pitney bowes

Big Data Quality SDK

Version 12.0

Guide SDK Qualité des Big Data

Table des matières

1 - Démarrage

In this section

Introduction

SDK qualité des Big Data vous permet de créer, de configurer et d'exécuter des jobs MapReduce, des jobs Spark et des fonctions définies par l'utilisateur Hive pour des opérations Data Quality sur une plate-forme Hadoop.

Grâce au SDK, vous pouvez créer et exécuter des jobs directement sur une plate-forme Hadoop. Vous pouvez ainsi supprimer les délais réseau et exécuter les processus Data Quality distribués en cluster, ce qui a pour effet d'améliorer radicalement les performances.

Les modules pris en charge par SDK qualité des Big Data sont les suivants :

1. Module Advanced Matching
2. Module Data Normalization
3. Module Universal Name
4. Module Universal Addressing

SDK Usage

Ce SDK peut actuellement être utilisé via :

1. API Java : prennent en charge MapReduce et Spark
2. Fonctions Hive définies par l'utilisateur

Reporting

SDK qualité des Big Data fournit une fonctionnalité *Génération de rapports* pour certains jobs. Cette fonctionnalité utilise des compteurs spécifiques pour chaque job pris en charge, ce qui vous permet de contrôler l'efficacité de correspondance obtenue par le job correspondant. Les différents compteurs suivent le nombre d'enregistrements en double, le nombre d'enregistrements uniques et d'autres paramètres utiles pour un job exécuté.

La fonction *Génération de rapports* est actuellement prise en charge dans ces jobs :

- Interflow Match
- Intraflow Match
- Transactional Match
- Open Name Parser
- Validate Address
- Validate Address Global
- Validate Address Loqate

Flux de travail

Pour utiliser le SDK, les composants nécessaires sont les suivants :

Installation SDK qualité des Big Data
Le fichier JAR SDK qualité des Big Data doit être installé sur votre système et disponible pour être utilisé par votre application.

Application client
L'application Java que vous devez créer pour appeler et exécuter les opérations de qualité des données requises à l'aide du SDK. Le fichier JAR SDK qualité des Big Data doit être importé dans votre application Java.

Plate-forme Hadoop
Lors de l'exécution d'un job à l'aide de SDK qualité des Big Data, les données sont tout d'abord lues à partir de la plate-forme Hadoop configurée et selon le traitement approprié, les données de sortie sont écrites dans la plate-forme Hadoop.

Pour ce faire, les détails d'accès de la plate-forme Hadoop doivent être configurés correctement sur votre machine. Pour plus d'informations, reportez-vous à la section **Résumé** à la page 8.

Données de référence
Les données de référence, requises par SDK qualité des Big Data, sont placées sur le cluster Hadoop.

API Java
Pour utiliser l'API Java, vous pouvez choisir de placer les données de référence à l'un des emplacements ci-dessous :

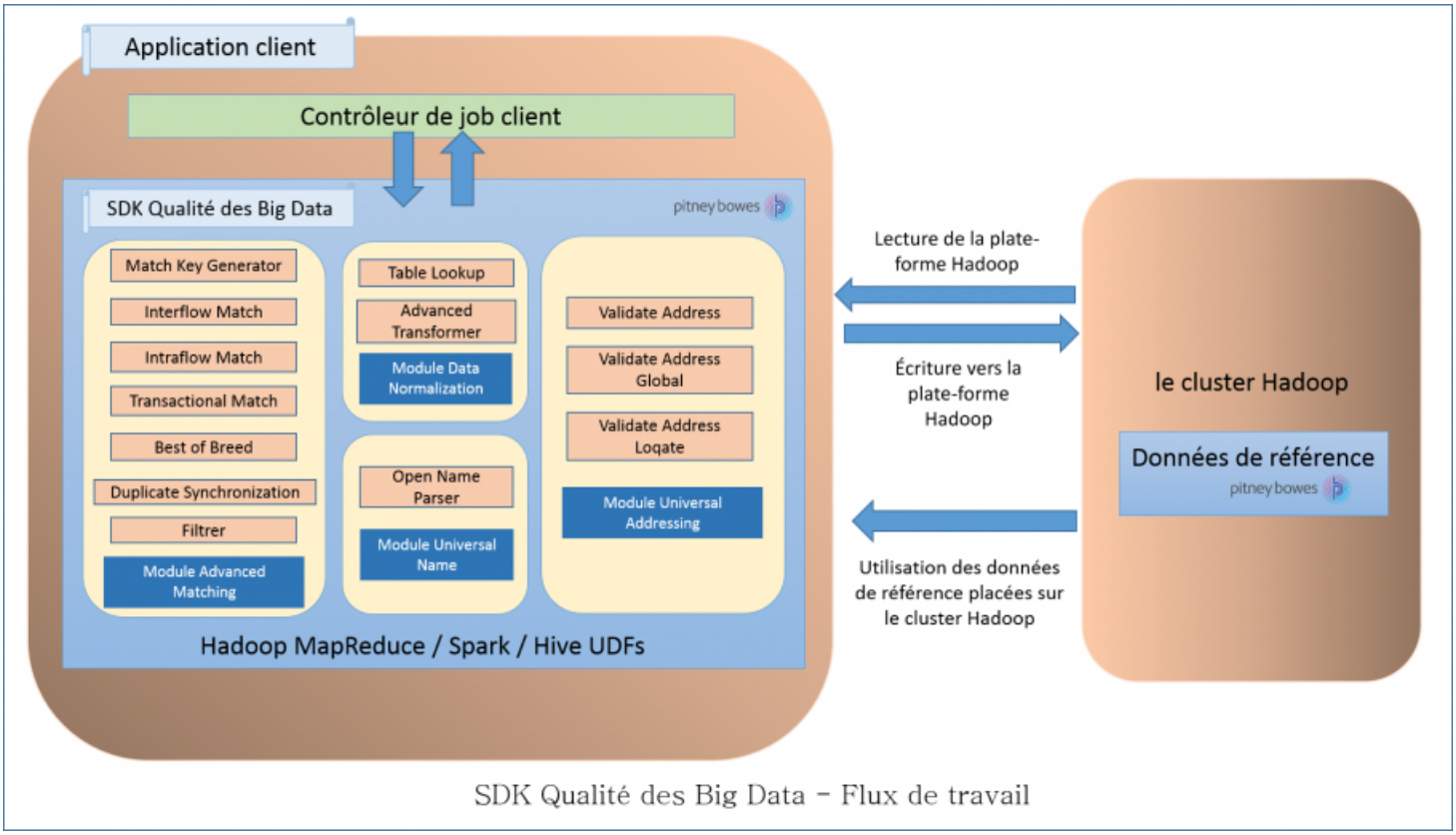
- **Nœuds de données locaux** : les données de référence sont placées sur tous les nœuds de données disponibles dans le cluster.

  Remarque : Il ne s'agit pas d'une méthode infaillible.

- **Hadoop Distributed File System (HDFS)**: Les données de références sont placées dans un répertoire HDFS. Ceci garantit que vos données sont parfaitement sécurisées.

Fonctions définies par l'utilisateur (UDF) Hive
Pour utiliser les UDF Hive, vous devez placer les données de référence sur chaque nœud de données local du cluster.

Remarque : Le SDK permet également une *mise en cache distribuée* pour améliorer les performances.



SDK Qualité des Big Data - Flux de travail

Qui doit utiliser le SDK ?

Le SDK qualité des Big Data est destiné aux :

1. clients souhaitant faire de la qualité des données sur les données résidant sur Hadoop.
2. développeurs Hadoop connaissant la programmation MapReduce ou Spark et désirant créer une solution autour d'un certain cas d'utilisation.
3. développeurs Hadoop souhaitant effectuer des opérations de *nettoyage*, d'*enrichissement*, de *déduplication*, et de *consolidation des données* sur les données existantes.
4. utilisateurs Hive qui ne sont pas familiarisés avec la complexité de MapReduce ou de Spark, mais qui sont à l'aise avec le langage de requête Hive (HQL), dont la syntaxe est semblable à SQL.

2 - Installation

In this section

Configuration minimale requise

Pour une utilisation Hadoop Distributed File System (HDFS) :

1. Java JDK version 1.7 et versions ultérieures.
2. Hadoop version 2.6 et versions ultérieures.
3. Spark 2.0.1 et versions ultérieures.

Pour une utilisation Hive :

1. Hive version 1.2.
2. Un client Hive de votre choix. Par exemple, Beeline.

Remarque : Spectrum™ Technology Platform peut être exécuté uniquement avec des clusters Hadoop.

Mises à jour de systèmes d'exploitation requises

Avant d'installer SDK qualité des Big Data, il est impératif d'appliquer toutes les dernières mises à jour produit disponibles pour votre système d'exploitation, en particulier celles qui corrigent des problèmes liés à Java.

Installation du SDK

Résumé

Utilisez le lien figurant dans votre courrier électronique de bienvenue pour télécharger le fichier zip. Un fichier ZIP de programme d'installation type est téléchargé, nommé comme `BigDataSDK120F0101.zip`.

Extrayez le contenu du fichier ZIP téléchargé sur votre ordinateur pour accéder au programme d'installation et exécutez le programme d'installation, qui vous guide tout au long de la procédure d'installation. Une fois installé, l'outil SDK est ajouté à votre système et placé à l'emplacement défini.

Vous pouvez ensuite importer le fichier JAR SDK qualité des Big Data dans votre projet et commencer à accéder aux API depuis votre ordinateur.

Modules pris en charge

SDK qualité des Big Data prend en charge les modules :

1. Module Advanced Matching
2. Module Data Normalization
3. Module Universal Name
4. Module Universal Addressing

> **Remarque :** Vous devez démarrer le service Acushare avant de créer le premier job *Validate Address* du module Universal Addressing. Pour plus d'informations, reportez-vous à la section **Arrêt du service acushare** à la page 12.

SDK Usage

Le SDK peut actuellement être utilisé via :

1. APIs Java
   - API MapReduce
   - API Spark
2. Fonctions Hive définies par l'utilisateur

Inclusions du programme d'installation

Le fichier ZIP d'installation du SDK contient ces composants :

1. `Readme.txt`
2. `sdkinst.bin` : programme d'installation pour les ordinateurs LINUX.
3. `sdkinst.exe` : programme d'installation pour les ordinateurs WINDOWS.

Installation du SDK sous Windows

Pour installer SDK qualité des Big Data sur un ordinateur Windows, procédez comme suit :

1. Téléchargez le fichier d'installation ZIP SDK qualité des Big Data conformément aux instructions de téléchargement contenues dans votre courrier électronique de bienvenue ou d'annonce de publication.
2. Extrayez tous les fichiers de l'archive à l'emplacement où vous souhaitez installer SDK qualité des Big Data.

3. Accédez au répertoire d'installation et recherchez le programme d'installation nommé *sdkinst.exe*.
4. Double-cliquez sur le fichier `sdkinst.exe`. L'Assistant d'installation s'affiche.
5. Cliquez sur **Suivant**. La fenêtre **Sélectionner le dossier d'installation** s'affiche.

   Ici, vous pouvez spécifier le dossier dans lequel vous souhaitez installer SDK qualité des Big Data. Par exemple, `C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK`.

   a) Cliquez sur le bouton **Sélectionner** pour sélectionner le dossier requis.
   b) Cliquez sur le bouton **Restaurer le dossier par défaut** pour sélectionner le dossier par défaut.

   Avertissement : Si vous sélectionnez un dossier autre que celui par défaut comme répertoire d'installation, assurez-vous que la longueur du chemin d'accès absolu d'installation ne dépasse pas 34 caractères.

   Le chemin d'accès d'installation par défaut de 27 caractères est acceptable :

```
/root/PBSpectrum_BigDataSDK
```

6. Cliquez sur **Suivant**.

   Sur l'écran **Récapitulatif avant installation**, passez en revue les informations d'installation.
7. Cliquez sur **Install**. SDK qualité des Big Data est installé sur votre ordinateur.
8. Cliquez sur **Terminé** à la fin de la procédure d'installation.
9. Vérifiez que vous avez correctement configuré le SDK. Accédez à l'emplacement d'installation du SDK, par exemple `C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK`.

Une fois que vous avez correctement installé le SDK sur votre ordinateur, ces dossiers sont ajoutés au répertoire d'installation :

- `API`
- `Documentation`
- `modules`
- `samples`
- `utilities`

Remarque : Pour utiliser les jobs du module Data Normalization, du module Universal Name ou du module Universal Addressing, vous devez installer les données de référence respectives de chaque module.

Installation du SDK sous Linux

Pour installer SDK qualité des Big Data à l'aide d'une ligne de commande sur un ordinateur Linux, procédez comme suit :

1. Téléchargez SDK qualité des Big Data conformément aux instructions de téléchargement contenues dans votre courrier électronique de bienvenue ou d'annonce de publication.

2. Extrayez tous les fichiers de l'archive à l'emplacement sur le serveur où vous souhaitez installer SDK qualité des Big Data.
3. Enregistrez le répertoire à cet emplacement.
4. Assurez-vous de disposer de l'autorisation `execute` sur les fichiers en saisissant la commande suivante :

   ```
   chmod a+x sdkinst.bin
   ```

5. Exécutez cette commande :

   ```
   ./sdkinst.bin
   ```

   Suivez les invites à l'invite de commande.

6. Lorsque vous y êtes invité, fournissez le répertoire dans lequel vous souhaitez installer le SDK.

   Par exemple, `/home/hadoop/BDQ_InstallPath`.

   Avertissement : Si vous sélectionnez un dossier autre que celui par défaut comme répertoire d'installation, assurez-vous que la longueur du chemin d'accès absolu d'installation ne dépasse pas 34 caractères.

   Le chemin d'accès d'installation par défaut de 27 caractères est acceptable :

   ```
   /root/PBSpectrum_BigDataSDK
   ```

   Un récapitulatif avant installation s'affiche.

7. Passez en revue le sommaire et appuyez sur `ENTER` pour poursuivre l'installation.
8. Consultez le fichier journal d'installation pour vérifier que SDK qualité des Big Data a été correctement installé.
9. Lorsque vous avez terminé, appuyez sur `ENTER` pour terminer et quitter le programme d'installation.

Une fois que vous avez correctement installé le SDK sur votre ordinateur, ces dossiers sont ajoutés au répertoire d'installation :

- `API`
- `Documentation`
- `modules`
- `samples`
- `utilities`

Remarque : Pour utiliser les jobs du module Data Normalization, du module Universal Name ou du module Universal Addressing, vous devez installer les données de référence respectives de chaque module.

Arrêt du service acushare

Avant de créer et d'exécuter le premier job *Validate Address*, vous devez exécuter le service Acushare sur chaque nœud du cluster Hadoop ou Spark.

> **Remarque :** Il s'agit d'une activité obligatoire ponctuelle qui doit être réalisée avant d'exécuter le premier job *Validate Address*.

Sur chaque nœud du cluster :

1. Copiez le script de configuration Acushare `sdkrts.bin` à partir du chemin d'accès à l'installation SDK qualité des Big Data à tout emplacement sur le nœud.

   Avertissement : Sur le serveur SDK, le script d'installation Acushare `sdkrts.bin` se trouve dans `<BDQ SDK_InstallPath>/SDK/utilities/dbloader/aq/runtime/bin`.

2. Connectez-vous au nœud avec des droits d'administrateur ou en tant qu'utilisateur racine.
3. Accédez au chemin d'accès où vous avez copié le script d'installation Acushare `sdkrts.bin`.
4. Assurez-vous de disposer de l'autorisation `execute` sur le fichier en saisissant la commande :

   ```
   chmod a+x sdkrts.bin
   ```

5. Exécutez le fichier du programme d'installation et suivez les invites :

   ```
   ./sdkrts.bin
   ```

6. Lorsque vous y êtes invité, appuyez sur ENTRÉE pour sélectionner le chemin d'accès à l'exécution par défaut `/root/slave_node`, ou saisissez un chemin d'accès absolu de votre choix.

   Important : Le chemin d'accès à l'exécution d'Acushare doit être le même sur tous les nœuds du cluster du job *Validate Address* à exécuter.

   > **Remarque :** Le chemin d'accès sélectionné doit se trouver sur le nœud avant de le préciser ici.

   Le service Acushare se lance automatiquement une fois l'installation correctement effectuée.

7. Sinon, pour démarrer le service Acushare manuellement sur un nœud, accédez à `<Acushare runtime path>/runtime` et exécutez le fichier de script `startrts.sh` avec l'argument `<Acushare runtime path>/runtime`.

Arrêt du service Acushare	Pour arrêter le service Acushare sur un nœud, accédez à `<Acushare runtime path>/runtime` et exécutez le fichier de script `stoprts.sh` avec l'argument `<Acushare runtime path>/runtime`.
Désinstallation du service Acushare	Pour désinstaller le service Acushare de n'importe quel nœud, exécutez le fichier de script `Uninstall_SDKRTS.sh` placé dans `<Acushare runtime path>/Uninstall`.

Données de référence

Présentation des données de référence

Les données de référence de Pitney Bowes définissent un ensemble de valeurs autorisées à utiliser par d'autres champs de données dans votre système afin de garantir la qualité des données. Elles améliorent la cohérence, la précision et la validité des données. Elles vous permettent d'extraire plus de valeur de vos données et d'obtenir des données de confiance du système Big Data.

Par exemple, si vous utilisez les données de référence avec le module Data Normalization, vous pouvez définir une identité client unique au sein de l'entreprise. Des informations client clairement définies constituent la première étape d'amélioration de l'efficacité opérationnelle.

Important : Pour les jobs *Validate Address* et *Vadidate Address Global*, les données de référence doivent être placées sur tous les nœuds de données du cluster Hadoop. Pour le job *Validate Address Loqate*, elles doivent être placées sur un nœud et ce dernier doit ensuite être monté sur tous les autres nœuds de données.

Structure du répertoire d'installation

Dans le répertoire d'installation du SDK, le répertoire `Utilities/dbloader` contient les dossiers enfants :

dataquality Contient les fichiers JAR et de script permettant d'installer les données de référence pour :

- Module Data Normalization
- Module Universal Name

> **Remarque :** Pour plus d'informations, reportez-vous à la section **Utilisation de données de référence : modules Data Normalization et Universal Name** à la page 14.

aq Contient :

- Le script `scripts/server/installdb_unc.sh` permettant d'installer les données de référence. Vous devez exécuter ce script pour installer ou extraire les données.
- Le dossier `runtime` contenant des informations de configuration du service Acushare pour le job *Validate Address* du module Universal Addressing.

> **Remarque :** Pour plus d'informations, reportez-vous à la section **Utilisation de données de référence : module Universal Addressing** à la page 14.

Utilisation de données de référence : modules Data Normalization et Universal Name

Pour utiliser les données de référence pour le **module Data Normalization** et le **module Universal Name**, vous devez exécuter le fichier de script du chargeur de données, par exemple `installdb_dnm`. L'exécution du fichier de script vous permet d'extraire Reference Data sur votre ordinateur.

Assurez-vous que le fichier de script, par exemple, `installerdb_dnm`, et que le fichier JAR se trouvent dans le même dossier.

1. Connectez-vous sur votre ordinateur.
2. Remplacez le répertoire par l'emplacement où vous avez installé le SDK.

   Une fois que vous avez correctement installé SDK qualité des Big Data sur votre ordinateur, vous devez disposer du chargeur Reference Data dans le répertoire `BDQ_InstallPath/SDK/utilities/dbloader/unix/bin`.

3. Exécutez le script du chargeur Reference Data. Par exemple, `installdb_dnm`.
   Une liste numérotée des stages est affichée, et vous êtes invité à sélectionner le stage.
4. Saisissez le numéro correspondant au stage pour lequel vous souhaitez charger les données.
5. Indiquez le chemin d'accès dans lequel les jeux de données de référence sont extraits et placés après le téléchargement.

   Les données de référence saisies sont les tables de base du module Data Normalization, les bases de données de noms fondamentaux et similaires, qui sont nécessaires pour effectuer les jobs des modules Data Normalization et Universal Name.

6. Indique le chemin d'accès au répertoire de sortie. Il s'agit du chemin au niveau duquel les données d'entrée seront extraites.
7. Le système vous demande si vous souhaitez afficher le fichier journal. Faites votre choix.
8. Le système commence à charger les données. Les données sont extraites dans le répertoire de sortie spécifié.

   Remarque : Répétez les étapes pour chaque stage.

Utilisation de données de référence : module Universal Addressing

Pour accéder aux données de référence et les utiliser, vous devez tout d'abord récupérer les données auprès du magasin électronique au format ZIP.

Pour *Validate Address Global* et *Validate Address Loqate*, il suffit d'extraire le contenu du fichier ZIP et les données de référence sont prêtes à être utilisées.

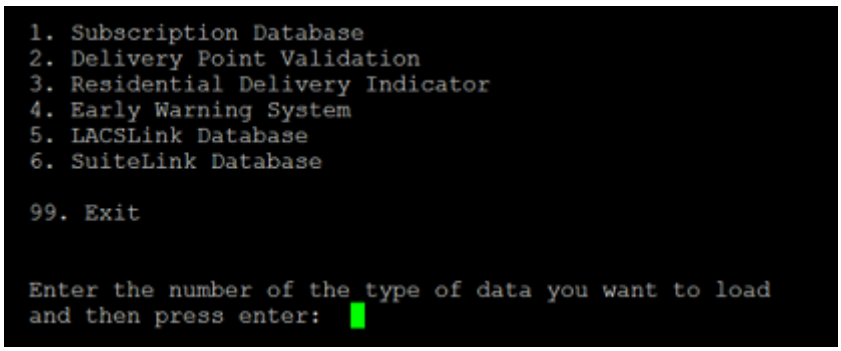
Pour *Validate Address*, suivez les étapes mentionnées pour extraire les données de référence sur votre ordinateur.

Remarque : Assurez-vous que l'autorisation `execute` est accordée au dossier `aq`.

