Composants qui affectent Connecteur :

L'indisponibilité de l'**API REST-SQL** affecte la fonctionnalité principale de Connecteur, à savoir l'échange de messages, car celui-ci est chargé de recevoir les demandes REST provenant des systèmes d'information clients, de les traiter (en convertissant la demande REST en une instruction SQL et la réponse de la base de données en une réponse REST) et de renvoyer les réponses REST au système d'information client. Cela n'affecte pas les fonctionnalités administratives de Connecteur ou du fournisseur Open ID Connect (OIDC) Casdoor.

Redis est utilisé pour stocker la licence Connecteur UXP 2. Pendant son indisponibilité, il n'est pas possible de télécharger une nouvelle licence dans l'interface utilisateur de Connecteur. Le cache Redis est chargé de conserver les licences, les certificats de confiance et les informations de configuration du service. Ces données sont accessibles par l'API REST-SQL pendant l'échange de messages, ce qui accélère le processus. Si Redis est en panne,

l'échange de messages cesse de fonctionner.

L'**Interface utilisateur de Connecteur**, l'**API de Connecteur** et la **Base de données de Connecteur** permettent ensemble aux utilisateurs d'effectuer des tâches administratives, telles que la configuration des services et des points de terminaison, dans l'interface utilisateur de Connecteur.

L'indisponibilité de l'**interface utilisateur de Connecteur** empêche l'utilisateur de Connecteur d'accéder à l'interface utilisateur de celui-ci, mais n'affecte pas l'échange de messages.

L'**API Connecteur** et la **base de données de Connecteur** sont toutes deux nécessaires au fonctionnement de l'échange de messages par l'intermédiaire de Connecteur.

L'**API Connecteur** contrôle périodiquement Redis pour vérifier la validité de la licence Connecteur UXP 2 et met à jour le cache Redis avec le nouveau résultat. Lors de l'échange de messages, s'il manque des informations nécessaires dans le cache Redis, l'API Connecteur récupère les données requises (par exemple, la configuration des services et des points de terminaison, les certificats de confiance) dans la **base de données de Connecteur**, les renvoie à l'API REST-SQL et les stocke dans le cache Redis en vue de futures demandes. En outre, l'API Connecteur crée des journaux d'audit.

Casdoor est utilisé pour l'authentification dans l'interface utilisateur de Connecteur et la gestion des utilisateurs via la console Casdoor. Si le fournisseur OIDC Casdoor est en panne, les utilisateurs ne peuvent pas se connecter à l'interface utilisateur Connecteur ou à la console de gestion Casdoor.

La validité de la licence Connecteur n'affecte que la fonctionnalité principale d'échange de messages. Si la **licence n'est pas valide**, l'échange de messages ne fonctionne pas, mais il est toujours possible d'exécuter des fonctions administratives dans l'interface utilisateur de Connecteur.

9.1.1. Obtenir les informations d'état

Pour recevoir les informations sur l'état de Connecteur, vous devez envoyer une demande GET au point de terminaison approprié, où *host* est l'adresse de la machine qui exécute le serveur d'application :

- Point de terminaison de l'échange de messages — <https://host:4400/api/status/data-exchange>
- Point de terminaison des fonctionnalités administratives — <https://host:4400/api/status/admin>

9.2. Journaux

Les fichiers journaux aident à résoudre les erreurs qui surviennent et à détecter d'éventuels comportements inattendus. Le tableau suivant répertorie les fichiers journaux liés à UXP dans Connecteur.

La lecture du fichier `/var/log/uxp-connector-v2/connector-v2-installation.log`

nécessite des privilèges root.

Emplacement du journal	Description	Configuration du journal
/var/log/uxp-connector-v2/uxp-connector-audit.log	Enregistrement des actions réussies et échouées de l'utilisateur dans l'interface utilisateur de Connecteur	/etc/uxp/connector-v2/connector-config/connector-logback.xml
/var/log/uxp-connector-v2/uxp-connector-application.log	Enregistrements des demandes adressées au serveur d'application	/etc/uxp/connector-v2/connector-config/connector-logback.xml
/var/log/uxp-connector-v2/uxp-connector-rest.log	Enregistrements du traitement des transactions SQL2REST	/etc/uxp/connector-v2/restsql-config/restsql-logback.xml
/var/log/uxp-connector-v2/connector-v2-installation.log	Enregistrements des actions liées au processus d'installation	N/A
Onglet « Logging & Auditing » (Journalisation et audit) dans l'interface utilisateur Casdoor	Événements d'authentification, activités des utilisateurs et actions au niveau du système dans Casdoor	N/A

9.3. Erreur de mémoire insuffisante

L'une des causes des erreurs de mémoire insuffisante peut être que le nombre de messages UXP entrants est supérieur à ce que la machine virtuelle Java (JVM) peut traiter avec la quantité maximale actuelle de mémoire allouée.

La solution à ce problème consiste à augmenter la mémoire allouée à la JVM. Il est recommandé d'augmenter la mémoire par petits incréments jusqu'à ce que le problème soit résolu.

Étant donné que Connecteur exécute des composants dans des conteneurs Docker, pour modifier la valeur de la mémoire, modifiez la variable d'environnement `JAVA_OPTS` dans le fichier `docker-compose.yml`.

Par exemple, pour fixer la valeur maximale du tas de mémoire à 1 024 Mo :

```
environment:
  -- JAVA_OPTS=-Xmx1024m
```

Après avoir mis à jour le fichier `docker-compose.yml`, redémarrez le(s) conteneur(s) :

- Pour redémarrer tous les conteneurs, exécutez :


```
docker compose down
```

```
docker compose up -d
```

- Pour redémarrer un conteneur spécifique, tel que uxp-connector, exécutez :

```
docker compose stop <uxp-connector>
```

```
docker compose up <uxp-connector> -d
```

Annexe A: Notes de mise à jour

Connecteur UXP 2

1.1.0 (12.2025)

- Prise en charge d'Ubuntu 24, optimisations des performances, prise en charge de la base de données Oracle.

1.0.0 (04.2025)

- Première version de Connecteur UXP 2.

1.0.0-alpha (09.2024)

- Une version alpha de Connecteur UXP 2.