1. Connectez-vous avec des droits d'administrateur ou en tant qu'utilisateur racine.
2. Enregistrez le répertoire à l'emplacement `<BDQ_Installation>/SDK/utilities/dbloader/aq/scripts/server`.
3. Exécutez le script `installdb_unc` via la commande :

   ```
   sh installdb_unc.sh <BDQ_Installation/SDK> <Acushare runtime path>
   ```

   Cette commande vérifie également si le service Acushare est en cours d'exécution. Si ce n'est pas le cas, cette commande lance le service.
4. Après l'exécution de cette commande, les options affichées sont :
   - **US Subscription** : appuyez sur 1 pour afficher une liste des types de chargement de données disponibles, comme mentionné à l'étape suivante.
   - **Exit** : appuyez sur 99 pour quitter.
5. Saisissez le numéro spécifique du type de données que vous souhaitez charger.



6. Indiquez le chemin d'accès aux jeux de données récupérées.

   Les données récupérées depuis le magasin électronique sont disponibles sous forme d'entrée de données de référence, ce qui est nécessaire pour effectuer les jobs du module Universal Addressing. Pour l'emplacement du fichier de sortie, le système affiche le chemin d'accès à la sortie par défaut.
7. L'emplacement du fichier d'entrée et l'emplacement de fichier de sortie sont affichés.

   Saisissez `c` pour continuer, `m` pour modifier le chemin d'accès par défaut ou `q` pour quitter.

```
The Residential Delivery Indicator load environment is currently set to:

Residential Delivery Indicator input file location:
Residential Delivery Indicator output file location: /root/SDK/utilities/dbloader/addressquality/s

Enter c to (c)ontinue
   or m to (m)odify
   or q to (q)uit

   ===>
```

Les données d'entrée sont extraites à l'emplacement de votre fichier de sortie désigné.

8. Le système vous invite à vérifier si votre nouvel emplacement de fichier RDI est correct ou non. Saisissez `y` ou `n`.

```
Please enter full path where you would like to install
the RDI file ==> /root/out

The new RDI file location will be:  /root/out
Is this correct?

Enter (y)es to continue.
      (n)o to try again.

   ===> y

The RDI file output location /root/out does not exist.
Do you want to create it now?

Enter (y)es to create the new RDI file area.
      (n)o to exit.

   ===>
```

Le système commence à charger les données. Les données sont extraites dans le répertoire de sortie spécifié.

Remarque : Répétez les étapes pour le type de données que vous souhaitez charger.

3 - Modules

In this section

Module Advanced Matching

Le module Advanced Matching rapproche des enregistrements entre et/ou à l'intérieur de n'importe quel nombre de fichier d'entrée. Vous pouvez également utiliser le Module Advanced Matching pour rapprocher divers champs, dont nom, adresse, nom et adresse, ou des champs n'ayant pas trait aux noms ou aux adresses, comme le numéro de sécurité sociale ou la date de naissance.

Le module fournit également des jobs pour consolider les enregistrements d'un groupe en sélectionnant un meilleur enregistrement à l'aide d'une configuration appropriée, ou en synchronisant tous les enregistrements d'un groupe donné, ou en filtrant un enregistrement particulier à partir d'un groupe d'enregistrements.

Jobs pris en charge

Le module Advanced Matching de SDK qualité des Big Data prend en charge les jobs :

1. Match Key Generator
2. Interflow Match
   - En générant une Match Key.
   - En utilisant la Match Key existante via les options Grouper par
3. Intraflow Match
   - En générant une Match Key.
   - En utilisant la Match Key existante via les options Grouper par
4. Transactional Match
   - En générant une Match Key.
   - En utilisant la Match Key existante via les options Grouper par
5. Best of Breed
6. Duplicate Synchronization
7. Filtrer

Remarque : Lorsque vous utilisez l'option Grouper par, la Match Key existe déjà dans le fichier d'entrée, à l'aide duquel l'opération Grouper par est effectuée.

Match Key Generator

Match Key Generator crée une clé non unique pour chaque enregistrement, qui peut ensuite être utilisée par les stages de rapprochement pour identifier les groupes d'enregistrements doublons potentiels. Les match keys facilitent la procédure de correspondance en vous permettant de regrouper les enregistrements par match key, puis de ne comparer les enregistrements que dans ces groupes.

La match key est créée à l'aide de règles que vous définissez et est constituée de champs d'entrée. Un algorithme sélectionné s'applique sur chaque champ d'entrée indiqué. Le résultat de chaque algorithme est ensuite concaténé pour créer un seul champ de match key.

Outre la création de match keys, vous pouvez également créer des match keys express, à utiliser ultérieurement dans le flux de données par un stage Intraflow Match ou un stage Interflow Match.

Vous pouvez créer plusieurs match keys et match keys express.

Par exemple, si l'enregistrement entrant est :

Prénom : Fred
Nom de famille : Mertz
Code postal : 21114-1687
Code de genre : M

Et que vous définissez une règle de match key qui génère une match key en combinant des données de l'enregistrement comme suit :

Champ de saisie	Position de début	Longueur
Code postal	1	5
Code postal	7	4
Nom de famille	1	5
Prénom	1	5
Code de genre	1	1

Alors, la clé serait :

211141687MertzFredM

Interflow Match

Interflow Match localise des rapprochements entre des enregistrements de données similaires en croisant deux flux d'enregistrements d'entrée. Le premier flux d'enregistrement est une source d'enregistrements suspects et le second flux est une source d'enregistrements candidats.

À l'aide des critères d'un groupe de rapprochements (par exemple une match key), Interflow Match identifie un groupe d'enregistrements pouvant potentiellement être des doublons d'un enregistrement suspect particulier.

Reporting

Le job Interflow Match vous permet de surveiller les résultats du job. Les compteurs disponibles sont les suivants :

DUPLICATE_COLLECTIONS	Nombre de collections de doublons comprenant un enregistrement suspect et ses enregistrements doublons regroupés par numéro de collection.
EXPRESS_MATCHES	Nombre de rapprochement express effectués dans une collection. On parle de rapprochement express lorsque les contenus d'un champ donné d''un enregistrement suspect et d'un candidat correspondent parfaitement. Il s'agit généralement d'une ExpressMatchKey fournie par Match Key Generator. Lorsqu'un rapprochement express est trouvé, aucune autre opération de traitement n'est conduite afin de déterminer si l'enregistrement suspect et l'enregistrement candidat sont considérés comme doublons.
AVERAGE_SCORE	Score de correspondance moyen de tous les doublons. Les valeurs possibles sont 0-100, où 0 indique une faible correspondance et 100 indique une correspondance exacte.
INPUT_SUSPECTS	Nombre d'enregistrements dans le flux d'entrée que l'outil de mise en correspondance a tenté de rapprocher d'autres enregistrements
SUSPECTS_WITH_DUPLICATES	Nombre de suspects d'entrée mis en correspondance avec au moins un enregistrement candidat.
UNIQUE_SUSPECTS	Nombre de suspects d'entrée qui n'ont été mis en correspondance avec aucun enregistrement candidat.
SUSPECTS_WITH_CANDIDATES	Nombre de suspects d'entrée qui disposaient d'au moins un enregistrement candidat dans leur groupe de correspondance et qui donc présentaient au moins une tentative de correspondance.

SUSPECTS_WITHOUT_CANDIDATES	Nombre de suspects d'entrée qui ne disposaient d'aucun enregistrement candidat dans leur groupe de correspondance et qui donc ne présentaient aucune tentative de correspondance.
TOTAL_DUPLICATE_CANDIDATES	Nombre total de candidats doublons trouvés.
TOTAL_DUPLICATE_SCORE	Score de correspondance total de tous les doublons.

Intraflow Match

Intraflow Match identifie des rapprochements entre des enregistrements de données similaires à l'intérieur d'un seul flux d'entrée. Vous pouvez créer des règles hiérarchiques basées sur des champs que vous avez définis ou créés dans d'autres stages du flux de données.

Reporting

Le job Intraflow Match vous permet de surveiller les résultats du job. Les compteurs disponibles sont les suivants :

INPUT_RECORDS	Nombre d'enregistrements dans le stage de correspondance avant que le tri de correspondance soit effectué.
DUPLICATE_RECORDS	Nombre d'enregistrements en double dans un groupe de correspondance, ce qui peut être soit un suspect ou un enregistrement candidat.
UNIQUE_RECORDS	Nombre d'enregistrement suspect ou candidat qui ne correspond à aucun autre enregistrement dans leur groupe de correspondance respectif. S'il est le seul enregistrement dans un groupe de doublons, un enregistrement de référence est automatiquement unique.
MATCH_GROUPS	(Grouper par) Les enregistrements regroupés par une Match Key.
DUPLICATE_COLLECTIONS	Nombre de collections de doublons comprenant un enregistrement suspect et ses enregistrements doublons regroupés par numéro de collection.
EXPRESS_MATCHES	Nombre de rapprochement express effectués dans une collection. On parle de rapprochement express lorsque les contenus d'un champ donné d''un enregistrement suspect et d'un candidat correspondent parfaitement. Il s'agit généralement d'une ExpressMatchKey fournie par Match Key Generator. Lorsqu'un rapprochement express est trouvé, aucune autre opération de traitement n'est conduite afin de déterminer si l'enregistrement suspect et l'enregistrement candidat sont considérés comme doublons.

AVERAGE_SCORE	Score de correspondance moyen de tous les doublons. Les valeurs possibles sont 0-100, où 0 indique une faible correspondance et 100 indique une correspondance exacte.
TOTAL_DUPLICATES	Nombre total de doublons trouvés.
TOTAL_SCORE	Score de correspondance total de tous les doublons.

Transactional Match

Transactional Match met en correspondance des enregistrements suspects avec des enregistrements candidats d'un groupe d'enregistrements, pour identifier les doublons. Les enregistrements sont d'abord regroupés dans une colonne sélectionnée, puis le premier enregistrement est marqué comme enregistrement suspect. Tous les enregistrements restants du groupe, appelé les enregistrements candidats, sont mis en correspondance par rapport à l'enregistrement suspect.

Si l'enregistrement candidat est un doublon, un numéro de groupe lui est attribué, le type d'enregistrement de rapprochement est étiqueté Doublon, et l'enregistrement est ensuite écrasé. Tous candidats sans rapprochement dans le groupe reçoit un numéro de groupe de 0, est étiqueté Unique, puis écrasé lui-aussi.

Reporting

Le job Transactional Match vous permet de surveiller les résultats du job. Les compteurs disponibles sont les suivants :

AVERAGE_SCORE	Score de correspondance moyen de tous les doublons. Les valeurs possibles sont 0-100, où 0 indique une faible correspondance et 100 indique une correspondance exacte.
INPUT_SUSPECTS	Nombre d'enregistrements dans le flux d'entrée que l'outil de mise en correspondance a tenté de rapprocher d'autres enregistrements
SUSPECTS_WITH_DUPLICATES	Nombre de suspects d'entrée mis en correspondance avec au moins un enregistrement candidat.
UNIQUE_SUSPECTS	Nombre de suspects d'entrée qui n'ont été mis en correspondance avec aucun enregistrement candidat.
SUSPECTS_WITH_CANDIDATES	Nombre de suspects d'entrée qui disposaient d'au moins un enregistrement candidat dans leur groupe de correspondance et qui donc présentaient au moins une tentative de correspondance.
SUSPECTS_WITHOUT_CANDIDATES	Nombre de suspects d'entrée qui ne disposaient d'aucun enregistrement candidat dans leur groupe de correspondance et qui donc ne présentaient aucune tentative de correspondance.
TOTAL_DUPLICATES_SCORE	Score de correspondance total de tous les doublons.

TOTAL_DUPLICATES	Nombre total de doublons trouvés.

Best of Breed

Best of Breed consolide les enregistrements doublons en sélectionnant les meilleures données parmi un groupe d'enregistrements doublons et en créant un nouvel enregistrement consolidé à l'aide des meilleures données. Ce « super » enregistrement s'appelle l'enregistrement Best of Breed. Vous définissez les règles à utiliser lors de la sélection des enregistrements à traiter. Une fois le traitement terminé, l'enregistrement Best of Breed est conservé par le système.

Duplicate Synchronization

Duplicate Synchronization détermine quels champs d'un groupe d'enregistrements doivent être copiés dans les champs correspondants de tous les enregistrements du groupe. Vous pouvez indiquer les règles auxquelles les enregistrements doivent satisfaire afin de copier les données d'un champ vers d'autres enregistrements du groupe. Une fois le traitement effectué, tous les enregistrements du groupe sont conservés.

Filtrer

Le stage Filter conserve ou enlève des enregistrements d'un groupe en fonction des règles que vous indiquez.

Module Data Normalization

Le module Data Normalization examine les termes dans un enregistrement et détermine si le terme est au format préféré.

- **Table Lookup** : ce stage évalue un terme et le compare avec une forme de ce terme ayant été préalablement validée. Si le terme n'est pas dans la forme requise, alors la version standard le remplace. Table Lookup peut aussi remplacer un mot complet par son abréviation, une abréviation par un mot entier, changer un surnom en nom entier ou corriger une faute d'orthographe
- **Advanced Transformer** : ce stage scanne et divise les chaînes de données en champs multiples, en plaçant les données extraites et non extraites dans un champ existant ou un nouveau champ.

Jobs pris en charge

Le module Data Normalization de SDK qualité des Big Data prend en charge les jobs :

1. Table Lookup
   - Recherche dans la table avec l'option de normalisation
   - Recherche dans la table avec l'option d'identification
   - Recherche dans la table avec l'option de catégorisation
2. Advanced Transformer
   - Advanced Transformer avec option d'extraction des données de table
   - Advanced Transformer avec option d'extraction des expressions régulières

Table Lookup

Le stage Table Lookup standardise les termes en s'appuyant sur une forme de ce terme ayant été préalablement validée et en appliquant la version standard. Cette évaluation s'effectue en cherchant dans une table afin d'y trouver le terme à normaliser.

Advanced Transformer

Le job Advanced Transformer scanne et divise des chaînes de données en différents champs à l'aide de tables ou d'expressions régulières. Extraction d'un terme ou nombre de mots spécifique en partant de la gauche ou de la droite de ce terme. Les données extraites et non extraites peuvent être placées dans un champ existant ou un nouveau champ.

Par exemple, si vous voulez extraire les informations liées à la Suite (STE) de ce champ d'adresse et la placer dans un champ séparé.

2300 BIRCH RD STE 100

Pour accomplir ceci, vous pouvez créer un Advanced Transformer qui extraira le terme STE et tous les mots à la droite du terme STE, ce qui laissera le champ sous la forme :

2300 BIRCH RD

Module Universal Addressing

Le module Universal Addressing est un module de gestion de qualité des adresses qui peut normaliser et valider des adresses et améliorer la délivrabilité de vos courriers. Ce module peut s'assurer que vos données d'adresse adhèrent aux normes de qualités établies par les autorités postales. Une adresse qui adhère à ces normes a plus de chance d'être délivrée dans les meilleurs délais. En outre, les expéditeurs qui suivent ces normes peuvent être éligibles à des réductions significatives sur les tarifs postaux. Pour plus d'informations sur les réduction tarifaires concernant le courrier aux États-Unis, reportez-vous à USPS Domestic Mail Manual (DMM, manuel sur le courrier domestique), disponible à l'adresse : **www.usps.com**.

> **Remarque :** Pour les jobs UAM, les données de référence doivent être placées uniquement sur des nœuds de données locaux du cluster.

Jobs pris en charge

Le module Universal Addressing de SDK qualité des Big Data prend en charge les jobs :

1. Validate Address

   > **Remarque :** Ce job prend actuellement en charge uniquement les validations d'adresse aux États-Unis.

2. Validate Address Global
3. Validate Address Loqate

Validate Address

Validate Address normalise et valide les adresses en utilisant les données postales des services postaux officiels. Validate Address peut corriger les informations et mettre en forme l'adresse en utilisant le format préféré par le service postal applicable. Il ajoute également les informations postales manquantes, comme les codes postaux, les noms de ville, les noms d'état ou de province, etc.

Validate Address renvoie également des indicateurs de résultats sur des tentatives de validation, telles que la validation par Validate Address de l'adresse, du niveau de confiance dans l'adresse de retour, de la raison de l'échec si l'adresse n'a pas pu être validée et plus encore.

Pendant la correspondance et la normalisation de l'adresse, Validate Address sépare les lignes d'adresse en composants et les compare avec les contenus des bases de données du module

d’Adressage Universel. Si une correspondance existe, l'adresse d'entrée est *normalisée* en fonction des informations de la base de données. S'il n'existe pas de correspondance dans la base de données, Validate Address peut éventuellement *formater* les adresses d'entrée. Le processus de mise en forme tente de structurer les lignes d'adresse conformément aux conventions du service postal approprié.

> **Remarque :** Actuellement, Validate Address prend en charge uniquement les adresses américaines.

Rapports CASS

Vous pouvez créer et exécuter le job Validate Address en mode certifié CASS™ à l’aide du SDK Qualité des Big Data.

En outre, vous pouvez choisir de générer ces types de rapports CASS :

1. Rapport CASS 3553
2. Rapport CASS détaillé
3. Rapport de synthèse Validate Address

Traitement certifié CASS

Le traitement Certifié CASS™ génère également le USPS CASS Detailed Report, qui contient certaines informations figurant également dans le rapport 3553, mais fournit des statistiques DPV, LACS et SuiteLink beaucoup plus détaillées. Le USPS CASS Detailed Report n'est pas obligatoire pour les remises sur les tarifs postaux et il n'est pas nécessaire de le soumettre avec votre courrier.

Le rapport détaillé CASS est généré en trois parties, comme suit :

1. *Détail CASS*
2. *Détail CASS 2*
3. *Détail CASS 3*

Pour plus d’informations sur les paramètres CASS lors de l’utilisation du SDK, reportez-vous aux sections **Utilisation d’un job Validate Address MapReduce** à la page 113 et **Utilisation d’un job Validate Address Spark** à la page 115. Pour obtenir des instructions d'utilisation des rapports, reportez-vous au *Guide Dataflow Designer*.

Rapport CASS 3553

Le rapport USPS CASS 3553 doit être remis à USPS avec le courrier pour avoir droit à certaines réductions. Le rapport contient les informations sur le logiciel que vous utilisez pour le traitement CASS, les informations sur la liste de vos nom et adresse, les informations sur votre fichier de sortie, les informations sur le préposé à l'expédition et d'autres statistiques sur votre courrier. Pour des informations détaillées sur le formulaire USPS 3553, rendez-vous sur **http://www.usps.com**.

Pour obtenir des instructions d'utilisation des rapports, reportez-vous au *Guide Dataflow Designer*.

Rapport CASS détaillé

Il n'est pas nécessaire de transmettre le USPS CASS Detailed Report à USPS pour bénéficier de certaines remises. Certaines informations de ce rapport sont identiques à celles du rapport 3553, mais il fournit des statistiques DPV, LACS et SuiteLink beaucoup plus détaillées.

Pour obtenir des instructions d'utilisation des rapports, reportez-vous au *Guide Dataflow Designer*.

Rapport de synthèse Validate Address

Le rapport de synthèse Validate Address répertorie les statistiques de synthèse concernant le job, telles que le nombre total d'enregistrements traités, le nombre d'adresses validées et plus.

Pour obtenir des instructions d'utilisation des rapports, reportez-vous au *Guide Dataflow Designer*.

Validate Address Global

Validate Address Global fournit une normalisation et une validation d'adresse améliorées des adresses en dehors des États-Unis et du Canada. Validate Address Global peut également valider des adresses aux États-Unis et au Canada, néanmoins la force de ce composant réside dans la validation des adresses situées dans les autres pays. Si vous traitez un grand nombre d'adresses en dehors des États-Unis et du Canada, il est recommandé d'utiliser Validate Address Global.

Validate Address Global fait partie du module Universal Addressing.

Validate Address Global effectue plusieurs étapes pour obtenir une adresse de qualité, notamment l'analyse, la validation et la mise en forme.

Analyse syntaxique d'adresse, Formatage et Standardisation

Restructurer des données d'adresse incorrectement renseignées est une tâche complexe et difficile surtout pour les adresses internationales. Les gens introduisent de nombreuses ambiguïtés en entrant des données d'adresse dans les systèmes informatiques. Parmi les problèmes, on trouve les éléments mal placés (tels que les noms personnels ou de société dans les champs d'adresse) ou des abréviations variantes qui ne sont pas seulement spécifiques à la langue mais aussi au pays. Validate Address Global identifie les éléments d'adresse dans les lignes d'adresse et les assigne aux champs appropriés. C'est un précurseur important à la validation effective. Sans restructuration, des situations de « aucune correspondance » peuvent en résulter.

Des éléments d'adresse correctement identifiés sont également importants lorsque les adresses doivent être tronquées ou raccourcies pour correspondre aux exigences spécifiques de longueur. Avec les bonnes informations dans les bons champs, les règles de troncation spécifiques peuvent être appliquées.

- Analyse et vérifie les lignes d'adresse et identifie les éléments individuels d'adresse
- Traite plus de 30 jeux de caractères différents
- Formate les adresses selon les règles postales du pays de destination
- Normalise les éléments d'adresse (tels que changer AVENUE en AVE)

Validation d'Adresse Globale

La validation d'adresse est le traitement de la correction où les données d'adresse correctement analysées sont comparées aux bases de données de référence fournies par les organisations postales ou d'autres fournisseurs de données. Validate Address Global valide les éléments d'adresse individuelle pour vérifier l'exactitude à l'aide de la technologie sophistiquée de correspondance approximative et produit une sortie standardisée et formatée selon les normes postales et des préférences utilisateur. Le type de validation FastCompletion peut être utilisé dans les applications de saisie rapide d'adresse. Cela permet d'entrer des données tronquées dans plusieurs champs d'adresse et génère des suggestions sur la base de cette saisie.

Dans certains cas, il est impossible de valider complètement une adresse. Ici Validate Address Global a une fonctionnalité unique d'évaluation de livrabilité qui classe les adresses en fonction de leur livrabilité probable.

Compteurs de reporting

Le job Validate Address Global vous permet de surveiller les statistiques du job une fois l'exécution terminée. Les compteurs fournissent des statistiques de reporting pour tous les pays pris en charge dans lesquels un job Validate Address Global donné est exécuté.

Pour obtenir une liste des pays pris en charge, reportez-vous à la section **Prise en charge du module et des codes ISO de pays** à la page 207.

Compteurs basés sur les pays

Ces compteurs fournissent des statistiques de reporting pour les différents pays pris en charge. Chaque libellé de compteur commence par le code de pays auquel la valeur de compteur correspond.

Par exemple, ces compteurs fournissent des statistiques de reporting pour les États-Unis :

1. UNITEDSTATES_STATUS_I4_COUNT
2. UNITEDSTATES_STATUS_S_COUNT
3. UNITEDSTATES_STATUS_I3_COUNT
4. UNITEDSTATES_FAILED_COUNT
5. UNITEDSTATES_STATUS_I2_COUNT
6. UNITEDSTATES_STATUS_C_COUNT
7. UNITEDSTATES_STATUS_V_COUNT

De plus, les mêmes compteurs sont répertoriés pour tous les pays pris en charge pour lesquels le job Validate Address Global est exécuté.

Compteurs de synthèse

Les compteurs de synthèse fournissent une sommation des valeurs de chaque type de compteur donné pour tous les pays.

Par exemple, le compteur `SUMMARY_FAILED_COUNT` est la somme des valeurs du compteur `FAILED_COUNT` pour tous les pays pris en charge dans lesquels un job Validate Address Global donné est exécuté.

1. SUMMARY_STATUS_I4_COUNT
2. SUMMARY_STATUS_I2_COUNT
3. SUMMARY_END_TIME
4. SUMMARY_START_TIME
5. SUMMARY_STATUS_V_COUNT
6. SUMMARY_STATUS_C_COUNT
7. SUMMARY_CHARSET
8. SUMMARY_DEFAULT_COUNTRY
9. SUMMARY_STATUS_I3_COUNT
10. SUMMARY_STATUS_S_COUNT
11. SUMMARY_FAILED_COUNT
12. COUNTRY: liste des codes de pays séparés par des virgules pour lesquels la validation d'adresse est exécutée.
13. SUMMARY_CASING : méthode de mise en casse de la sortie. Pour obtenir des informations détaillées, reportez-vous à la section *Options* du stage *Validate Address Global* dans le *Guide Addressing*.

Validate Address Loqate

Validate Address normalise et valide les adresses en utilisant les données d'adresse postale des services postaux officiels. Validate Address Loqate peut corriger les informations et mettre l'adresse en forme au format préféré par le service postal concerné. Elle ajoute également les informations postales manquantes, comme les codes postaux, les noms de ville, les noms d'état/province, et plus encore.

Validate Address Loqate renvoie également des indicateurs de résultat sur des tentatives de validation, indiquant par exemple si Validate Address Loqate a validé l'adresse, le niveau de confiance de l'adresse renvoyée, la raison de l'échec si l'adresse n'a pas pu être validée, etc.

Lors de la mise en correspondance et de la normalisation de l'adresse, Validate Address Loqate sépare les lignes d'adresse en composants et les compare aux contenus des bases de données du module Universal Addressing. Si une correspondance existe, l'adresse d'entrée est normalisée en fonction des informations de la base de données. En l'absence de correspondance dans la base de données, ValidateAddress Loqate peut éventuellement formater les adresses d'entrée. Le processus de mise en forme tente de structurer les lignes d'adresse conformément aux conventions du service postal approprié. Validate Address Loqate fait partie du module Universal Addressing.

Compteurs de reporting

Le job Validate Address Loqate vous permet de surveiller les résultats du job. Les compteurs disponibles sont les suivants :

1. Code postal d'origine confirmé via Address Match
2. Nombre total d'enregistrements en correspondance

3. Non-correspondance de résidence
4. Nombre total d’enregistrements tentés par Address Validation
5. Nombre d'enregistrements d'entrée
6. Non-correspondance de plage de numéros
7. Nombre total d’enregistrements valides en entrée
8. Aucun code postal disponible
9. Nombre total sans correspondance enregistrée
10. Nombre total corrigés
11. Nombre total d'enregistrements sans correspondance
12. Code postal corrigé via Address Match
13. Adresse standard renvoyée correctement
14. Enregistrements d'adresse traités
15. Rue sans correspondance
16. Code postal d'origine conservé
17. Enregistrements traités par LOQATE

Module Universal Name

Pour obtenir la plus grande précision de standardisation possible, vous devez décomposer les chaînes de données en plusieurs champs. SDK qualité des Big Data fournit des fonctionnalités d'analyse avancées vous permettant d'analyser les noms de personnes, les noms d'entreprises et de nombreux autres termes et abréviations.

Jobs pris en charge

Le module Universal Name de SDK qualité des Big Data prend en charge le job :

1. Open Name Parser

Open Name Parser

L'Open Parser décompose et analyse vos données d'entrée provenant de plusieurs cultures du monde différentes en utilisant une grammaire simple mais puissante. En utilisant cette grammaire, vous pourrez définir une séquence d'expressions représentant des modèles de domaine utiles au parsing (décomposition analytique) de vos données d'entrée. Open Parser recueille également des statistiques et note les correspondances de parsing afin de vous aider à décider de l'efficacité de vos grammaires de parsing.

Reporting

Le rapport de synthèse Open Name Parser répertorie les statistiques de synthèse concernant le job, comme le nombre total d'enregistrements d'entrée et le nombre total d'enregistrements ne contenant aucune donnée de nom, ainsi que différentes statistiques d'analyse.

Résultats généraux

INPUT_RECORDS	Nombre d'enregistrements dans l'entrée.
NO_NAME_DATA_RECORDS	Nombre d'enregistrements dans l'entrée qui ne contiennent pas de données de nom à analyser.
NAMES_PARSED_OUT	Nombre de noms contenus dans l'entrée qui ont été analysés.
LOWEST_NAME_PARSING_SCORE	Score d'analyse le plus faible attribué à un nom dans l'entrée.
HIGHEST_NAME_PARSING_SCORE	Score d'analyse le plus élevé attribué à un nom dans l'entrée.
AVERAGE_NAME_PARSING_SCORE	Score d'analyse moyen attribué à tous les noms anamysés dans l'entrée.

Résultats d'analyse de noms de personnes

PERSONAL_NAME_RECORDS	Nombre de noms personnel dans l'entrée.
CONJOINED_NAMES_PARSED	Nombre de noms analysés provenant d'enregistrements qui contenaient des noms liés. Par exemple, si votre fichier d'entrée compte cinq enregistrements avec deux noms liés et sept enregistrements avec trois noms liés, la valeur de ce compteur est 31, conformément à l'équation suivante : (5 x 2) + (7 x 3).
TWO_CONJOINED_NAMES_RECORDS	Nombre d'enregistrements en entrée contenant deux noms liés.
THREE_CONJOINED_NAMES_RECORDS	Nombre d'enregistrements en entrée contenant trois noms liés.
TITLE_OF_RESPECT_NAMES	Nombre de noms analysés contenant une civilité.
MATURITY_SUFFIX_NAMES	Nombre de noms analysés contenant un suffixe générationnel.
GENERAL_SUFFIX_NAMES	Nombre de noms analysés contenant un suffixe général.
ACCOUNT_DESCRIPTION_PERSONAL_NAMES	Nombre de noms analysés contenant une description de compte.

TOTAL_REVERSE_ORDER_NAMES	Nombre de noms analysés figurant dans l'ordre inverse et produisant une valeur « True » dans le champ de sortie `IsReverseOrder` .

Résultats d'analyse de noms d'entreprises

BUSINESS_NAME_RECORDS	Nombre d'enregistrements en entrée contenant des noms commerciaux+.
FIRM_SUFFIX_NAMES	Nombre de noms analysés contenant un suffixe d'entreprise.
ACCOUNT_DESCRIPTION_BUSINESS_NAMES	Nombre d'enregistrements en entrée contenant une description de compte.
TOTAL_DBA_RECORDS	Nombre d'enregistrements en entrée contenant des conjonctions Doing Business As (DBA), produisant une valeur « True » dans les deux champs de sortie `isPersonal` and `isFirm`.
TOTAL_PARSED	Nombre total de noms analysés.
TOTAL_NAME_PARSING_SCORE	Score d'analyse total de tous les noms.

4 - L'API Java

In this section

Introduction

Une classe *Java* est un bleu ou un prototype définissant les variables et les méthodes communes à tous les objets d'un certain type. Elle définit l'implémentation d'un type particulier d'instance.

Un *objet* Java est une instance d'une classe Java. Il s'agit d'une instance en temps réel de classes Java, créée en utilisant une machine virtuelle Java. Une instance de classe, gérée à l'aide d'une variable, encapsule les informations de la classe en temps réel.

Les *méthodes* d'une classe définissent les différentes fonctions qu'une classe ou son objet doit effectuer. Les méthodes sont similaires aux fonctions ou procédures des langages procéduraux tel que le C.

Les *paramètres* sont utilisés pour transmettre les informations qu'un objet requiert pour effectuer une certaine tâche.

Les objets logiciels Java interagissent et communiquent l'un avec l'autre à l'aide de *messages*.

Pour plus d'informations sur la technologie Java, reportez-vous à **www.oracle.com/java**.

Composants de l'API Java du SDK

Les composants clés pour utiliser un job SDK qualité des Big Data via l'API Java sont les suivants :

Fichiers JAR

1. Fichiers JAR Hadoop.
2. Fichiers JAR du module auquel appartient le job SDK qualité des Big Data souhaité, comme indiqué dans le tableau :

Module	Job	Fichier JAR
Module Advanced Matching	Tous les jobs AMM	*amm.core-12.0.jar*
Module Data Normalization	Tous les jobs DNM	*dnm.core-12.0.jar*
Module Universal Addressing	Validate Address	*uam-universaladdress.core-12.0.jar*
Module Universal Addressing	Validate Address Global	*uam-global.core-12.0.jar*
Module Universal Addressing	Validate Address Loqate	*uam-loqate.core-12.0.jar*

Module	Job	Fichier JAR
Module Universal Name	Tous les jobs UNM	*unm.core-12.0.jar*

Fichiers de configuration	Fichiers au format XML contenant tous les paramètres et toutes les valeurs nécessaires pour exécuter un job, y compris les règles de correspondance, les détails du fichier d'entrée, les détails du fichier de sortie, les détails de configuration MapReduce ou Spark, etc. Des exemples de fichiers de configuration XML se trouvent sous `<Big Data Quality bundle>\samples\configuration`.
Application Java client	Application Java pour utiliser l'API pour créer et exécuter le job SDK qualité des Big Data souhaité fourni par son API Java.
Plate-forme Hadoop	Le job créé accède à la plate-forme Hadoop configurée pour accéder aux données d'entrée et placer les données de sortie dans un fichier.

Utilisation du SDK

Le SDK peut être utilisé pour exécuter des jobs SDK qualité des Big Data via l'une de ces deux approches :

1. Sur une console, exécutez directement les fichiers JAR spécifiques au module et transmettez les différents fichiers de propriétés de configuration au format XML sous forme d'arguments des commandes.

   Pour les jobs MapReduce, exécutez la commande `hadoop`, tandis que pour les jobs Spark, exécutez la commande `submit-spark`.

   Pour connaître les étapes, reportez-vous à la section **Utilisation de fichiers de propriétés de configuration** à la page 35.

2. Créez votre propre projet client Java en important le fichier JAR du module SDK qualité des Big Data pertinent, indiquez toutes configurations requises du job de votre choix au sein de votre projet client et exécutez-le.

   Pour connaître les étapes, reportez-vous à la section **Création d'une application Java** à la page 37.

Utilisation de fichiers de propriétés de configuration

Assurez-vous que SDK qualité des Big Data est installé sur votre ordinateur.

Vous pouvez exécuter un job SDK qualité des Big Data à l'aide des fichiers JAR spécifiques au module et des fichiers de configuration au format XML.

Les exemples de propriétés de configuration sont livrés avec le SDK Qualité des Big Data et placés sous `<Big Data Quality bundle>\samples\configuration`.

> **Remarque :** Pour obtenir une liste des fichiers JAR propres au module, reportez-vous à la section **Composants de l'API Java du SDK** à la page 34.

1. Pour un système Linux, ouvrez une invite de commande.

   Pour les systèmes Windows et Unix, ouvrez un client SSH tel que Putty.

2. Pour un job *MapReduce*, utilisez la commande `hadoop`.

   Suivant le job que vous souhaitez exécuter :

   1. Transmettez le nom du fichier JAR de ce module.
   2. Transmettez le nom de la classe de pilote `RunMRSampleJob`.
   3. Transmettez les différents fichiers de configuration sous forme de liste d'arguments. Chaque clé d'argument accepte le chemin d'accès à un seul fichier de propriétés de configuration, où chaque fichier contient plusieurs propriétés de configuration.

   La syntaxe de la commande est la suivante :

   ```
   hadoop jar <Name of module JAR file> RunMRSampleJob [-config <Path to
   configuration file>] [-debug] [-input <Path to input configuration
   file>] [-conf <Path to MapReduce configuration file>] [-output <Path
   of output directory>]
   ```

   Par exemple, pour un job MapReduce MatchKeyGenerator :

   ```
   hadoop jar amm.core.12.0.jar RunMRSampleJob -config
   /home/hadoop/matchkey/mkgConfig.xml -input
   /home/hadoop/matchkey/inputFileConfig.xml -conf
   /home/hadoop/matchkey/mapReduceConfig.xml -output
   /home/hadoop/matchkey/outputFileConfig.xml
   ```

3. Pour un job *Spark*, utilisez la commande `spark-submit`.

   Suivant le job que vous souhaitez exécuter :

   1. Transmettez le nom du fichier JAR de ce module.
   2. Transmettez le nom de la classe de pilote `RunSparkSampleJob`.
   3. Transmettez les différents fichiers de configuration sous forme de liste d'arguments. Chaque clé d'argument accepte le chemin d'accès à un seul fichier de propriétés de configuration, où chaque fichier contient plusieurs propriétés de configuration.

   La syntaxe de la commande est la suivante :

   ```
   spark-submit --class RunSparkSampleJob <Name of module JAR file> [-config
   <Path to configuration file>] [-debug] [-input <Path to input
   configuration file>] [-conf <Path to Spark configuration file>] [-output
   <Path of output directory>]
   ```

   Par exemple, pour un job Spark MatchKeyGenerator :

```
spark-submit --class RunSparkSampleJob amm.core.12.0.jar -config
/home/hadoop/spark/matchkey/matchKeyGeneratorConfig.xml -input
/home/hadoop/spark/matchkey/inputFileConfig.xml -output
/home/hadoop/spark/matchkey/outputFileConfig.xml
```

Remarque : Pour afficher une liste de clés d'argument prises en charge pour les commandes `hadoop` ou`spark-submit`, exécutez les commandes :

```
hadoop --help
```

ou

```
spark-submit --help
```

Création d'une application Java

Assurez-vous que SDK qualité des Big Data est installé sur votre ordinateur.

Pour utiliser le SDK :

1. Créez un projet Java pour utiliser le SDK nécessaire à l'aide de l'une de ces méthodes :
   a) Créez un projet Java spécifique pour exécuter l'opération de qualité des données requise.
      À l'aide de cette méthode, vous devez créer des projets Java distincts pour chaque job de qualité des données que vous voulez lancer.
   b) Créez un projet Java courant pour exécuter l'une des opérations Data Quality souhaitées en utilisant les arguments de l'exécution correspondante.
      À l'aide de cette méthode, vous ne créez qu'un seul projet Java qui accepte des arguments d'exécution correspondant à l'opération de qualité des données de votre choix.
2. Importez le fichier JAR spécifique au module SDK qualité des Big Data dans votre projet pour utiliser le SDK. Pour obtenir une liste des fichiers JAR propres au module, reportez-vous à la section **Composants de l'API Java du SDK** à la page 34.
3. Importez les fichiers JAR Hadoop requis dans votre projet.
4. Créez votre application pour exécuter les jobs de qualité des données de votre choix, avec les configurations appropriées.
5. Générez votre projet, à l'aide de n'importe quel outil intégré comme Maven ou Ant.
   Un fichier JAR de votre projet est créé en conséquence.

   Par exemple,`MatchKeyGeneratorClient-with-dependencies.jar` est créé.
6. Placez le fichier JAR de votre projet sur la plate-forme Hadoop.
7. Sur la plate-forme Hadoop, dans une invite de commande, remplacez le répertoire vers le chemin d'accès où vous avez placé votre fichier JAR.

8. Exécutez le fichier JAR de votre projet à l'aide de la commande :

```
hadoop jar <name of the JAR of your client project> <fully qualified
name of the main class>
```

Par exemple :

```
hadoop jar MatchKeyGeneratorClient-with-dependencies.jar
com.company.bdq.amm.mr.MatchKeyGeneratorJob
```

Le job de votre choix est créé et exécuté sur la plate-forme Hadoop.

Votre application Java accède aux données d'entrée à partir du chemin d'accès spécifié sur la plate-forme Hadoop et crée et exécute le job sur la plate-forme Hadoop. La sortie du job est exportée dans un fichier sur le chemin d'accès de sortie spécifié sur la plate-forme Hadoop.

Entités API communes

ConjoinedRule

Objectif
Type de règle de consolidation utilisé lorsque plusieurs règles doivent être jointes à l'aide des opérateurs `AND` et `OR`. Une règle conjointe peut inclure des règles simples comme composants. Reportez-vous à la section **SimpleRule** à la page 41.

Cette classe permet de définir des règles pour les jobs du module Advanced Matching et du module Data Normalization.

ConsolidationCondition

Objectif
Définir les règles de consolidation et l'action correspondante pour les jobs du module Advanced Matching et du module Data Normalization.

ConsolidationRule

Objectif
Indiquer la règle de consolidation en fonction de laquelle il doit être déterminé si une action sur un enregistrement est requise ou non.

Cette classe permet de définir des règles de consolidation pour les jobs du module Advanced Matching et du module Data Normalization.

ConsolidationAction

Objectif
Pour spécifier le champ qui doit être copié dans d'autres enregistrements d'un groupe pour une condition de consolidation donnée.

Cette classe permet de définir des actions de consolidation pour les jobs du module Advanced Matching et du module Data Normalization.

FilePath

Objectif
Spécifier les détails d'un fichier texte d'entrée ou de sortie pour exécuter un job.

JobConfig< Type de processus T extends >

Objectif
Une interface pour spécifier les configurations Hadoop d'un job.

MRJobConfig

Objectif
Spécifier des configurations Hadoop pour tout job MapReduce.

SparkJobConfig

Objectif
Spécifier des configurations Hadoop pour tout job Spark.

JobDetail< Type de processus T extends >

Objectif
Stocke les informations de base nécessaires pour la création d'un job.

JobFactory

Objectif
Interface de base permettant de spécifier les instances et les détails des jobs à créer.

JobPath

Objectif
La classe parente pour spécifier les détails de la source d'entrée et de la destination de sortie d'un job.

OrcFilePath

Spécifier les chemins d'accès d'entrée ou de sortie aux fichiers de format ORC pour exécuter un job.

ProcessType

Objectif
Interface de balisage parent de tous les types de processus pris en charge, comme MapReduce et Spark.

MRProcessType

Objectif
Spécifier le type de processus pour les jobs MapReduce.

SparkProcessType

Objectif
Spécifier le type de processus Spark pour les jobs.

ReferenceDataPath

Objectif
Spécifier le chemin d'accès de données de référence d'un job.

ReportManager

Objectif
Une interface pour extraire les statistiques de reporting d'un job.

SimpleRule

Objectif
Type de règle de consolidation. Une règle simple peut être utilisée seule et comme composant d’une règle conjointe. Reportez-vous à la section **ConjoinedRule** à la page 38.

Exceptions

JobException

Objectif
Gère les exceptions spécifiques, en affichant des messages appropriés.

Jobs module Advanced Matching

Module commun API

AdvanceMatchDetail < Type de processus T extends >

Objectif
Pour spécifier les détails d'un job du module Advanced Matching.

AdvanceMatchFactory

Objectif
Une classe usine singleton pour créer des instances de jobs Module Advanced Matching.

GroupbyOption< Type de processus T extends >

Objectif
Spécifier la colonne sur laquelle le groupement doit être effectué pour un job Advanced Matching.

GroupbyMROption

Objectif
Spécifier la colonne sur laquelle le groupement doit être effectué pour un job MapReduce Advanced Matching.

GroupbySparkOption

Objectif
Spécifier la colonne sur laquelle le regroupement doit être effectué pour un job Advanced Matching Spark.

MatchKeySettings

Objectif
Conserve une `List` de clés de correspondance pour un job Match Key Generator.

MatchRule

Objectif
Permet de créer des règles de correspondance pour les jobs Advanced Matching.

Cela s'effectue par la définition d'une hiérarchie de nœuds parents et enfants. Chaque nœud correspond à l'un des champs d'entrée à mettre en correspondance.

ChildMatchRule

Objectif
Indiquer un nœud enfant d'une règle de correspondance, qui mappe vers un champ et certains algorithmes et autres propriétés.

ParentMatchRule

Objectif
Pour indiquer un nœud parent d'une règle de correspondance, qui est un regroupement logique d'autres nœuds parents et de nœuds enfants.

Scénarios spéciaux

Enregistrements avec une colonne Group-By vide
Tous les enregistrements avec une valeur Group-By vide sont marqués comme des enregistrements non conformes et placés dans des fichiers séparés dans le dossier HDFS de sortie.

Ces fichiers non conformes sont nommés comme suit :

Enregistrements non conformes dans des fichiers candidats	Les enregistrements de fichiers candidats avec une colonne Group-By vide sont rejetés comme des enregistrements non conformes et insérés dans des fichiers suivant la convention de nommage de fichiers `malformedRecordsCandidate-m-<5 digit numeral>`. Par exemple, `malformedRecordsCandidate-m-00000`,`malformedRecordsCandidate-m-00001`. Cela s'applique aux jobs Interflow Match.
Enregistrements non conformes dans des fichiers suspects	Les enregistrements de fichiers suspects avec une colonne Group-By vide sont rejetés comme des enregistrements non conformes et insérés dans des fichiers suivant la convention de nommage de fichiers `malformedRecordsSuspect-m-<5 digit numeral>`. Par exemple, `malformedRecordsSuspect-m-00000`,`malformedRecordsSuspect-m-00001`. Cela s'applique aux jobs Interflow Match.

Enregistrements non conformes dans des fichiers d'entrée	Les enregistrements de fichiers d'entrée avec une colonne Group-By vide sont rejetés comme des enregistrements non conformes et insérés dans des fichiers suivant la convention de nommage de fichiers `malformedRecords-m-<5 digit numeral>`. Par exemple,`malformedRecords-m-00000`,`malformedRecords-m-00001`. Cela s'applique aux jobs Intraflow Match, Transactional Match, Best of Breed, Duplicate Synchronization et Filter.

Compteurs d'enregistrements non conformes

Le nombre d'enregistrements non conformes d'un job exécuté est stocké dans les compteurs :

- MALFORMED_CANDIDATE_RECORDS
- MALFORMED_SUSPECT_RECORDS
- MALFORMED_RECORDS

> **Remarque :** Les valeurs de ces compteurs sont accessibles via l'appel de la méthode `getCounters()` de l'instance `AdvanceMatchFactory`.

Match Key Generator

Présentation

Le job Match Key Generator vous permet de générer des clés de correspondance.

> **Remarque :** Pour générer une clé de correspondance pour les données, vous devez exécuter le job Match Key Generator une fois avant d'exécuter les autres jobs.

Entités API

MatchKeyGeneratorDetail

Objectif
Spécifier les détails d'un job Match Key Generator.

Paramètres d'entrée

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.
Match Key Settings	Combinaison des colonnes et des algorithmes à appliquer pour générer la clé de correspondance, requise pour effectuer la correspondance. **Remarque :** Vous devez spécifier au moins une clé de correspondance. Vous pouvez spécifier plusieurs clés de correspondance, si nécessaire.
Nom du job	Nom du job.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Match Key Generator :

Colonne	Description	Valeur de sortie
MatchKey	La clé générée pour identifier les enregistrements.	Clé générée en fonction des colonnes et des algorithmes sélectionnés pour générer la clé de correspondance. **Remarque :** Le nombre de colonnes de Match Key nommées par l'utilisateur générées dans la sortie dépend des paramètres du job.

Utilisation d'un job Match Key Generator MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Match Key Generator en créant une instance de `MatchKeyGeneratorDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez les paramètres de clé de correspondance pour effectuer la mise en correspondance en créant et en configurant une instance de `MatchKeySettings`. Pour plus d'informations, reportez-vous à l'échantillon de code correspondant.
   b) Créez une instance de `MatchKeyGeneratorDetail` en transmettant une instance de type `JobConfig` et l'instance `MatchKeySettings` créée comme arguments à son constructeur.
   Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.
   c) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `MatchKeyGeneratorDetail`.
   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.
   d) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `MatchKeyGeneratorDetail`.
   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.
   e) Définissez le nom du job à l'aide du champ `jobName` de l'instance `MatchKeyGeneratorDetail`.
3. Pour créer un job MapReduce, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `MatchKeyGeneratorDetail` comme argument.
   La méthode `createJob()` crée le job et renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.

Utilisation d'un job Match Key Generator Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Match Key Generator en créant une instance de `MatchKeyGeneratorDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez les paramètres de clé de correspondance pour effectuer la mise en correspondance en créant et en configurant une instance de `MatchKeySettings`. Pour plus d'informations, reportez-vous à l'échantillon de code correspondant.
   b) Créez une instance de `MatchKeyGeneratorDetail` en transmettant une instance de type `JobConfig` et l'instance `MatchKeySettings` créée comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.
   c) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `MatchKeyGeneratorDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.
   d) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `MatchKeyGeneratorDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.
   e) Définissez le nom du job à l'aide du champ `jobName` de l'instance `MatchKeyGeneratorDetail`.
3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `MatchKeyGeneratorDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

Interflow Match

Présentation

Le job Interflow vous permet de générer une clé de correspondance, de regrouper des enregistrements à l'aide de la clé de correspondance et de réaliser une correspondance croisée (intermatching) sur des enregistrements provenant de différentes sources de données.

Entités API

InterMatchDetail

Objectif
Spécifier les détails d'un job Interflow Match.

InterMatchComparisonOption

Objectif
Pour spécifier des options de comparaison lors de la définition d'un job Interflow Match, si l'enregistrement suspect doit être comparé à tous les enregistrements candidats, ou à un enregistrement candidat sélectionné.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, pour créer une option Group-By, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés.
Match Rule	Définissez autant de règles parents et enfants que nécessaire pour créer un objet`MatchRule`. Pour plus d'informations, reportez-vous à la section **MatchRule** à la page 43.

Paramètre	Description
Candidate File	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte candidat sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier candidat. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier candidat. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier candidat. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier suspect. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. **Important :** Les fichiers suspects et candidats doivent se présenter au même format. Soit les deux doivent être des fichiers texte, soit ils doivent être des fichiers de format ORC. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Suspect File	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte suspect sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier suspect. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier suspect. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier suspect. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier suspect. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.
Match Key Settings	Combinaison des colonnes et des algorithmes à appliquer pour générer la clé de correspondance, requise pour effectuer la correspondance. **Remarque :** Spécifiez une seule clé de correspondance. **Avertissement :** Définissez les paramètres de clé de correspondance uniquement si vous souhaitez générer une clé de correspondance avant d'effectuer la mise en correspondance.
Nom du job	Nom du job.
Express Match Column	Nom de la colonne à utiliser pour la mise en correspondance express d'enregistrements.
Setting Collection Number Zero to Unique Records	Définissez cette valeur sur `true` pour définir le nombre de collections d'enregistrements uniques sur 0 (zéro).

Paramètre	Description
Comparison Option	Vous permet de sélectionner l'une des deux options : • Compare the Suspect record to all Candidate records : Spécifiez si des enregistrements uniques doivent être renvoyés ou non dans la sortie. • Compare the Suspect record to the selected Candidate record only : Spécifiez le nombre maximal d'enregistrements doublons à rechercher et renvoyer.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Interflow Match :

Colonne	Description	Valeur de sortie
Collection Number	Identifie une collection de doublons.	Les valeurs possibles sont `0-0-1`, `0-0-2`, etc.
Express Match Identified	Indique si un rapprochement a été obtenu à l'aide de la clé de correspondance express.	1. Pour un enregistrement candidat dupliqué correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est `Y`. 2. Pour un enregistrement candidat dupliqué correspondant obtenu sans utiliser de clé de correspondance express, la valeur de sortie est vide. 3. Pour un enregistrement candidat unique correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est `N`. 4. Pour un enregistrement suspect correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est vide.
Interflow Source Type	Indique si l'enregistrement d'entrée est un enregistrement suspect ou un enregistrement candidat.	Les valeurs possibles sont `S` pour un enregistrement suspect et `C` pour un enregistrement candidat.

Colonne	Description	Valeur de sortie
Match Record Type	Identifie le type d'enregistrement de correspondance dans une collection.	Les valeurs possibles sont `S` (enregistrement suspect), `D` (enregistrement doublon) et `U` (enregistrement unique).
Match Score	Identifie le score global entre deux enregistrements.	Plage de valeurs possibles de 0 (zéro) à 100 pour les enregistrements doublons et uniques, où 0 indique une faible correspondance et 100 une correspondance de très bonne qualité. **Remarque :** Pour les enregistrements suspects, cette valeur est 0.

Utilisation d'un job Interflow Match MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Interflow Match en créant une instance de `InterMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.
   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.
   c) Créez une instance de `InterMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.
   d) Définissez les détails du fichier de candidats à l'aide du champ `candidateFilePath` de l'instance `InterMatchDetail`.

      Pour un fichier candidat texte, créez une instance de `FilePath` avec les détails pertinents du fichier candidat en appelant le constructeur approprié. Pour un fichier candidat ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier candidat ORC comme argument.
   e) Définissez les détails du fichier de suspects à l'aide du champ `suspectFilePath` de l'instance `InterMatchDetail`.

      Pour un fichier suspect texte, créez une instance de `FilePath` avec les détails pertinents du fichier suspect en appelant le constructeur approprié. Pour un fichier suspect ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier suspect ORC comme argument.

Important : Les fichiers suspects et candidats doivent se présenter au même format. Soit les deux doivent être des fichiers texte, soit ils doivent être des fichiers de format ORC.

f) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `InterMatchDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

g) Définissez le nom du job à l'aide du champ `jobName` de l'instance `InterMatchDetail`.

h) Définissez la colonne Express Match à l'aide du champ `expressMatchColumn` de l'instance `InterMatchDetail`, si nécessaire.

i) Définissez l'indicateur `collectionNumberZerotoUniqueRecords` de l'instance `InterMatchDetail` sur `true` pour affecter le numéro de collection 0 (zéro) à un enregistrement unique. La valeur par défaut est `true`.

Si vous ne souhaitez pas affecter le numéro de collection zéro à des enregistrements uniques, définissez cet indicateur sur `false`.

j) Définissez l'option de comparaison à l'aide du champ `comparisonOption` de l'instance `InterMatchDetail`. Dans ce champ, définissez la valeur requise en utilisant la classe **InterMatchComparisonOption** à la page 49 pour sélectionner l'une des deux options :

- **Compare the Suspect record to all Candidate records** : Spécifiez si des enregistrements uniques doivent être renvoyés ou non dans la sortie.
- **Compare the Suspect record to the selected Candidate record only** : Spécifiez le nombre maximal d'enregistrements doublons à rechercher et renvoyer.

k) Définissez l'indicateur `compressOutput` de l'instance `InterMatchDetail` sur `true` pour compresser la sortie du job.

l) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Interflow Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance Interflow. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `InterMatchDetail`.

Remarque : Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer un job MapReduce, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `InterMatchDetail` comme argument.

La méthode `createJob()` crée le job et renvoie une `List` d'instances de `ControlledJob`.

4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Interflow Match Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Interflow Match en créant une instance de `InterMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.
   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.
   c) Créez une instance de `InterMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.
   d) Définissez les détails du fichier de candidats à l'aide du champ `candidateFilePath` de l'instance `InterMatchDetail`.

      Pour un fichier candidat texte, créez une instance de `FilePath` avec les détails pertinents du fichier candidat en appelant le constructeur approprié. Pour un fichier candidat ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier candidat ORC comme argument.
   e) Définissez les détails du fichier de suspects à l'aide du champ `suspectFilePath` de l'instance `InterMatchDetail`.

      Pour un fichier suspect texte, créez une instance de `FilePath` avec les détails pertinents du fichier suspect en appelant le constructeur approprié. Pour un fichier suspect ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier suspect ORC comme argument.

      Important : Les fichiers suspects et candidats doivent se présenter au même format. Soit les deux doivent être des fichiers texte, soit ils doivent être des fichiers de format ORC.
   f) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `InterMatchDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.
   g) Définissez le nom du job à l'aide du champ `jobName` de l'instance `InterMatchDetail`.

h) Définissez la colonne Express Match à l'aide du champ `expressMatchColumn` de l'instance `InterMatchDetail`, si nécessaire.

i) Définissez l'indicateur `collectionNumberZerotoUniqueRecords` de l'instance `InterMatchDetail` sur `true` pour affecter le numéro de collection 0 (zéro) à un enregistrement unique. La valeur par défaut est `true`.

Si vous ne souhaitez pas affecter le numéro de collection zéro à des enregistrements uniques, définissez cet indicateur sur `false`.

j) Définissez l'option de comparaison à l'aide du champ `comparisonOption` de l'instance `InterMatchDetail`. Dans ce champ, définissez la valeur requise en utilisant la classe **InterMatchComparisonOption** à la page 49 pour sélectionner l'une des deux options :

- **Compare the Suspect record to all Candidate records** : Spécifiez si des enregistrements uniques doivent être renvoyés ou non dans la sortie.
- **Compare the Suspect record to the selected Candidate record only** : Spécifiez le nombre maximal d'enregistrements doublons à rechercher et renvoyer.

k) Définissez l'indicateur `compressOutput` de l'instance `InterMatchDetail` sur `true` pour compresser la sortie du job.

l) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Interflow Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance Interflow. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `InterMatchDetail`.

> **Remarque :** Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `InterMatchDetail` comme argument.
   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.
4. Affichez les compteurs pour voir les statistiques de reporting du job.

Intraflow Match

Résumé

Le job Intraflow vous permet de générer une clé de correspondance, de regrouper des enregistrements à l'aide de la clé de correspondance et de réaliser une correspondance interne (intramatching) sur les enregistrements de la même source de données.

Entités API

IntraMatchDetail

Objectif

Spécifier les détails d'un job Intraflow Match.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, pour créer une option Group-By, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés.
Match Rule	Définissez autant de règles parents et enfants que nécessaire pour créer un objet`MatchRule`. Pour plus d'informations, reportez-vous à la section **MatchRule** à la page 43.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.
Nom du job	Nom du job.
Express Match Column	Nom de la colonne à utiliser pour la mise en correspondance express d'enregistrements.
Setting Collection Number Zero to Unique Records	Définissez cette valeur sur `true` pour définir le nombre de collections d'enregistrements uniques sur 0 (zéro).
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Paramètre	Description
Match Key Settings	Combinaison des colonnes et des algorithmes à appliquer pour générer la clé de correspondance, requise pour effectuer la correspondance. **Remarque :** Spécifiez une seule clé de correspondance. **Avertissement :** Définissez les paramètres de clé de correspondance uniquement si vous souhaitez générer une clé de correspondance avant d'effectuer la mise en correspondance.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Intraflow Match :

Colonne	Description	Valeur de sortie
Collection Number	Identifie une collection de doublons.	Les valeurs possibles sont `0-0-1`, `0-0-2`, etc.
Express Match Identified	Indique si un rapprochement a été obtenu à l'aide de la clé de correspondance express.	1. Pour un enregistrement candidat dupliqué correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est `Y`. 2. Pour un enregistrement candidat dupliqué correspondant obtenu sans utiliser de clé de correspondance express, la valeur de sortie est vide. 3. Pour un enregistrement candidat unique correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est vide. 4. Pour un enregistrement suspect correspondant obtenu à l'aide d'une clé de correspondance express, la valeur de sortie est vide.
Match Record Type	Identifie le type d'enregistrement de correspondance dans une collection.	Les valeurs possibles sont `S` (enregistrement suspect), `D` (enregistrement doublon) et `U` (enregistrement unique).

Colonne	Description	Valeur de sortie
Match Score	Identifie le score global entre deux enregistrements.	Plage de valeurs possibles de 0 (zéro) à 100 pour les enregistrements doublons et uniques, où 0 indique une faible correspondance et 100 une correspondance de très bonne qualité. **Remarque :** Pour les enregistrements suspects, cette valeur est 0.

Utilisation d'un job Intraflow Match MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Intraflow Match en créant une instance de `IntraMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.
   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.
   c) Créez une instance de `IntraMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.
   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `IntraMatchDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.
   e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `IntraMatchDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.
   f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `IntraMatchDetail`.
   g) Définissez la colonne Express Match à l'aide du champ `expressMatchColumn` de l'instance `IntraMatchDetail`, si nécessaire.

h) Définissez l'indicateur `collectionNumberZerotoUniqueRecords` de l'instance `IntraMatchDetail` sur `true` pour affecter le numéro de collection 0 (zéro) à un enregistrement unique. La valeur par défaut est `true`.

Si vous ne souhaitez pas affecter le numéro de collection zéro à des enregistrements uniques, définissez cet indicateur sur `false`.

i) Définissez l'indicateur `compressOutput` de l'instance `IntraMatchDetail` sur `true` pour compresser la sortie du job.

j) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Intraflow Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance Intraflow. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `IntraMatchDetail`.

Remarque : Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer un job MapReduce, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `IntraMatchDetail` comme argument.

   La méthode `createJob()` crée le job et renvoie une `List` d'instances de `ControlledJob`.

4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Intraflow Match Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Intraflow Match en créant une instance de `IntraMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.

   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

   Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.

   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.

   c) Créez une instance de `IntraMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `IntraMatchDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `IntraMatchDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `IntraMatchDetail`.

g) Définissez la colonne Express Match à l'aide du champ `expressMatchColumn` de l'instance `IntraMatchDetail`, si nécessaire.

h) Définissez l'indicateur `collectionNumberZerotoUniqueRecords` de l'instance `IntraMatchDetail` sur `true` pour affecter le numéro de collection 0 (zéro) à un enregistrement unique. La valeur par défaut est `true`.

Si vous ne souhaitez pas affecter le numéro de collection zéro à des enregistrements uniques, définissez cet indicateur sur `false`.

i) Définissez l'indicateur `compressOutput` de l'instance `IntraMatchDetail` sur `true` pour compresser la sortie du job.

j) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Intraflow Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance Intraflow. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `IntraMatchDetail`.

> **Remarque :** Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `IntraMatchDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Transactional Match

Présentation

Le job Transactional Match vous permet de mettre en correspondance des enregistrements suspects avec des enregistrements candidats d'un groupe d'enregistrements, pour identifier les doublons.

Entités API

TransactionalMatchDetail

Objectif

Spécifier les détails d'un job Transactional Match.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, pour créer une option Group-By, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés.
Match Rule	Définissez autant de règles parents et enfants que nécessaire pour créer un objet`MatchRule`. Pour plus d'informations, reportez-vous à la section **MatchRule** à la page 43.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.
Return Unique Candidates	Indicateur permettant de spécifier si les candidats uniques doivent être renvoyés dans le cadre de la sortie.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.
Match Key Settings	Combinaison des colonnes et des algorithmes à appliquer pour générer la clé de correspondance, requise pour effectuer la correspondance. **Remarque :** Spécifiez une seule clé de correspondance. **Avertissement :** Définissez les paramètres de clé de correspondance uniquement si vous souhaitez générer une clé de correspondance avant d'effectuer la mise en correspondance.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Transactional Match :

Paramètre	Description	Valeur de sortie
Match Record Type	Identifie le type d'enregistrement de correspondance dans une collection.	Les valeurs possibles sont `S` (enregistrement suspect), `D` (enregistrement doublon) et `U` (enregistrement unique).
Match Score	Identifie le score global entre deux enregistrements.	Plage de valeurs possibles de 0 (zéro) à 100 pour les enregistrements doublons et uniques, où 0 indique une faible correspondance et 100 une correspondance de très bonne qualité. **Remarque :** Pour les enregistrements suspects, cette valeur est 0.
Has Duplicates	Indique si les enregistrements suspects possèdent des doublons ou non.	Pour les enregistrements suspects, les valeurs de sortie possibles sont les suivantes : • `Y` (si des doublons sont présents) OU • `N` (si les doublons sont absents) Pour les enregistrements en double, la valeur de sortie est `D`. Pour les enregistrements uniques, la valeur de sortie est `U`.

Utilisation d'un job Transactional Match MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Transactional Match en créant une instance de `TransactionalMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.
   Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.
   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.

c) Créez une instance de `TransactionalMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `TransactionalMatchDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `TransactionalMatchDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `TransactionalMatchDetail`.

g) Définissez l'indicateur `returnUniqueCandidates` de l'instance `TransactionalMatchDetail` sur `true` pour renvoyer les enregistrements candidats uniques en sortie. La valeur par défaut est `true`.

h) Définissez l'indicateur `compressOutput` de l'instance `TransactionalMatchDetail` sur `true` pour compresser la sortie du job.

i) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Transactional Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance transactionnelle. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `TransactionalMatchDetail`.

> **Remarque :** Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer un job MapReduce, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `TransactionalMatchDetail` comme argument.

   La méthode `createJob()` crée le job et renvoie une `List` d'instances de `ControlledJob`.

4. Exécutez le job créé à l'aide d'une instance de `JobControl`.

5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Transactional Match Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Transactional Match en créant une instance de `TransactionalMatchDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.
   b) Générez les règles de correspondance du job en créant une instance de `MatchRule`.
   c) Créez une instance de `TransactionalMatchDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `MatchRule` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.
   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `TransactionalMatchDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.
   e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `TransactionalMatchDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.
   f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `TransactionalMatchDetail`.
   g) Définissez l'indicateur `returnUniqueCandidates` de l'instance `TransactionalMatchDetail` sur `true` pour renvoyer les enregistrements candidats uniques en sortie. La valeur par défaut est `true`.
   h) Définissez l'indicateur `compressOutput` de l'instance `TransactionalMatchDetail` sur `true` pour compresser la sortie du job.
   i) Si les données d'entrée n'ont pas de clés de correspondance, vous devez spécifier les paramètres de clé de correspondance pour exécuter tout d'abord le job Match Key Generator pour générer des clés de correspondance, avant de pouvoir exécuter le job Transactional Match.

Pour générer les clés de correspondance pour les données d'entrée, spécifiez les paramètres de clé de correspondance en créant et en configurant une instance de`MatchKeySettings` pour générer une clé de correspondance avant d'effectuer la correspondance transactionnelle. Définissez cette instance à l'aide du champ `matchKeySettings` de l'instance `TransactionalMatchDetail`.

Remarque : Pour savoir comment définir les paramètres de clé de correspondance, consultez les exemples de code.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `TransactionalMatchDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.
4. Affichez les compteurs pour voir les statistiques de reporting du job.

Best of Breed

Présentation

Le job Best of Breed consolide les enregistrements doublons en sélectionnant les meilleures données parmi un groupe d'enregistrements doublons et en créant un nouvel enregistrement consolidé à l'aide des meilleures données.

Entités API

BestOfBreedConfiguration

Pour spécifier les règles de modélisation et de consolidation pour effectuer le job de consolidation Best of Breed.

BestofBreedDetail

Objectif

Spécifier les détails d'un job de consolidation Best of Breed.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Spécifiez le champ à l'aide duquel un seul enregistrement best of breed est créé par la fusion d'un groupe d'enregistrements similaires. Un enregistrement best of breed est créé pour chaque groupe d'enregistrements. Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés.
Best of Breed Configuration	Définissez les règles de modélisation et de consolidation à l'aide desquelles l'enregistrement best of breed doit être créé pour chaque groupe d'enregistrements similaires.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Best of Breed :

Paramètre	Description	Valeur de sortie
Collection Record Type	Identifie les enregistrements modèles et Best of Breed dans une collection d'enregistrements doublons.	Si un enregistrement modèle est défini, les valeurs possibles sont les suivantes : **Primary** Si l'enregistrement est l'enregistrement modèle sélectionné d'une collection. **Secondary** Si l'enregistrement n'est pas l'enregistrement modèle sélectionné d'une collection. **BestOfBreed** Si l'enregistrement est le nouvel enregistrement créé Best of Breed de la collection. Si aucun enregistrement modèle n'est défini, la seule valeur possible est **BestOfBreed**.

Remarque : Outre **Collection Record Type**, les autres colonnes de sortie sont affichées uniquement si elles sont définies lors de la création des conditions de consolidation de la configuration Best of Breed.

Utilisation d'un job Best of Breed MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Best of Breed en créant une instance de `BestofBreedDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

   Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.

   b) Générez les règles de consolidation et de modélisation du job en créant une instance de `BestOfBreedConfiguration`. Dans cette instance :

1. Définissez l'enregistrement modèle pour la consolidation à l'aide d'une instance de `ConsolidationCondition`, qui se compose d'instances `ConsolidationRule`.
2. Définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

   Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

   Remarque : Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

c) Créez une instance de `BestofBreedDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `BestOfBreedConfiguration` créée ci-dessus comme arguments à son constructeur.

   Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `BestofBreedDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `BestofBreedDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `BestofBreedDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `BestofBreedDetail` sur `true` pour compresser la sortie du job.

3. Pour créer un job MapReduce, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `BestofBreedDetail` comme argument.

   La méthode `createJob()` crée le job et renvoie une `List` d'instances de `ControlledJob`.

4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Best of Breed Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Best of Breed en créant une instance de `BestofBreedDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.

   b) Générez les règles de consolidation et de modélisation du job en créant une instance de `BestOfBreedConfiguration`. Dans cette instance :

      1. Définissez l'enregistrement modèle pour la consolidation à l'aide d'une instance de `ConsolidationCondition`, qui se compose d'instances `ConsolidationRule`.
      2. Définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

      Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

      > **Remarque :** Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

   c) Créez une instance de `BestofBreedDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `BestOfBreedConfiguration` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `BestofBreedDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

   e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `BestofBreedDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `BestofBreedDetail`.
g) Définissez l'indicateur `compressOutput` de l'instance `BestofBreedDetail` sur `true` pour compresser la sortie du job.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `BestofBreedDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Duplicate Synchronization

Présentation

Le job Duplicate Synchronization vous permet de déterminer les champs d'un groupe d'enregistrements à copier dans les champs correspondants de tous les enregistrements du groupe.

Entités API

DuplicateSynchronizationConfiguration

Spécifier les règles de consolidation pour effectuer le job de consolidation Duplicate Synchronization.

DuplicateSyncDetail

Objectif

Spécifier les détails d'un job de consolidation Duplicate Synchronization.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Indique le champ à utiliser pour créer des groupes d'enregistrements à synchroniser. Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, pour créer une option Group-By, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Remarque :** S'il n'existe aucun groupe dans l'entrée, définissez ce paramètre sur null. Dans ce cas, les données dans leur entier sont considérées comme un seul groupe.
Configuration de Duplicate Synchronization	Règles en fonction desquelles les champs d'un enregistrement sont copiés dans les autres enregistrements d'une collection.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Colonnes de sortie

Suivant les conditions de consolidation définies dans le paramètre d'entrée *Configuration de Duplicate Synchronization*, des colonnes peuvent être ajoutées à la sortie en plus des colonnes d'entrée, le cas échéant.

Utilisation d'un job Duplicate Synchronization MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Duplicate Synchronization en créant une instance de `DuplicateSyncDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.

a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.

b) Générez les conditions de consolidation du job en créant une instance de `DuplicateSynchronizationConfiguration`. Dans cette instance, définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

> **Remarque :** Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

c) Créez une instance de `DuplicateSyncDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `DuplicateSynchronizationConfiguration` créée ci-dessus comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `DuplicateSyncDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `DuplicateSyncDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `DuplicateSyncDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `DuplicateSyncDetail` sur `true` pour compresser la sortie du job.

3. Créez le job à l'aide de l'instance précédemment créée de `AdvanceMatchFactory` pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `DuplicateSyncDetail` comme argument.
La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.

5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Duplicate Synchronization Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Duplicate Synchronization en créant une instance de `DuplicateSyncDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.

   b) Générez les conditions de consolidation du job en créant une instance de `DuplicateSynchronizationConfiguration`. Dans cette instance, définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

      Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

      > **Remarque :** Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

   c) Créez une instance de `DuplicateSyncDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `DuplicateSynchronizationConfiguration` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `DuplicateSyncDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

   e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `DuplicateSyncDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez

une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `DuplicateSyncDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `DuplicateSyncDetail` sur `true` pour compresser la sortie du job.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `DuplicateSyncDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Filtre

Présentation

Le job Filter conserve ou supprime des enregistrements d'un groupe d'enregistrements en fonction des règles que vous indiquez.

Entités API

FilterConfiguration

Spécifier les règles de consolidation pour effectuer le job de consolidation Filter.

FilterDetail

Objectif

Spécifier les détails d'un job de consolidation Filter.

Paramètres d'entrée

Paramètre	Description
Option Group-By	Indique le champ à utiliser pour créer des groupes d'enregistrements à filtrer. Le job Filter conserve un ou plusieurs enregistrements de chaque groupe. Pour un job *MapReduce*, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Number of Reducer Tasks** Nombre de tâches de réduction requises pour regrouper les enregistrements. Pour un job *Spark*, pour créer une option Group-By, transmettez les arguments : **Colonne GroupBy** Nom de la colonne à l'aide de laquelle les enregistrements doivent être regroupés. **Remarque :** S'il n'existe aucun groupe dans l'entrée, définissez ce paramètre sur null. Dans ce cas, les données dans leur entier sont considérées comme un seul groupe.
Filter Configuration	Définit les conditions de consolidation en fonction desquelles le job conserver un ou plusieurs enregistrements de chaque groupe.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Colonnes de sortie

Les colonnes de sortie sont les mêmes que les colonnes d'entrée. Aucune colonne supplémentaire n'est ajoutée à la sortie.

Utilisation d'un job Filter MapReduce

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Filter en créant une instance de `FilterDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

Utilisez une instance de **GroupbyMROption** à la page 42 pour spécifier la colonne Group-By et le nombre de réducteurs requis.

b) Générez les règles de consolidation du job en créant une instance de `FilterConfiguration`. Dans cette instance, définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

> **Remarque :** Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

c) Créez une instance de `FilterDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `FilterConfiguration` créée ci-dessus comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `FilterDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `FilterDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `FilterDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `FilterDetail` sur `true` pour compresser la sortie du job.

3. Créez le job à l'aide de l'instance précédemment créée de `AdvanceMatchFactory` pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `FilterDetail` comme argument.
La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `AdvanceMatchFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Filter Spark

1. Créez une instance de `AdvanceMatchFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Filter en créant une instance de `FilterDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Spécifiez la colonne à l'aide de laquelle les enregistrements doivent être regroupés en créant une instance de `GroupbyOption`.

      Utiliser une instance de **GroupbySparkOption** à la page 42 pour spécifier la colonne Group-By.

   b) Générez les règles de consolidation du job en créant une instance de `FilterConfiguration`. Dans cette instance, définissez les conditions de consolidation à l'aide d'instances de `ConsolidationCondition`, et en reliant les conditions à l'aide d'opérateurs logiques.

      Chaque instance de `ConsolidationCondition` est définie à l'aide d'une instance `ConsolidationRule` et de son instance `ConsolidationAction` correspondante.

      > **Remarque :** Chaque instance de `ConsolidationRule` peut être définie soit à l'aide d'une seule instance de `SimpleRule`, soit à l'aide d'une hiérarchie d'instances `SimpleRule` enfants et d'instances `ConjoinedRule` imbriquées, liées à l'aide d'opérateurs logiques. Voir **Énumération JoinType** à la page 196 et **Énumération Operation** à la page 195.

   c) Créez une instance de `FilterDetail` en transmettant une instance de type `JobConfig`, l'instance `GroupbyOption` créée et l'instance `FilterConfiguration` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `FilterDetail`.

      Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

   e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `FilterDetail`.

      Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

   f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `FilterDetail`.
   g) Définissez l'indicateur `compressOutput` de l'instance `FilterDetail` sur `true` pour compresser la sortie du job.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `AdvanceMatchFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `FilterDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.
4. Affichez les compteurs pour voir les statistiques de reporting du job.

Jobs du module Data Normalization

Module commun API

DataNormalizationDetail< Type de processus T extends >

Objectif
Spécifier les détails d'un job du module Data Normalization.

DataNormalizationFactory

Objectif
Une classe usine singleton pour créer des instances de jobs Module Data Normalization

Recherche dans la table

Présentation

Le job Table Lookup standardise les termes en s'appuyant sur une forme de ce terme ayant été préalablement validée et en appliquant la version standard.

Entités API

AbstractTableLookupRule

Objectif
Préciser la règle à utiliser pour Table Lookup.

Catégoriser

Objectif
Spécifier la règle de normalisation d’un job Table Lookup.

Identifier

Objectif
Pour indiquer la règle d'identification d’un job Table Lookup.

Normaliser

Objectif
Pour indiquer la règle de normalisation d’un job Table Lookup.

TableLookupDetail

Objectif
Spécifier les détails d'un job Table Lookup.

TableLookupConfiguration

Objectif
Standardiser les termes en fonction d'une forme de ce terme ayant été précédemment validée, et appliquer la version standardisée à tous les enregistrements.

Paramètres d'entrée

Paramètre	Description
Configuration de Table Lookup	Standardiser les termes en fonction d'une forme de ce terme ayant été précédemment validée, et appliquer la version standardisée à tous les enregistrements. Les règles peuvent être de type `Standardize`, `Categorize` ou `Identify`.
Reference Data Path	Spécifier les détails du chemin d'accès aux données de référence.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l’instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l’instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Table Lookup :

Colonne	Description	Valeur de sortie
Destination	Pour les options de règle `Standardize` et `Categorize`, cette colonne de sortie est ajoutée si un nouveau nom de colonne, non présent dans l'entrée, est spécifié comme colonne de destination. Le nom de la colonne est tel que vous l'avez saisi. **Remarque :** Pour la colonne de destination, vous pouvez sélectionner une colonne source existante ou saisir un nouveau nom de colonne.	Valeur normalisée des colonnes sources, mises en correspondance avec les données de table.
Standardization Term Identified	Indique si le terme standardisé a été identifié ou non.	Les valeurs possibles sont `Yes` et `No`.

Utilisation d'un job Table Lookup MapReduce

1. Créez une instance de `DataNormalizationFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Table Lookup en créant une instance de `TableLookupDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Configurez les règles Table Lookup en créant une instance de `TableLookupConfiguration`. Dans cette instance :
      Ajoutez une instance de type `AbstractTableLookupRule`. Cette instance `AbstractTableLookupRule` doit être définie à l'aide de l'une des classes suivantes : `Standardize`, `Categorize` ou `Identify`, correspondant à la catégorie de règle Table Lookup de votre choix.
   b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.
   c) Créez une instance de `TableLookupDetail` en transmettant une instance de type `JobConfig` et les instances `TableLookupConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.
      Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.
   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `TableLookupDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `TableLookupDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `TableLookupDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `TableLookupDetail` sur `true` pour compresser la sortie du job.

3. Pour créer un job MapReduce, utilisez l'instance de `DataNormalizationFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `TableLookupDetail` comme argument.
La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `DataNormalizationFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Table Lookup Spark

1. Créez une instance de `DataNormalizationFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Table Lookup en créant une instance de `TableLookupDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.

a) Configurez les règles Table Lookup en créant une instance de `TableLookupConfiguration`. Dans cette instance :

Ajoutez une instance de type `AbstractTableLookupRule`. Cette instance `AbstractTableLookupRule` doit être définie à l'aide de l'une des classes suivantes : `Standardize`, `Categorize` ou `Identify`, correspondant à la catégorie de règle Table Lookup de votre choix.

b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.

c) Créez une instance de `TableLookupDetail` en transmettant une instance de type `JobConfig` et les instances `TableLookupConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `TableLookupDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `TableLookupDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `TableLookupDetail`.

g) Définissez l'indicateur `compressOutput` de l'instance `TableLookupDetail` sur `true` pour compresser la sortie du job.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `DataNormalizationFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `TableLookupDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Advanced Transformer

Présentation

Le job Advanced Transformer scanne et divise des chaînes de données en différents champs à l'aide de tables ou d'expressions régulières. Il extrait un terme ou un nombre de mots spécifique à gauche ou à droite d'un terme.

Entités API

AbstractAdvancedTransformerRules

Objectif

Classe parente permettant d'indiquer les règles d'un job Advanced Transformer.

AdvancedTransformerDetail

Objectif
Spécifier les détails d'un job Advanced Transformer.

AdvancedTransformerConfiguration

Objectif
Analyser et diviser les chaînes de données en plusieurs champs à l'aide de tables ou d'expressions régulières.

RegularExpressionExtraction

Objectif
Pour spécifier les règles d'extraction des données à l'aide d'expressions régulières.

RegularExpressionGroupItem

Objectif
Pour spécifier une partie d'une expression régulière parente. Chaque partie d'une expression régulière parente peut être stockée dans un champ de sortie différent.

TableDataExtraction

Objectif
Définir les règles d'extraction des données de la table.

Paramètres d'entrée

Paramètre	Description
Advanced Transformer Configuration	Analyser et diviser les chaînes de données en plusieurs champs à l'aide de tables ou d'expressions régulières. Permet l'extraction d'un terme ou d'un nombre de mots spécifique à gauche ou à droite d'un terme. Les données extraites et non extraites sont placées dans un champ existant ou dans un nouveau champ. Les règles Advanced Transformer peuvent être définies à l'aide d'une instance de type `AdvancedTransformerConfiguration`. Cette instance doit être une instance de `TableDataExtraction` ou de `RegularExpressionExtraction`.
Reference Data Path	Spécifier les détails du chemin d'accès aux données de référence.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Advanced Transformer :

Colonne	Description	Valeur de sortie
Non-Extracted Data	Cette colonne de sortie est ajoutée si un nouveau nom de colonne, non présent dans l'entrée, est spécifié en tant que colonne Non-Extracted Data. Le nom de la colonne est tel que vous l'avez saisi. **Remarque :** Pour la colonne Non-Extracted Data, vous pouvez sélectionner une colonne source existante ou saisir un nouveau nom de colonne.	Données non extraites de l'enregistrement respectif basées sur le terme spécifié.
Extracted Data	Cette colonne de sortie est ajoutée si un nouveau nom de colonne, non présent dans l'entrée, est spécifié en tant que colonne Extracted Data. Le nom de la colonne est tel que vous l'avez saisi. **Remarque :** Pour la colonne Extracted Data, vous pouvez sélectionner une colonne source existante ou saisir un nouveau nom de colonne.	Données extraites de l'enregistrement respectif basées sur le terme spécifié.
Advanced Transform Term Identified	Indique si le terme a été identifié ou non.	Les valeurs possibles sont `Yes` et `No`.

Utilisation d'un job Advanced Transformer MapReduce

1. Créez une instance de `DataNormalizationFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Advanced Transformer en créant une instance de `AdvancedTransformerDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Configurez les règles Advanced Transformer en créant une instance de`AdvancedTransformerConfiguration`. Dans cette instance :

   Ajoutez une instance de type `AbstractAdvancedTransformerRules`. Cette instance `AbstractAdvancedTransformerRules` doit être définie à l'aide de l'une des classes suivantes : `TableDataExtraction` ou `RegularExpressionExtraction`, correspondant à la catégorie de règle Advanced Transformer de votre choix.

b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.

c) Créez une instance de `AdvancedTransformerDetail` en transmettant une instance de type `JobConfig` et les instances `AdvancedTransformerConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `AdvancedTransformerDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `AdvancedTransformerDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `AdvancedTransformerDetail`.

3. Pour créer un job MapReduce, utilisez l'instance de `DataNormalizationFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `AdvancedTransformerDetail` comme argument.
La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `DataNormalizationFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Advanced Transformer Spark

1. Créez une instance de `DataNormalizationFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Advanced Transformer en créant une instance de `AdvancedTransformerDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.

a) Configurez les règles Advanced Transformer en créant une instance de`AdvancedTransformerConfiguration`. Dans cette instance :

Ajoutez une instance de type `AbstractAdvancedTransformerRules`. Cette instance `AbstractAdvancedTransformerRules` doit être définie à l'aide de l'une des classes

suivantes : `TableDataExtraction` ou `RegularExpressionExtraction`, correspondant à la catégorie de règle Advanced Transformer de votre choix.

b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.

c) Créez une instance de `AdvancedTransformerDetail` en transmettant une instance de type `JobConfig` et les instances `AdvancedTransformerConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `AdvancedTransformerDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `AdvancedTransformerDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `AdvancedTransformerDetail`.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `DataNormalizationFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `AdvancedTransformerDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Jobs du module Universal Addressing

Module commun API

UniversalAddressingDetail<T extends ProcessType>

Objectif
Spécifier les détails d'un job du module Universal Name.

UniversalAddressingFactory

Objectif
Une classe usine singleton pour créer des instances de jobs du module Universal Addressing.

Validate Address

Entités API

UAMAddressingDetail<T extends ProcessType>

Objectif
Spécifier les détails d'un job Validate Address.

UniversalAddressEngineConfiguration

Objectif
Définir différentes configurations, telles que le *chemin d'accès aux données de référence* et le *chemin d'accès à l'exécution COBOL*, requises pour créer et exécuter le job Validate Address.

Il s'agit de paramètres ponctuels.

UAMAddressingFactory

Objectif
Une classe usine singleton pour créer des instances de jobs Validate Address.

Cette instance est utilisée pour générer les compteurs de reporting et les rapports CASS.

UniversalAddressGeneralConfiguration

Objectif
Définir les configurations JVM requises pour créer et exécuter le job Validate Address.

UniversalAddressValidateInputConfiguration

Objectif
Configurer les paramètres d'entrée pour créer et exécuter le job Validate Address. Il s'agit d'un paramètre de règle qui comporte plusieurs options. Ces paramètres varient pour chaque job.

Paramètres d'entrée

Paramètre	Description
Configuration du moteur Universal Address	Pour définir différentes configurations d'exécution du job : **1.** Chemin d'accès à la base de données DPV **2.** Chemin d'accès à la base de données Suite Link **3.** Chemin d'accès à la base de données EWS **4.** Chemin d'accès à la base de données RDI **5.** Chemin d'accès à la base de données Lacs **6.** Reference Data Path **7.** Chemin d'accès à l'exécution COBOL **8.** Répertoire de modules

Paramètre	Description
Configuration d'entrée Universal Address Validate	

Paramètre	Description
	Pour configurer les paramètres d’entrée : **1.** Adresse standard de sortie **2.** Éléments d'adresse de sortie **3.** Données postales de sortie **4.** Entrée analysée de sortie **5.** Blocs d'adresse de sortie **6.** Sortie formatée sur un échec **7.** Casse de sortie **8.** Séparateur de code postal de sortie **9.** Caractères multinationaux de sortie **10.** Exécution de DPV **11.** Exécution de RDI **12.** Exécution d'ESM **13.** Exécution d'ASM **14.** Exécution d'EWS **15.** Exécution de LACS Link **16.** Exécution de LOT **17.** Échec de correspondance CMRA **18.** Extraction de raison sociale **19.** Extraction d'Urb **20.** Rapport de sortie 3553 **21.** Rapport de sortie SERP **22.** Résumé du rapport de sortie SERP **23.** Détails CASS de sortie **24.** Codes de renvoi au niveau des champs de sortie **25.** Conservation des correspondances multiples **26.** Nombre maximal de résultats **27.** Format d'adresse standard **28.** Ligne PMB d’adresse standard **29.** Format de nom de ville **30.** Long format de ville Vanity **31.** Format de pays de sortie **32.** Pays d'origine **33.** Exactitude de correspondance de rue **34.** Exactitude de correspondance de société **35.** Exactitude de correspondance cardinale **36.** Logique d'adresse double **37.** Condition d'état de succès DPV **38.** Nom de fichier de liste du rapport **39.** Nom de processeur de liste du rapport **40.** Numéro de liste du rapport **41.** Adresse d'expéditeur du rapport **42.** Nom d'expéditeur du rapport **43.** Ligne de ville d'expéditeur du rapport **44.** Recherche de ligne d’adresse sur un échec **45.** Alias de rue de sortie

Paramètre	Description
	46. Blocs VeriMove de sortie **47.** DPV détermine No Stat **48.** DPV détermine Vacancy **49.** Alias abrégé de sortie **50.** Alias préféré de sortie **51.** Ville préférée de sortie **52.** Exécution de Suite Link **53.** Suppression de Zplus Phantom Carrier R777
Configuration générale Universal Address	Pour définir des configurations JVM : **1.** Type de fichier DPV **2.** Modèle de mémoire DPV **3.** Modèle de mémoire Lacs Link **4.** Modèle de mémoire Suite Link
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.
Compress Output	Indicateur permettant de spécifier si la sortie doit être compressée. Définissez cette valeur sur `true` pour compresser la sortie.

Paramètre	Description
Rapports CASS	Configurations permettant de générer le rapport CASS. Appelez l'une des méthodes surchargés `generateCASSReport()` à l'aide de l'instance `UAMAddressingFactory`. Les rapports CASS sont générés au format PDF. Les paramètres sont les suivants : **Compteurs** Carte des compteurs à inclure dans le rapport CASS. **Nom du job** Nom du job. Il est inclus dans le nom de fichier du rapport CASS. **Chemin d'accès** Répertoire dans lequel le rapport CASS créé est placé. Il s'agit d'une valeur d'entrée facultative pour les rapports CASS. Le `path` doit pointer vers l'emplacement du cluster ou du client, suivant que le job SDK est exécuté dans un environnement de cluster ou sur votre poste client, respectivement. **Remarque :** Si le `path` n'est pas spécifié, le nouveau rapport CASS est placé dans le répertoire de travail en cours. **Type de rapport** Type de rapport CASS à générer. Vous pouvez spécifier une ou plusieurs valeurs de **Énumération UAMCASSReportType** à la page 205.

Colonnes de sortie

1. AdditionalInputData
2. AddressLine1
3. AddressLine2
4. AddressLine3
5. AddressLine4:
6. AddressLine5
7. City
8. Country
9. FirmName
10. PostalCode
11. PostalCode.AddOn
12. PostalCode.Base
13. StateProvince
14. USUrbanName
15. AdditionalInputData
16. ApartmentLabel
17. ApartmentLabel2

18. ApartmentNumber
19. ApartmentNumber2
20. HouseNumber
21. LeadingDirectional
22. POBox
23. PrivateMailbox
24. PrivateMailbox.Type
25. RRHC
26. StateProvince
27. StreetName
28. StreetSuffix
29. TrailingDirectional
30. USUrbanName
31. ApartmentLabel.Input
32. ApartmentNumber.Input
33. City.Input
34. Country.Input
35. FirmName.Input
36. HouseNumber.Input
37. LeadingDirectional.Input
38. POBox.Input
39. PostalCode.Input
40. PrivateMailbox.Input
41. PrivateMailbox.Type.Input
42. RRHC.Input
43. StateProvince.Input
44. StreetName.Input
45. StreetSuffix.Input
46. TrailingDirectional.Input
47. USUrbanName.Input
48. PostalBarCode
49. USAltAddr
50. USBCCheckDigit
51. USCarrierRouteCode
52. USCongressionalDistrict
53. USCountyName
54. USFinanceNumber
55. USFIPSCountyNumber
56. USLACS
57. USLastLineNumber
58. AddressFormat

59. Confidence
60. CouldNotValidate
61. CountryLevel
62. MatchScore
63. MultimatchCount
64. MultipleMatches
65. ProcessedBy
66. RecordType
67. RecordType.Default
68. État
69. Status.Code
70. Status.Description
71. AddressRecord.Result
72. ApartmentLabel.Result
73. ApartmentNumber.Result
74. City.Result
75. Country.Result
76. FirmName.Result
77. HouseNumber.Result
78. LeadingDirectional.Result
79. POBox.Result
80. PostalCode.Result
81. PostalCodeCity.Result
82. PostalCode.Source
83. PostalCode.Type
84. RRHC. Résultat
85. RRHC. Type
86. StateProvince.Result
87. Street.Result
88. StreetName.AbbreviatedAlias.Result
89. StreetName.Alias.Type
90. StreetName.PreferredAlias.Result
91. StreetName.Result
92. StreetSuffix.Result
93. TrailingDirectional.Result
94. USUrbanName.Result
95. USLOTCode
96. USLOTHex
97. USLOTSequence
98. USLACS. ReturnCode
99. RDI

100 DPV
101 CMRA
102 DPVFootnote
103 DPVVacant
104 DPVNoStat
105 SuiteLinkReturnCode
106 SuiteLinkMatchCode
107 SuiteLinkFidelity
108 VeriMoveDataBlock

Remarque : Pour obtenir les descriptions des champs, reportez-vous à la rubrique *Validate Address* du *Guide Addressing* de Spectrum™ Technology Platform.

Utilisation d'un job Validate Address MapReduce

Avertissement : Avant de créer et d'exécuter le premier job Validate Address, assurez-vous que le service Acushare est en cours d'exécution. Pour connaître les étapes, reportez-vous à la section **Arrêt du service acushare** à la page 12.

1. Créez une instance de `UAMAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address en créant une instance de `UAMAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Pour configurer les paramètres d'entrée du job, créez une instance de `UniversalAddressValidateInputConfiguration`.

      Définissez les valeurs des différents champs requis de cette instance en utilisant les énumérations **Énumération PreferredCity** à la page 203, **Énumération CasingType** à la page 202, **Énumération CityNameFormat** à la page 202, **Énumération OutputCountryFormat** à la page 202, **Énumération StandardAddressFormat** à la page 202, **Énumération StandardAddressPMBLine** à la page 203, **Énumération StreetMatchingStrictness** à la page 203, **Énumération FirmMatchingStrictness** à la page 203, **Énumération DirectionalMatchingStrictness** à la page 203, **Énumération DualAddressLogic** à la page 202 et **Énumération DPVSuccessStatusCondition** à la page 204, le cas échéant.

      Important : Pour lancer Validate Address en mode certifié CASS™, définissez les champs `outputReport3553`, `outputCASSDetail` et `outputReportSummary` de cette instance sur `true`. Le contenu des rapports CASS est valide uniquement lorsque le job est exécuté en mode certifié CASS™. Sinon, des PDF de rapports vierges sont générés.

   b) Définissez les détails du *chemin d'accès aux données de référence* en créant une instance de `LocalReferenceDataPath`.
   c) Pour configurer les différents paramètres d'exécution du job, créez une instance de `UAMUSAddressingEngineConfiguration` en transmettant l'instance `LocalReferenceDataPath` précédemment créée, le *chemin d'accès à l'exécution COBOL*

et le *chemin d'accès au répertoire de modules* sous forme de valeurs `String` comme arguments à son constructeur.

Une fois l'instance `UAMUSAddressingEngineConfiguration` créée, définissez les valeurs de ses différents champs requis.

d) Pour configurer les paramètres JVM, créez une instance de `UniversalAddressGeneralConfiguration`.

Utiliser les énumérations **Énumération DPVFileType** à la page 204, **Énumération DPVMemoryModel** à la page 204, **Énumération LacsLinkMemoryModel** à la page 204 et **Énumération SuiteLinkMemoryModel** à la page 204.

e) Créez une instance de `UAMAddressingDetail` en transmettant une instance de type `JobConfig` et les instances `UAMUSAddressingEngineConfiguration`, `UniversalAddressGeneralConfiguration` et `UniversalAddressValidateInputConfiguration` créées précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

1. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `UAMAddressingDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

2. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `UAMAddressingDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

3. Définissez le nom du job à l'aide du champ `jobName` de l'instance `UAMAddressingDetail`.
4. Définissez l'indicateur `compressOutput` de l'instance `UAMAddressingDetail` sur `true` pour compresser la sortie du job.

3. Pour créer un job MapReduce, utilisez l'instance de `UAMAddressingFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `UAMAddressingDetail` comme argument.
   La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job, utilisez l'instance précédemment créée `UAMAddressingFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.
   Une `Map` de compteurs est reçue.

6. Pour générer les rapports CASS après l'exécution réussie d'un job, utilisez l'instance précédemment créée de `UAMAddressingFactory` pour appeler la méthode `generateCASSReport()`. Vous pouvez appeler l'une des versions surchargées de la méthode `generateCASSReport()`.

   Suivant la signature de la méthode `generateCASSReport()` utilisée, transmettez comme arguments la `Map` des compteurs de rapport dérivés à l'étape précédente, le `jobName`, le `path` où le rapport CASS généré doit être enregistré et le `reportType` requis à créer.

   Le `path` doit pointer vers l'emplacement du cluster ou du client, suivant que le job SDK est exécuté dans un environnement de cluster ou sur votre poste client, respectivement.

   > **Remarque :** Si le `path` n'est pas spécifié, le nouveau rapport CASS est placé dans le répertoire de travail en cours.

   Les valeurs du paramètre `reportType` doivent provenir de **Énumération UAMCASSReportType** à la page 205. Vous pouvez spécifier un ou plusieurs types de rapport dans ce paramètre.

Utilisation d'un job Validate Address Spark

Avertissement : Avant de créer et d'exécuter le premier job Validate Address, assurez-vous que le service Acushare est en cours d'exécution. Pour connaître les étapes, reportez-vous à la section **Arrêt du service acushare** à la page 12.

1. Créez une instance de `UAMAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address en créant une instance de `UAMAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Pour configurer les paramètres d'entrée du job, créez une instance de `UniversalAddressValidateInputConfiguration`.

      Définissez les valeurs des différents champs requis de cette instance en utilisant les énumérations **Énumération PreferredCity** à la page 203, **Énumération CasingType** à la page 202, **Énumération CityNameFormat** à la page 202, **Énumération OutputCountryFormat** à la page 202, **Énumération StandardAddressFormat** à la page 202, **Énumération StandardAddressPMBLine** à la page 203, **Énumération StreetMatchingStrictness** à la page 203, **Énumération FirmMatchingStrictness** à la page 203, **Énumération DirectionalMatchingStrictness** à la page 203, **Énumération DualAddressLogic** à la page 202 et **Énumération DPVSuccessStatusCondition** à la page 204, le cas échéant.

      Important : Pour lancer Validate Address en mode certifié CASS™, définissez les champs `outputReport3553`, `outputCASSDetail` et `outputReportSummary` de cette instance sur `true`. Le contenu des rapports CASS est valide uniquement lorsque le job est exécuté en mode certifié CASS™. Sinon, des PDF de rapports vierges sont générés.

   b) Définissez les détails du *chemin d'accès aux données de référence* en créant une instance de `LocalReferenceDataPath`.

c) Pour configurer les différents paramètres d'exécution du job, créez une instance de `UAMUSAddressingEngineConfiguration` en transmettant l'instance `LocalReferenceDataPath` précédemment créée, le *chemin d'accès à l'exécution COBOL* et le *chemin d'accès au répertoire de modules* sous forme de valeurs `String` comme arguments à son constructeur.

Une fois l'instance `UAMUSAddressingEngineConfiguration` créée, définissez les valeurs de ses différents champs requis.

d) Pour configurer les paramètres JVM, créez une instance de `UniversalAddressGeneralConfiguration`.

Utiliser les énumérations **Énumération DPVFileType** à la page 204, **Énumération DPVMemoryModel** à la page 204, **Énumération LacsLinkMemoryModel** à la page 204 et **Énumération SuiteLinkMemoryModel** à la page 204.

e) Créez une instance de `UAMAddressingDetail` en transmettant une instance de type `JobConfig` et les instances `UAMUSAddressingEngineConfiguration`, `UniversalAddressGeneralConfiguration` et `UniversalAddressValidateInputConfiguration` créées précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

1. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `UAMAddressingDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

2. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `UAMAddressingDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

3. Définissez le nom du job à l'aide du champ `jobName` de l'instance `UAMAddressingDetail`.
4. Définissez l'indicateur `compressOutput` de l'instance `UAMAddressingDetail` sur `true` pour compresser la sortie du job.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `UAMAddressingFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `UAMAddressingDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job, utilisez l'instance précédemment créée `UAMAddressingFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.
   Une `Map` de compteurs est reçue.
5. Pour générer les rapports CASS après l'exécution réussie d'un job, utilisez l'instance précédemment créée de `UAMAddressingFactory` pour appeler la méthode `generateCASSReport()`. Vous pouvez appeler l'une des versions surchargées de la méthode `generateCASSReport()`.

   Suivant la signature de la méthode `generateCASSReport()` utilisée, transmettez comme arguments la `Map` des compteurs de rapport dérivés à l'étape précédente, le `jobName`, le `path` où le rapport CASS généré doit être enregistré et le `reportType` requis à créer.

   Le `path` doit pointer vers l'emplacement du cluster ou du client, suivant que le job SDK est exécuté dans un environnement de cluster ou sur votre poste client, respectivement.

   > **Remarque :** Si le `path` n'est pas spécifié, le nouveau rapport CASS est placé dans le répertoire de travail en cours.

   Les valeurs du paramètre `reportType` doivent provenir de **Énumération UAMCASSReportType** à la page 205. Vous pouvez spécifier un ou plusieurs types de rapport dans ce paramètre.

Validate Address Global

Entités API

GlobalAddressingDetail<T extends ProcessType>

Objectif
Spécifier les détails d'un job Validate Address Global.

GlobalAddressingEngineConfiguration

Objectif
Définir les configurations de base de données requises pour créer et exécuter le job Validate Address Global.

GlobalAddressingFactory

Objectif
Une classe usine singleton pour créer des instances de jobs Validate Address Global.

GlobalAddressingGeneralConfiguration

Objectif
Définir les configurations JVM requises pour créer et exécuter le job Validate Address Global.

GlobalAddressingInputConfiguration

Objectif

Configurer les paramètres d'entrée pour créer et exécuter le job Validate Address Global.

Paramètres d'entrée

Paramètre	Description
Configuration du moteur Validate Address Global	Pour définir des configurations de base de données : **1.** Type de base de données **2.** Type de préchargement **3.** Reference Data Path **4.** Si tous les pays sont pris en charge. Si ce n'est pas, liste des pays pris en charge
Configuration de l'entrée Validate Address Global	Pour configurer ces paramètres pour l'entrée : **1.** Type État ou Province dans le résultat **2.** Portée de la correspondance dans le processus **3.** Forcer le code pays ISO3 dans l'entrée **4.** Code pays ISO3 par défaut dans l'entrée **5.** Délimiteur de format dans l'entrée **6.** Délimiteur de format dans le résultat **7.** Inclure les entrées dans le résultat **8.** Type de pays dans le résultat **9.** Niveau d'optimisation du processus **10.** Langue préférée du résultat **11.** Mode de processus **12.** Script préféré dans le résultat **13.** Nombre maximal de résultats **14.** Casse du résultat
Configuration générale Validate Address Global	Pour définir des configurations JVM : **1.** Taille de cache **2.** Nombre maximal de threads **3.** Nombre maximal d'objets d'adresse **4.** Plages d'extension **5.** Extension de plage flexible **6.** Journalisation des transactions active **7.** Utilisation de mémoire maximale en Mo
Code de déverrouillage	Pour déverrouiller les données de la base de données.

Paramètre	Description
Reference Data Path	Spécifier les détails du chemin d'accès aux données de référence. **Remarque :** Pour les jobs UAM, les données de référence doivent être placées uniquement sur des nœuds de données locaux du cluster.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.

Colonnes de sortie

Données d'adresse

1. AddressBlock1-9
2. AddressLine1-6
3. AdministrativeDistrict
4. ApartmentLabel
5. ApartmentNumber
6. BlockName
7. BuildingName
8. City
9. City.AddInfo
10. City.SortingCode
11. Contact
12. Country

13. Comté
14. FirmName
15. Floor
16. HouseNumber
17. LastLine
18. LeadingDirectional
19. Localité
20. POBox
21. PostalCode
22. PostalCode.AddOn
23. PostalCode.Base
24. Room
25. SecondaryStreet
26. StateProvince
27. StreetName
28. StreetSuffix
29. SubBuilding
30. Suburb
31. Territory
32. TrailingDirectional

Données de saisie d'origine

1. AddressLine1.Input
2. AddressLine2.Input
3. AddressLine3.Input
4. AddressLine4.Input
5. AddressLine5.Input
6. AddressLine6.Input
7. City.Input
8. StateProvince.Input
9. PostalCode.Input
10. Contact.Input
11. Country.Input
12. FirmName.Input
13. Street.Input
14. Number.Input
15. Building.Input
16. SubBuilding.Input
17. DeliveryService.Input

Avertissement : Les champs d'entrée `AddressLine2.Input`,`AddressLine3.Input`, `AddressLine4.Input`,`AddressLine5.Input`, et `AddressLine6.Input` sont inclus dans la

sortie uniquement si le champ `resultIncludeInputs` de la classe `GlobalAddressingInputConfiguration` est défini sur `true`. Sinon, seuls les champs `AddressLineX.input` faisant partie de l'entrée sont inclus dans la sortie.

Codes de résultat

1. AddressType
2. Confidence
3. CountOverflow
4. ElementInputStatus
5. ElementRelevance
6. ElementResultStatus
7. MailabilityScore
8. ModeUsed
9. MultimatchCount
10. ProcessStatus
11. État
12. Status.Code
13. Status.Description

> **Remarque :** Pour obtenir les descriptions des champs, reportez-vous à la rubrique *Validate Address Global* du *Guide Addressing* de Spectrum™ Technology Platform.

Utilisation d'un job Validate Address Global MapReduce

1. Créez une instance de `GlobalAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address Global en créant une instance de `GlobalAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Configurez les paramètres d'initialisation JVM en créant une instance de `GlobalAddressingGeneralConfiguration`.

   Utiliser les énumérations **Énumération CacheSize** à la page 201, **Énumération RangesToExpand** à la page 201 et **Énumération FlexibleRangeExpansion** à la page 201.
   b) Définissez les détails du chemin d'accès aux données de référence en créant une instance de `LocalReferenceDataPath`.
   c) Configurez les paramètres de base de données nécessaires en créant une instance de `GlobalAddressingEngineConfiguration` en transmettant l'instance `LocalReferenceDataPath` ci-dessus comme argument.
      1. Définissez le *type de préchargement* dans cette instance à l'aide de l'énumération **Énumération PreloadingType** à la page 198.
      2. Définissez le *type de base de données* à l'aide de **Énumération DatabaseType** à la page 197.

3. Définissez les pays pris en charge à l'aide de **Énumération CountryCodes** à la page 198.
4. Si tous les pays sont pris en charge, définissez l'attribut `isAllCountries` sur true. Sinon, indiquez la liste séparée par des virgules des valeurs **Énumération CountryCodes** à la page 198 de la valeur de chaîne `supportedCountries`.

d) Configurez les paramètres d'entrée en créant une instance de `GlobalAddressingInputConfiguration`.

Pour définir les valeurs des différents champs de cette instance, utilisez les énumérations **Énumération CountryCodes** à la page 198, **Énumération StateProvinceType** à la page 198, **Énumération CountryType** à la page 198, **Énumération PreferredScript** à la page 199, **Énumération PreferredLanguage** à la page 199, **Énumération Casing** à la page 199, **Énumération OptimizationLevel** à la page 199, **Énumération Mode** à la page 199 et **Énumération MatchingScope** à la page 200, le cas échéant.

e) Définissez la clé de déverrouillage des données sous forme de valeur `String` dans une `List`.

f) Créez une instance de `GlobalAddressingDetail`, en transmettant une instance de type `JobConfig`, la `List` des valeurs de code de déverrouillage, l'instance `GlobalAddressingEngineConfiguration` et l'instance `GlobalAddressingInputConfiguration` créée précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

1. Définissez les configurations d'initialisation JVM en définissant le champ `generalConfiguration` de l'instance `GlobalAddressingDetail` sur l'instance `GlobalAddressingGeneralConfiguration` créée ci-dessus.
2. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `GlobalAddressingDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

3. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `GlobalAddressingDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

4. Définissez le nom du job à l'aide du champ `jobName` de l'instance `GlobalAddressingDetail`.

3. Pour créer un job MapReduce, utilisez l'instance de `GlobalAddressingFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `GlobalAddressingDetail` comme argument.
   La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `GlobalAddressingFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Validate Address Global Spark

1. Créez une instance de `GlobalAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address Global en créant une instance de `GlobalAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Configurez les paramètres d'initialisation JVM en créant une instance de `GlobalAddressingGeneralConfiguration`.
      Utiliser les énumérations **Énumération CacheSize** à la page 201, **Énumération RangesToExpand** à la page 201 et **Énumération FlexibleRangeExpansion** à la page 201.
   b) Définissez les détails du chemin d'accès aux données de référence en créant une instance de `LocalReferenceDataPath`.
   c) Configurez les paramètres de base de données nécessaires en créant une instance de `GlobalAddressingEngineConfiguration` en transmettant l'instance `LocalReferenceDataPath` ci-dessus comme argument.
      1. Définissez le *type de préchargement* dans cette instance à l'aide de l'énumération **Énumération PreloadingType** à la page 198.
      2. Définissez le *type de base de données* à l'aide de **Énumération DatabaseType** à la page 197.
      3. Définissez les pays pris en charge à l'aide de **Énumération CountryCodes** à la page 198.
      4. Si tous les pays sont pris en charge, définissez l'attribut `isAllCountries` sur true. Sinon, indiquez la liste séparée par des virgules des valeurs **Énumération CountryCodes** à la page 198 de la valeur de chaîne `supportedCountries`.
   d) Configurez les paramètres d'entrée en créant une instance de `GlobalAddressingInputConfiguration`.
      Pour définir les valeurs des différents champs de cette instance, utilisez les énumérations **Énumération CountryCodes** à la page 198, **Énumération StateProvinceType** à la page 198, **Énumération CountryType** à la page 198, **Énumération PreferredScript** à la page 199, **Énumération PreferredLanguage** à la page 199, **Énumération Casing** à la page 199, **Énumération OptimizationLevel** à la page 199, **Énumération Mode** à la page 199 et **Énumération MatchingScope** à la page 200, le cas échéant.

e) Définissez la clé de déverrouillage des données sous forme de valeur `String` dans une `List`.

f) Créez une instance de `GlobalAddressingDetail`, en transmettant une instance de type `JobConfig`, la `List` des valeurs de code de déverrouillage, l'instance `GlobalAddressingEngineConfiguration` et l'instance `GlobalAddressingInputConfiguration` créée précédemment comme arguments à son constructeur.

Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

1. Définissez les configurations d'initialisation JVM en définissant le champ `generalConfiguration` de l'instance `GlobalAddressingDetail` sur l'instance `GlobalAddressingGeneralConfiguration` créée ci-dessus.
2. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `GlobalAddressingDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

3. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `GlobalAddressingDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

4. Définissez le nom du job à l'aide du champ `jobName` de l'instance `GlobalAddressingDetail`.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `GlobalAddressingFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `GlobalAddressingDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Validate Address Loqate

Entités API

LoqateAddressingDetail<T extends ProcessType>

Objectif
Spécifier les détails d'un job Validate Address Loqate.

LoqateAddressingEngineConfiguration

Objectif
Définir les configurations de base de données requises pour créer et exécuter le job Validate Address Loqate.

LoqateAddressingFactory

Objectif
Une classe usine singleton pour créer des instances de jobs Validate Address Loqate.

LoqateAddressingGeneralConfiguration

Objectif
Définir les configurations JVM requises pour créer et exécuter le job Validate Address Loqate.

LoqateAddressingValidateConfiguration

Objectif
Configurer les paramètres d'entrée pour créer et exécuter le job Validate Address Loqate job.

Paramètres d'entrée

Paramètre	Description
Configuration du moteur Validate Address Loqate	Pour définir les configurations pour réaliser des validations : **1.** Verbose **2.** Outil Infos **3.** Format d'adresse de sortie **4.** Entrée de journal **5.** Sortie de journal **6.** Nom du fichier journal **7.** Seuil absolu du score de correspondance **8.** Facteur de seuil du score de correspondance **9.** Nombre maximal de résultats de codes postaux **10.** Correspondance de référence stricte
Configuration de Validate Address Loqate Validate	Pour configurer ces paramètres pour l'entrée : **1.** Inclusion d'une adresse standard **2.** Inclusion des éléments d'adresse en correspondance **3.** Éléments d'adresse d'entrée normalisés **4.** Renvoi de blocs de données d'adresse **5.** Casse de sortie **6.** Inclusion de codes de résultat pour les champs individuels **7.** Renvoi de plusieurs adresses **8.** Échec en cas de correspondances multiples trouvées **9.** Nombre d'adresses multiples **10.** Format de pays **11.** Pays par défaut **12.** Alphabet du script **13.** Renvoi des champs d'adresse géocodée **14.** Niveau d'acceptation **15.** Score de correspondance minimal **16.** Formatage des données à l'aide des conventions AMAS **17.** Gestion des doublons **18.** Gestion des doublons de champ unique **19.** Gestion des doublons de champs multiples **20.** Gestion des doublons de champs non standard **21.** Gestion des doublons de champs de sortie

Paramètre	Description
Configuration générale Validate Address Loqate	Pour définir des configurations JVM : **1.** Objets inactifs maximaux **2.** Objets inactifs minimaux **3.** Objets actifs maximaux **4.** Délai d'attente maximal **5.** Action lorsque épuisé **6.** Test sur emprunt **7.** Test sur retour **8.** Test quand inactif **9.** Temps entre les exécutions d'éviction en millisecondes **10.** Nombre de tests par exécution d'éviction **11.** Temps d'inactivité mini. avant éviction en millisecondes
Reference Data Path	Spécifier les détails du chemin d'accès aux données de référence. **Remarque :** Pour les jobs UAM, les données de référence doivent être placées uniquement sur des nœuds de données locaux du cluster.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.

Colonnes de sortie

1. AdditionalInputData
2. AddressLine1-4
3. City
4. Country
5. FirmName
6. PostalCode
7. PostalCode.AddOn
8. PostalCode.Base
9. StateProvince
10. AddressBlock1-9
11. ApartmentLabel
12. ApartmentNumber
13. ApartmentNumber2
14. Building

15. City
16. Country
17. County *
18. FirmName
19. HouseNumber
20. LeadingDirectional
21. POBox
22. PostalCode
23. Principality *
24. StateProvince
25. StreetAlias
26. StreetName
27. StreetSuffix
28. Subcity *
29. Substreet *
30. TrailingDirectional
31. ApartmentLabel.Input
32. ApartmentNumber.Input
33. City.Input
34. Country.Input
35. County.Input *
36. FirmName.Input
37. HouseNumber.Input
38. LeadingDirectional.Input
39. POBox.Input
40. PostalCode.Input
41. Principality.Input *
42. StateProvince.Input
43. StreetAlias.Input
44. StreetName.Input
45. StreetSuffix.Input
46. Subcity.Input *
47. Substreet.Input *
48. TrailingDirectional.Input
49. Geocode.MatchCode
50. Latitude
51. Longitude
52. SearchDistance
53. Confidence
54. CouldNotValidate
55. MatchScore

56. ProcessedBy
57. État
58. Status.Code
59. Status.Description
60. ApartmentLabel.Result
61. ApartmentNumber.Result
62. City.Result
63. Country.Result
64. County.Result *
65. FirmName.Result
66. HouseNumber.Result
67. LeadingDirectional.Result
68. POBox.Result
69. PostalCode.Result
70. PostalCode.Type
71. Principality.Result *
72. StateProvince.Result
73. StreetAlias.Result
74. StreetName.Result
75. StreetSuffix.Result
76. Subcity.Result *
77. Substreet.Result *
78. TrailingDirectional.Result
79. Barcode
80. DPID
81. FloorNumber
82. FloorType
83. PostalBoxNum

*Il s'agit d'un sous-champ, qui peut ne contenir aucune donnée.

Tableau 1 : Codes de correspondance de centroïde de code postal/rue/ville

Élément	Code de correspondance
Point d'adresse	P4
Point d'adresse interpolé	I4
Centroïde de rue	P3
Centroïde de code postal/ville	A3/P2/A2

Remarque : Pour obtenir les descriptions des champs, reportez-vous à la rubrique *Validate Address Loqate* du *Guide Addressing* de Spectrum™ Technology Platform.

Utilisation d'un job Validate Address Loqate MapReduce

1. Créez une instance de `LoqateAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address Loqate en créant une instance de `LoqateAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Configurez les paramètres d'initialisation JVM en créant une instance de `LoqateAddressingGeneralConfiguration`.

      Utilisez l'énumération **Énumération ExhaustedAction** à la page 200.
   b) Configurez les paramètres de base de données nécessaires en créant une instance de `LoqateAddressingEngineConfiguration` et définissez les différents champs.
   c) Configurez les paramètres de validation d'adresse en créant une instance de `LoqateAddressingValidateConfiguration`.

      Pour définir les valeurs des différents champs de cette instance, utilisez les énumérations **Énumération AcceptanceLevel** à la page 200, **Énumération CountryCodes** à la page 198, **Énumération OutputCasing** à la page 201, **Énumération CountryFormat** à la page 201 et **Énumération ScriptAlphabet** à la page 201.
   d) Définissez les détails du chemin d'accès aux données de référence en créant une instance de `LocalReferenceDataPath`.
   e) Créez une instance de `LoqateAddressingDetail` en transmettant une instance de type `JobConfig`, l'instance `LocalReferenceDataPath` et l'instance `LoqateAddressingValidateConfiguration` créée ci-dessus comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.

      1. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `LoqateAddressingDetail`.

         Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

      2. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `LoqateAddressingDetail`.

         Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

3. Définissez le nom du job à l'aide du champ `jobName` de l'instance `LoqateAddressingDetail`.

3. Pour créer un job MapReduce, utilisez l'instance de `LoqateAddressingFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `LoqateAddressingDetail` comme argument.
   La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `LoqateAddressingFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Validate Address Loqate Spark

1. Créez une instance de `LoqateAddressingFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Validate Address Loqate en créant une instance de `LoqateAddressingDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40. Pour ce faire, les étapes sont les suivantes :
   a) Configurez les paramètres d'initialisation JVM en créant une instance de `LoqateAddressingGeneralConfiguration`.
      Utilisez l'énumération **Énumération ExhaustedAction** à la page 200.
   b) Configurez les paramètres de base de données nécessaires en créant une instance de `LoqateAddressingEngineConfiguration` et définissez les différents champs.
   c) Configurez les paramètres de validation d'adresse en créant une instance de `LoqateAddressingValidateConfiguration`.
      Pour définir les valeurs des différents champs de cette instance, utilisez les énumérations **Énumération AcceptanceLevel** à la page 200, **Énumération CountryCodes** à la page 198, **Énumération OutputCasing** à la page 201, **Énumération CountryFormat** à la page 201 et **Énumération ScriptAlphabet** à la page 201.
   d) Définissez les détails du chemin d'accès aux données de référence en créant une instance de `LocalReferenceDataPath`.
   e) Créez une instance de `LoqateAddressingDetail` en transmettant une instance de type `JobConfig`, l'instance `LocalReferenceDataPath` et l'instance `LoqateAddressingValidateConfiguration` créée ci-dessus comme arguments à son constructeur.
      Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.
      1. Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `LoqateAddressingDetail`.
         Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC,

créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

2. Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `LoqateAddressingDetail`.

   Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

3. Définissez le nom du job à l'aide du champ `jobName` de l'instance `LoqateAddressingDetail`.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `LoqateAddressingFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `LoqateAddressingDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

Jobs du module Universal Name

Module commun API

UniversalNameDetail< Type de processus T extends >

Objectif
Spécifier les détails d'un job du Module Universal Name.

UniversalNameFactory

Objectif
Une classe usine singleton pour créer des instances de jobs du module Universal Name.

Open Name Parser

Entités API

OpenNameParserDetail

Objectif
Spécifier les détails d'un job Open Name Parser.

OpenNameParserConfiguration

Objectif
Décomposer les noms de personnes et d'entreprises et d'autres termes du champ de données `name` en leurs éléments de composant.

Paramètres d'entrée

Paramètre	Description
Configuration d'Open Name Parser	Décomposer les noms de personnes et d'entreprises et d'autres termes du champ de données `name` en leurs éléments de composant.
Reference Data Path	Spécifier les détails du chemin d'accès aux données de référence.
Configurations des jobs	Configurations Hadoop du job. Pour un job MapReduce, l'instance doit être de type **MRJobConfig** à la page 39. Pour un job Spark, l'instance doit être de type **SparkJobConfig** à la page 39.

Paramètre	Description
Fichier d'entrée	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte d'entrée sur la plate-forme Hadoop. **Record Separator** Séparateur d'enregistrements utilisé dans le fichier d'entrée. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier d'entrée. **Qualificateur de texte** Le caractère utilisé pour entourer les valeurs de texte dans un fichier délimité. **Header Row Fields** Série de champs d'en-tête du fichier d'entrée. **Skip First Row** Indicateur spécifiant si la première ligne doit être ignorée lors de la lecture des enregistrements du fichier d'entrée. Cette option doit être définie sur `true` au cas où la première ligne est une ligne d'en-tête. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC d'entrée sur la plate-forme Hadoop. *Paramètres communs :* **Rapprochements de champs** Carte de paires clé/valeur, avec les noms de colonne existante comme clés et les noms de colonne de sortie souhaitée comme valeurs.

Paramètre	Description
Fichier de sortie	*Pour les fichiers texte :* **Chemin d'accès au fichier** Chemin d'accès au fichier texte de sortie sur la plate-forme Hadoop. **Field Separator** Séparateur utilisé entre deux champs consécutifs d'un enregistrement dans le fichier de sortie. **Avertissement :** Appelez le constructeur approprié de`FilePath`. *Pour les fichiers de format ORC :* **Chemin d'accès au fichier ORC** Chemin d'accès au fichier de format ORC de sortie sur la plate-forme Hadoop. *Paramètres communs :* **Écraser** Indicateur spécifiant si le fichier de sortie doit écraser tout fichier existant du même nom. **Create Output Header** Indicateur spécifiant si le fichier d'en-tête doit être créé ou non sur le serveur Hadoop.
Nom du job	Nom du job.

Colonnes de sortie

En plus des colonnes d'entrée, les colonnes suivantes sont ajoutées lors de la génération de la sortie d'un job Open Name Parser :

	Format	Description
AccountDescription	Chaîne	Description de compte qui fait partie du nom. Par exemple, dans « Mary Jones Account # 12345 », la description de compte est « Account#12345 ».
Champs liés aux noms de sociétés		

	Format	Description
FirmConjunction	Chaîne	Indique que le nom d'une société contient une conjonction comme « d/b/a » (doing business as), « o/a » (operating as) ou « t/a » (trading as).
FirmName	Chaîne	Le nom d'une société. Par exemple, « Pitney Bowes ».
FirmSuffix	Chaîne	Suffixe d'entreprise. Par exemple, « Co. » et « Inc. »
IsFirm	Chaîne	Indique que le nom est celui d'une entreprise et non d'un individu. Les valeurs valides sont true ou false.
Champs liés aux noms des personnes individuelles		
Conjunction	Chaîne	Indique que le nom contient une conjonction comme « and », « or » ou « & ».
CultureCode	Chaîne	Les codes de culture contenus dans les données d'entrée.
CultureCodeUsedToParse	Chaîne	Identifie la grammaire propre à une culture utilisée pour analyser les données. **Null (empty)** Culture mondiale (par défaut) **de** Allemand **es** Espagnol **ja** Japonais
FirstName	Chaîne	Prénom d'une personne.
GeneralSuffix	Chaîne	Suffixe général/professionnel d'une personne. Par exemple, MD ou PhD.

	Format	Description
IsParsed	Chaîne	Indique si un enregistrement de sortie a été analysé ou non. Les valeurs valides sont true ou false.
IsPersonal	Chaîne	Indique si le nom est celui d'un individu et non d'une société. Les valeurs valides sont true ou false.
IsReverseOrder	Chaîne	Indique si le nom d'entrée est dans l'ordre inverse ou non. Les valeurs valides sont true ou false.
LastName	Chaîne	Nom de famille d'une personne. Inclut le nom de famille paternel.
LeadingData	Chaîne	Informations autres que le nom qui apparaissent avant un nom.
MaturitySuffix	Chaîne	Suffixe de maturité/générationnel d'une personne. Par exemple, Jr. ou Sr.
MiddleName	Chaîne	Deuxième prénom d'une personne.
Name.	Chaîne	Nom de personne ou d'entreprise fourni en entrée.
NameScore	Chaîne	Indique le score moyen des jetons connus et inconnus pour chaque nom. La valeur de NameScore est comprise entre 0 et 100, tel que défini dans la grammaire d'analyse. 0 signifie qu'aucun résultat n'a été trouvé.
SecondaryLastName	Chaîne	Dans la grammaire d'analyse espagnole, nom de famille de la mère d'une personne.
TitleOfRespect	Chaîne	Informations qui apparaissent avant un nom, comme « Mr. », « Mrs. » ou « Dr. ».
TrailingData	Chaîne	Informations autres que le nom qui apparaissent après un nom.

	Format	Description
Champs liés aux noms conjoints		
Conjunction2	Chaîne	Indique qu'un deuxième nom lié contient une conjonction comme « and », « or » ou « & ».
Conjunction3	Chaîne	Indique qu'un troisième nom lié contient une conjonction comme « and », « or » ou « & ».
FirmName2	Chaîne	Nom d'une deuxième entreprise liée. Par exemple : « Baltimore Gas & Electric dba Constellation Energy ».
FirmSuffix2	Chaîne	Suffixe d'une deuxième entreprise liée.
FirstName2	Chaîne	Prénom d'un deuxième nom lié.
FirstName3	Chaîne	Prénom d'un troisième nom lié.
GeneralSuffix2	Chaîne	Suffixe général/professionnel d'un deuxième nom lié. Par exemple, MD ou PhD.
GeneralSuffix3	Chaîne	Suffixe général/professionnel d'un troisième nom lié. Par exemple, MD ou PhD.
IsConjoined	Chaîne	Indique que le nom d'entrée est lié. Exemple de nom conjoint : « John and Jane Smith ». Les valeurs sont true ou false.
LastName2	Chaîne	Nom de famille d'un deuxième nom lié.
LastName3	Chaîne	Nom de famille d'un troisième nom lié.

	Format	Description
MaturitySuffix2	Chaîne	Suffixe de maturité/générationnel d'un deuxième nom lié. Par exemple, Jr. ou Sr.
MaturitySuffix3	Chaîne	Suffixe de maturité/générationnel d'un troisième nom lié. Par exemple, Jr. ou Sr.
MiddleName2	Chaîne	Deuxième prénom d'un deuxième nom lié.
MiddleName3	Chaîne	Deuxième prénom d'un troisième nom lié.
TitleOfRespect2	Chaîne	Informations qui apparaissent avant un deuxième nom lié, comme « Mr. », « Mrs. » ou « Dr. ».
TitleOfRespect3	Chaîne	Informations qui apparaissent avant un troisième nom lié, comme « Mr. », « Mrs. » ou « Dr. ».

Utilisation d'un job Open Name Parser MapReduce

1. Créez une instance de `UniversalNameFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Open Name Parser en créant une instance de `OpenNameParserDetail` définissant `ProcessType`. L'instance doit utiliser le type **MRProcessType** à la page 40.
   a) Configurez les règles Open Name Parser en créant une instance de `OpenNameParserConfiguration`.
   b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.
   c) Créez une instance de `OpenNameParserDetail` en transmettant une instance de type `JobConfig` et les instances `OpenNameParserConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.

      Le paramètre `JobConfig` doit être une instance de type **MRJobConfig** à la page 39.
   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `OpenNameParserDetail`.

Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `OpenNameParserDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `OpenNameParserDetail`.

3. Pour créer un job MapReduce, utilisez l'instance de `UniversalNameFactory` précédemment créée pour appeler sa méthode `createJob()`. Dans ce cas, transmettez l'instance ci-dessus de `OpenNameParserDetail` comme argument.
   La méthode `createJob()` renvoie une `List` d'instances de `ControlledJob`.
4. Exécutez le job créé à l'aide d'une instance de `JobControl`.
5. Pour afficher les compteurs de reporting suite à l'exécution correcte d'un job MapReduce, utilisez l'instance précédemment créée `UniversalNameFactory` pour appeler sa méthode `getCounters()`, en transmettant le job créé comme argument.

Utilisation d'un job Open Name Parser Spark

1. Créez une instance de `UniversalNameFactory` à l'aide de sa méthode statique `getInstance()`.
2. Fournissez les détails d'entrée et de sortie du job Open Name Parser en créant une instance de `OpenNameParserDetail` définissant `ProcessType`. L'instance doit utiliser le type **SparkProcessType** à la page 40.
   a) Configurez les règles Open Name Parser en créant une instance de `OpenNameParserConfiguration`.
   b) Définissez les détails du type d'emplacement et du chemin d'accès des données de référence en créant une instance de `ReferenceDataPath`. Reportez-vous à la section **Énumération ReferenceDataPathLocation** à la page 195.
   c) Créez une instance de `OpenNameParserDetail` en transmettant une instance de type `JobConfig` et les instances `OpenNameParserConfiguration` et `ReferenceDataPath` créées précédemment comme arguments à son constructeur.

   Le paramètre `JobConfig` doit être une instance de type **SparkJobConfig** à la page 39.

   d) Définissez les détails du fichier d'entrée à l'aide du champ `inputPath` de l'instance `OpenNameParserDetail`.

   Pour un fichier d'entrée texte, créez une instance de `FilePath` avec les détails pertinents du fichier d'entrée en appelant le constructeur approprié. Pour un fichier d'entrée ORC, créez

une instance de `OrcFilePath` avec le chemin d'accès au fichier d'entrée ORC comme argument.

e) Définissez les détails du fichier de sortie à l'aide du champ `outputPath` de l'instance `OpenNameParserDetail`.

Pour un fichier de sortie texte, créez une instance de `FilePath` avec les détails pertinents du fichier de sortie en appelant le constructeur approprié. Pour un fichier de sortie ORC, créez une instance de `OrcFilePath` avec le chemin d'accès au fichier de sortie ORC comme argument.

f) Définissez le nom du job à l'aide du champ `jobName` de l'instance `OpenNameParserDetail`.

3. Pour créer et exécuter le job Spark, utilisez l'instance de `UniversalNameFactory` précédemment créée pour appeler sa méthode `runSparkJob()`. Dans ce cas, transmettez l'instance ci-dessus de `OpenNameParserDetail` comme argument.

   La méthode `runSparkJob()` exécute le job et renvoie une `Map` des compteurs de reporting du job.

4. Affichez les compteurs pour voir les statistiques de reporting du job.

5 - Fonctions Hive définies par l'utilisateur

In this section

Introduction

Apache Hive fournit des fonctions définies par l'utilisateur (UDF). Une UDF peut être définie pour réaliser les actions requises et atteindre les objectifs souhaités.

SDK qualité des Big Data fournit un ensemble de fonctions définies par l'utilisateur et de fonctions d'agrégation définies par l'utilisateur Hive permettant d'exécuter les jobs Data Quality répertoriés.

Fonctions définies par l'utilisateur (UDF)
Une fonction définie par l'utilisateur traite un enregistrement à la fois.

Les jobs de type UDF sont les suivants :

- Match Key Generator
- Table Lookup
- Advanced Transformer
- Open Name Parser

Fonctions d'agrégation définies par l'utilisateur (UDAF)
Une fonction d'agrégation définie par l'utilisateur commence par regrouper les enregistrements dans des collections en fonction du champ de jointure, puis elle traite une collection d'enregistrements à la fois.

Les jobs de type UDAF sont les suivants :

- Interflow Match
- Intraflow Match
- Transactional Match
- Best of Breed
- Duplicate Synchronization
- Filtrer
- Validate Address
- Validate Address Global
- Validate Address Loqate

Composants d'une fonction Hive **SDK Qualité des Big Data**

Les composants clés nécessaires pour exécuter un UDF Hive SDK qualité des Big Data sont :

Fichier JAR	Le fichier JAR Hive SDK qualité des Big Data du module auquel appartient l'UDF Hive Data Quality souhaité. Il doit être enregistré avant de pouvoir utiliser toute UDF.

UDF/UDAF de job	Chaque job Data Quality est fourni sous forme de fonction définie par l'utilisateur (UDF) ou de fonction d'agrégation définie par l'utilisateur (UDAF).
Alias	Alias affecté à une UDF Hive. Ceci est facultatif.
Configurations	Règles spécifiées au format JSON et autres détails de configuration, suivant le job à exécuter.
En-tête	Champs d'en-tête de la table d'entrée au format séparé par des virgules.
Table d'entrée	Table qui fournit les enregistrements d'entrée respectifs pour l'UDF Hive à exécuter.
Table de candidats	Table qui fournit les enregistrements candidats pour l'UDF Hive à exécuter, en cas d'UDAF Interflow Match.
Table de suspects	Table qui fournit les enregistrements suspects pour l'UDF Hive à exécuter, en cas d'UDAF Interflow Match.
Hive.Map.Aggr	Pour activer ou désactiver l'agrégation de données entre Mapper et Reducer, définissez cette variable d'environnement Hive sur `false`. Par défaut, `Hive.Map.Aggr = true` et les données sont regroupés. Définissez cette valeur sur `false` pour tous les jobs Hive du SDK. **Remarque :** Cette configuration est obligatoire pour tous les UDAF.
Configurations générales	Configurations de mémoire requises pour exécuter le job. **Remarque :** Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.
Configurations d'entrée	Paramètres des données d'entrée. **Remarque :** Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.
Configurations de moteur	Définition des différentes configurations telles que les paramètres de base de données, le *chemin d'accès à l'exécution COBOL* et le *type préchargement*. **Remarque :** Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.
LD_LIBRARY_PATH	Définition de cette variable d'environnement sur les chemins d'accès aux différentes bibliothèques COBOL requises lors de l'exécution des jobs Hive. **Remarque :** Cette configuration est requise uniquement pour les UDAF Hive Validate Address.

Type de processus	Spécification du niveau de validation souhaité à utiliser dans un job Hive donné du SDK. Actuellement, seule la validation d'adresse est prise en charge. Définissez cette valeur sur `VALIDATE`. **Remarque :** Cette configuration est requise uniquement pour les UDAF Hive Validate Address et Validate Address Loqate.
Sortie	Sortie de l'UDF Hive, qui peut être affichée sur la console ou exportée vers un fichier de sortie.
Requête	Requête à exécuter l'UDF Hive requise. Pour chaque job, vous pouvez effectuer l'une des opérations suivantes à l'aide de la syntaxe de requête applicable : • Afficher la sortie du job sur la console. • Enregistrer la sortie du job dans un fichier de sortie indiqué.

Utilisation d'un UDF Hive

Pour exécuter chaque job basé sur UDF Hive, vous pouvez soit exécuter ces étapes individuellement sur votre client Hive au cours d'une seule session, soit créer un fichier HQL compilant de manière séquentielle toutes les étapes requises et l'exécuter en une fois.

1. Dans votre client Hive, connectez-vous à la base de données Hive requise.
2. Enregistrez le fichier JAR du module SDK qualité des Big Data donné auquel appartient l'UDF Hive Data Quality souhaité.
3. En cas d'UDAF Validate Address, pour définir le chemin d'accès aux bibliothèques COBOL, définissez la variable d'environnement `LD_LIBRARY_PATH` comme suit :

```
set mapreduce.admin.user.env =
LD_LIBRARY_PATH=/home/hduser/~/runtime/lib:
/home/hduser/~/runtime/bin:/home/hduser/~/server/modules/universaladdress/lib,
 ACU_RUNCBL_JNI_ONLOAD_DISABLE=1, G1RTS=/home/hduser/~/ ;
```

4. En cas d'UDAF Validate Address Global, ajoutez également le fichier *libAddressDoctor5.so*.
5. En cas d'UDAF Validate Address Loqate, ajoutez ces fichiers requis au cache distribué.
   - `loqate-core.car`
   - `LoqateVerificationLevel.csv`
   - `Loqate.csv`
   - `countryTables.csv`
   - `countryNameTables.csv`

6. Créez un alias pour l'UDF Hive du job Data Quality que vous souhaitez lancer.
   Par exemple :

```
CREATE TEMPORARY FUNCTION matchkeygenerator as
'com.pb.bdq.amm.process.hive.matchkeygenerator.MatchKeyGeneratorUDF';
```

7. Indiquez les configurations telles que la règle de rapprochement, le champ de tri, la colonne Express Match et d'autres détails pour le job et affectez-les aux propriétés de configuration ou de variable respectives.

   Remarque : La règle doit être au format JSON.

   Par exemple :

```
set rule='{"matchKeys":[{"expressMatchKey":false,
"matchKeyField":"MatchKey1",
"rules":[{"algorithm":"Soundex"," field":"businessname",
"startPosition":1, "length":0,"active":true, "sortInput":null,
"removeNoiseCharacters":false}]},
{"expressMatchKey":false, "matchKeyField":"MatchKey2",
"rules":[{"algorithm":"Koeln", "field":"businessname",
"startPosition":1, "length":0, "active":true, "sortInput":null,
 "removeNoiseCharacters":false}]}]}';
```

   Remarque : Assurez-vous d'utiliser les propriétés de configuration dans les configurations de job respectives. Par exemple, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, `pb.bdq.consolidation.sort.field` et ainsi de suite, là où cela est indiqué dans les échantillons de fichiers HQL respectifs.

8. Spécifiez les champs d'en-tête de la table d'entrée au format séparé par des virgules et assignez-les à une propriété de configuration ou de variable.

```
set pb.bdq.match.header='businessname,recordid';
```

   Remarque : Veillez à utiliser la propriété de configuration, le cas échéant. Par exemple, `pb.bdq.match.header`, `pb.bdq.consolidation.header` et ainsi de suite, là où cela est indiqué dans les échantillons de fichiers HQL respectifs.

9. Désactivez l'agrégation de données entre Reducer et Mapper en définissant la configuration de la variable d'environnement `Hive.Map.Aggr` sur `false`, comme indiqué dans l'exemple ci-dessous :

```
set hive.map.aggr = false;
```

   Remarque : Cette configuration est obligatoire pour tous les UDAF.

10. Définissez les configurations générales pour l'exécution du job, comme indiqué dans l'exemple ci-dessous :

```
set pb.bdq.uam.universaladdress.general.configuration =
{"dFileType":"SPLIT", "dMemoryModel":"MEDIUM",
"lacsLinkMemoryModel":"MEDIUM", "suiteLinkMemoryModel":"MEDIUM"};
```

Remarque : Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.

11. Définissez les configurations d'entrée pour l'exécution du job, comme indiqué dans l'exemple ci-dessous :

```
set pb.bdq.uam.universaladdress.input.configuration =
{"outputStandardAddress":true, "outputPostalData":false,
"outputParsedInput":false, "outputAddressBlocks":true,
"performUSProcessing":true, "performCanadianProcessing":false,
"performInternationalProcessing":false, "outputFormattedOnFail":false,
 "outputCasing":"MIXED", "outputPostalCodeSeparator":true,
"outputMultinationalCharacters":false, "performDPV":false,
"performRDI":false, "performESM":false, "performASM":false,
"performEWS":false, "performLACSLink":false, "performLOT":false,
"failOnCMRAMatch":false, "extractFirm":false, "extractUrb":false,
"outputReport3553":false, "outputReportSERP":false,
"outputReportSummary":true, "outputCASSDetail":false,
"outputFieldLevelReturnCodes":false, "keepMultimatch":false,
"maximumResults":10,
"standardAddressFormat":"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT",
"standardAddressPMBLine":"STANDARD_ADDRESS_PMB_LINE_NONE",
"cityNameFormat":"CITY_FORMAT_STANDARD", "vanityCityFormatLong":true,
 "outputCountryFormat":"ENGLISH", "homeCountry":"United States",
"streetMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"firmMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"directionalMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"dualAddressLogic":"DUAL_NORMAL", "dpvSuccessfulStatusCondition":"A",
 "reportListFileName":"", "reportlistProcessorName":"",
"reportlistNumber":1, "reportMailerAddress":"", "reportMailerName":"",
 "reportMailerCityLine":"", "canReportMailerCPCNumber":"",
"canReportMailerAddress":"", "canReportMailerName":"",
"canReportMailerCityLine":"", "internationalCityStreetSearching":100,
 "addressLineSearchOnFail":true, "outputStreetAlias":true,
"outputVeriMoveBlock":false, "dpvDetermineNoStat":false,
"dpvDetermineVacancy":false, "outputAbbreviatedAlias":false,
"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4",
"performSuiteLink":false, "suppressZplusPhantomCarrierR777":false,
"canStandardAddressFormat":"D", "canEnglishApartmentLabel":"APT",
"canFrenchApartmentLabel":"APP", "canFrenchFormat":"C",
"canOutputCityFormat":"D", "canOutputCityAlias":true,
"canDualAddressLogic":"D", "canPreferHouseNum":false,
"canSSLVRFLG":false, "canRuralRouteFormat":"A", "canNonCivicFormat":"A",
```

```
 "canDeliveryOfficeFormat":"I", "canEnableSERP":false,
"canSwitchManagedPostalCodeConfidence":false, "stats":null,
"counts":null, "z3seg":null, "serpStats":null, "dpvSeedList":null,
"lacsSeedList":null, "zipInputSet":null, "reportName":null,
"currentUser":null, "jobName":null, "jobId":null, "jobRequest":false,
 "properties":{"DPVDetermineVacancy":"N", "DualAddressLogic":"N",
"ExtractUrb":"N", "CanFrenchFormat":"C", "AddressLineSearchOnFail":"Y",
 "OutputFieldLevelReturnCodes":"N", "OutputFormattedOnFail":"N",
"OutputStreetNameAlias":"Y", "OutputReportSERP":"N",
"OutputAddressBlocks":"Y", "ExtractFirm":"N",
"CanEnglishApartmentLabel":"APT", "OutputPreferredCity":"Z",
"FirmMatchingStrictness":"M", "CanFrenchApartmentLabel":"APP",
"KeepMultimatch":"N", "StandardAddressPMBLine":"N",
"PerformSuiteLink":"N", "CanStandardAddressFormat":"D",
"DPVSuccessfulStatusCondition":"A", "PerformLACSLink":"N",
"PerformUSProcessing":"Y", "PerformEWS":"N",
"StandardAddressFormat":"C", "SuppressZplusPhantomCarrierR777":"N",
"HomeCountry":"United States", "ReportMailerAddress":"",
"OutputReport3553":"N", "OutputVeriMoveDataBlock":"N",
"CanDeliveryOfficeFormat":"I", "OutputAbbreviatedAlias":"N",
"PerformCanadianProcessing":"N", "PerformDPV":"N",
"PerformInternationalProcessing":"N", "CanSSLVRFlg":"N",
"StreetMatchingStrictness":"M",
"InternationalCityStreetSearching":"100",
"canSwitchManagedPostalCodeConfidence":"N", "CanDualAddressLogic":"D",
 "PerformASM":"N", "OutputCasing":"M", "ReportListFileName":"",
"CanReportMailerAddress":"", "ReportMailerCityLine":"",
"CanReportMailerCPCNumber":"", "ReportListProcessorName":"",
"CanOutputCityAlias":"Y", "DirectionalMatchingStrictness":"M",
"CanRuralRouteFormat":"A", "CanOutputCityFormat":"D",
"ReportListNumber":"1", "CanReportMailerCityLine":"",
"OutputMultinationalCharacters":"N", "EnableSERP":"N",
"CanNonCivicFormat":"A", "OutputShortCityName":"S",
"OutputPostalCodeSeparator":"Y", "FailOnCMRAMatch":"N",
"PerformLOT":"N", "OutputCountryFormat":"E", "CanPreferHouseNum":"N",
 "CanReportMailerName":"", "PerformRDI":"N", "ReportMailerName":"",
"PerformESM":"N", "OutputReportSummary":"Y",
"OutputVanityCityFormatLong":"Y", "OutputPreferredAlias":"N",
"DPVDetermineNoStat":"N", "MaximumResults":"10"}}};
```

Remarque : Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.

12. Définissez les configurations de moteur pour l'exécution du job, comme indiqué dans l'exemple ci-dessous :

```
set pb.bdq.uam.universaladdress.engine.configurations = {
"referenceData":{
"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/tapan/addressquality/",
```

```
"modulesDir":"/home/hduser/tapan/addressquality/modules",
"dpvDbPath":null, "suiteLinkDBPath":null, "ewsDBPath":null,
"rdiDBPath":null, "lacsDBPath":null};
```

Remarque : Cette configuration est requise uniquement pour les UDAF Hive du module Universal Addressing.

13. Définissez le type de processus pour indiquer le niveau de validation souhaité. Actuellement, nous prenons en charge uniquement la validation d'adresse.

    Par exemple, dans le job *Validate Address*, définissez le *type de processus* comme suit :

```
set pb.bdq.uam.universaladdress.process.type=VALIDATE;
```

Remarque : Cette configuration est requise uniquement pour les UDAF Hive Validate Address et Validate Address Loqate.

14. Pour exécuter le job et afficher la sortie du job sur la console, écrivez la requête, comme indiqué dans l'exemple ci-dessous :

```
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;
```

Pour exécuter le job et placer la sortie du job dans un fichier donné, écrivez la requête, comme indiqué dans l'exemple ci-dessous :

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/MatchKey/' row format
delimited FIELDS TERMINATED BY ',' MAP FIELDS TERMINATED BY ':'
COLLECTION ITEMS TERMINATED BY '|' LINES TERMINATED BY '\n' STORED AS
 TEXTFILE
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;
```

Remarque : Veillez à utiliser l'alias précédemment définie pour l'UDF.

Important : Pour tous les jobs UDAF, utilisez les propriétés de configuration respectives sous forme de variables lors de la définition des paramètres d'entrée, là où cela est indiqué dans les exemples de fichier HQL respectifs.

Par exemple, `pb.bdq.match.rule`, `pb.bdq.match.express.column`, `pb.bdq.consolidation.sort.field` et ainsi de suite.

Fonctionnalités du module Advanced Matching

Match Key Generator

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION matchkeygenerator as
'com.pb.bdq.amm.process.hive.matchkeygenerator.MatchKeyGeneratorUDF';

-- Match Key Generator is implemented as a UDF (User Defined function).
 It processes one row at a time and generates a map of match keys for
each row.

-- Set rule and header
set rule='{"matchKeys":[{"expressMatchKey":false,
"matchKeyField":"MatchKey1",
"rules":[{"algorithm":"Soundex"," field":"businessname",
"startPosition":1, "length":0,"active":true, "sortInput":null,
"removeNoiseCharacters":false}]},
{"expressMatchKey":false, "matchKeyField":"MatchKey2",
"rules":[{"algorithm":"Koeln", "field":"businessname", "startPosition":1,
 "length":0, "active":true, "sortInput":null,
"removeNoiseCharacters":false}]}]}';

set header='businessname,recordid';

-- Execute query on the desired table to display the job output on
console. This query returns a map of key value for each row containing
 matchkeys as per rule passed.
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;


-- Query to dump output to a directory in file system
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/MatchKey/' row format
delimited FIELDS TERMINATED BY ',' MAP FIELDS TERMINATED BY ':'
COLLECTION ITEMS TERMINATED BY '|' LINES TERMINATED BY '\n' STORED AS
```

```
TEXTFILE
SELECT businessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
businessname, recordid) AS ret FROM cust ) bar;

--Sample data in input table customer
--+-----------------------------------------+----------------+--+
--|           cust.businessname             | cust.recordid  |
--+-----------------------------------------+----------------+--+
--| Internal Revenue Service                | 0              |
--| Juan F Vera-Monroig                     | 1              |
--| Leonardo Pagan-Reyes                    | 2              |
--| Academia San Joaquin Colegios/Academias | 3              |
--| Nereida Portalatin-Padua                | 4              |
--+-----------------------------------------+----------------+--+


--Sample output for input query
+-----------------------------------------+----------+-----------+-----------------+--+
|               businessname              | recordid | matchkey1 |
    matchkey2     |
+-----------------------------------------+----------+-----------+-----------------+--+
| Internal Revenue Service                | 0        | I536      |
0627657368738     |
| Juan F Vera-Monroig                     | 1        | J511      |
063376674         |
| Leonardo Pagan-Reyes                    | 2        | L563      |
567214678         |
| Academia San Joaquin Colegios/Academias | 3        | A235      |
0426864645484268  |
| Nereida Portalatin-Padua                | 4        | N631      |
67217252612       |
+-----------------------------------------+----------+-----------+-----------------+--+
```

Interflow Match

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';
```

```
-- This rowid is needed by Interflow Match to maintain the order of rows
 while creating groups. This is a UDF (User Defined Function) and
associates an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION InterMatch as
'com.pb.bdq.amm.process.hive.interflow.InterMatchUDAF';

-- Inter Flow is implemented as a UDAF (User Defined Aggregation
function). It processes one group of rows at a time based on join field
 and generates the result for that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.match.rule'

set pb.bdq.match.rule={"type":"Parent",
"missingDataMethod":"IgnoreBlanks", "threshold":100.0, "weight":0,
"children":[{"type":"Child", "missingDataMethod":"IgnoreBlanks",
"threshold":80.0, "weight":0, "matchWhenNotTrue":false,
"scoringMethod":"Maximum",
"algorithms":[{"name":"EditDistance", "weight":0, "options":null},
{"name":"Metaphone", "weight":0, "options":null}],
"crossMatchField":[], "suspectField":"firstname", "candidateField":null},
{"type":"Child", "missingDataMethod":"IgnoreBlanks", "threshold":80.0,
 "weight":0,
"matchWhenNotTrue":false, "scoringMethod":"Maximum",
"algorithms":[{"name":"KeyboardDistance", "weight":0, "options":null},
{"name":"Metaphone3", "weight":0, "options":null}], "crossMatchField":[],
"suspectField":"lastname", "candidateField":null}],
"scoringMethod":"Average", "matchingMethod":"AllTrue", "name":"NameData",
 "matchWhenNotTrue":false};


-- Set the header for suspect table using configuration property
'pb.bdq.suspect.header'
set
pb.bdq.match.suspect.header=name,firstname,lastname,matchkey,middlename,recordid;

-- Set the header for candidate table using configuration property
'pb.bdq.candidate.header'
set
pb.bdq.match.candidate.header=name,firstname,lastname,matchkey,middlename,recordid;

-- Set the sorting field to the candidates unique id's alias used in
the query. This is not from the input data.
set pb.bdq.match.sort.field=c_id;

-- Set the express match column(optional)
set pb.bdq.match.express.column=matchkey;

-- Set sort field name to the alias used in the query, using
configuration property 'pb.bdq.match.inter.comparison'
```

```
set pb.bdq.match.inter.comparison=maxNumOfDuplicates,2;

-- Optionally, one can also set
'pb.bdq.match.inter.comparison=returnUniqueCandidates,true';

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.collectnumber.zero'
set pb.bdq.match.unique.collectnumber.zero=false;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
 reading.

SELECT lateralview.record ["MatchRecordType"],
 lateralview.record ["MatchScore"],
 lateralview.record ["HasDuplicate"],
 lateralview.record ["CollectionNumber"],
 coalesce(lateralview.record ["ExpressMatched"], ''),
 lateralview.record ["SourceType"],
 lateralview.record ["name"],
 lateralview.record ["firstname"],
 lateralview.record ["lastname"],
 lateralview.record ["matchkey"],
 lateralview.record ["middlename"],
 lateralview.record ["recordid"]
FROM (
 SELECT interMatch(s_id, s_name, s_firstname, s_lastname, s_matchkey,
s_middlename, s_recordid, c_id,c_name, c_firstname, c_lastname,
c_matchkey, c_middlename, c_recordid) AS
 OUTPUT
 FROM (
  SELECT suspects.suspect_id AS s_id,
   suspects.NAME AS s_name,
   suspects.firstname AS s_firstname,
   suspects.lastname AS s_lastname,
   suspects.matchkey AS s_matchkey,
   suspects.middlename AS s_middlename,
   suspects.recordid AS s_recordid,
   candidates.candidate_id AS c_id,
   candidates.NAME AS c_name,
   candidates.firstname AS c_firstname,
   candidates.lastname AS c_lastname,
   candidates.matchkey AS c_matchkey,
   candidates.middlename AS c_middlename,
   candidates.recordid AS c_recordid
  FROM

     (
   SELECT rowid(*) AS suspect_id
    ,*
   FROM namedataintersuspect
   ) AS suspects LEFT JOIN
   (
```

```
    SELECT rowid(*) AS candidate_id
    ,*
    FROM namedataintercandidate
   ) AS candidates
   on suspects.matchkey = candidates.matchkey

  ) AS joinrecords
 GROUP BY joinrecords.s_matchkey
 ) AS innerResult LATERAL VIEW explode(innerResult.OUTPUT) lateralview
 AS record;



-- Query to dump data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/intermatch/output'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
collection items terminated by '||' map keys terminated by ':'
SELECT lateralview.record ["MatchRecordType"],
 lateralview.record ["MatchScore"],
 lateralview.record ["HasDuplicate"],
 lateralview.record ["CollectionNumber"],
 coalesce(lateralview.record ["ExpressMatched"], ''),
 lateralview.record ["SourceType"],
 lateralview.record ["name"],
 lateralview.record ["firstname"],
 lateralview.record ["lastname"],
 lateralview.record ["matchkey"],
 lateralview.record ["middlename"],
 lateralview.record ["recordid"]
FROM (
 SELECT interMatch(s_id, s_name, s_firstname, s_lastname, s_matchkey,
s_middlename, s_recordid, c_id,c_name, c_firstname, c_lastname,
c_matchkey, c_middlename, c_recordid) AS
 OUTPUT
 FROM (
  SELECT suspects.suspect_id AS s_id,
   suspects.NAME AS s_name,
   suspects.firstname AS s_firstname,
   suspects.lastname AS s_lastname,
   suspects.matchkey AS s_matchkey,
   suspects.middlename AS s_middlename,
   suspects.recordid AS s_recordid,
   candidates.candidate_id AS c_id,
   candidates.NAME AS c_name,
   candidates.firstname AS c_firstname,
   candidates.lastname AS c_lastname,
   candidates.matchkey AS c_matchkey,
   candidates.middlename AS c_middlename,
   candidates.recordid AS c_recordid
  FROM
```

```
    (
   SELECT rowid(*) AS suspect_id
    ,*
   FROM namedataintersuspect
   ) AS suspects LEFT JOIN
   (
    SELECT rowid(*) AS candidate_id
    ,*
    FROM namedataintercandidate
   ) AS candidates
   on suspects.matchkey = candidates.matchkey

  ) AS joinrecords
 GROUP BY joinrecords.s_matchkey
 ) AS innerResult LATERAL VIEW explode(innerResult.OUTPUT) lateralview
 AS record;


-- Sample input Suspect data

--+----------------+---------+---------------+------------+-----------+----------+
--| name             | firstname| lastname        | matchkey    |
middlename | recordid  |
--+----------------+---------+---------------+------------+-----------+----------+
--| LAURA  ABADSANTOS| LAURA    | ABADSANTOS      | L           |
       | 1         |
--+----------------+---------+---------------+------------+-----------+----------+

-- Sample input candidate data

--+----------------+---------+---------------+------------+-----------+----------+
--| name             | firstname| lastname        | matchkey    |
middlename | recordid  |
--+----------------+---------+---------------+------------+-----------+----------+
--| KATHRYN E ABATE  | KATHRYN  | ABATE           | L           |  E
       | 3         |
--| ANNA  ABAYEV     | ANNA     | ABAYEV          | L           |
       | 5         |
--+----------------+---------+---------------+------------+-----------+----------+

-- Sample output data

-+--------+-----+------+--------+--------+-----+------+-----+-----+----+-----+------+
--|MatchRecordType|MatchScore|HasDuplicate|CollectionNumber|ExpressMatched|SourceType|
 name       | firstname| lastname|matchkey|middlename| recordid  |
-+--------+-----+------+--------+--------+-----+------+-----+-----+----+-----+------+
--|S              |0         |Y           |0-0-1           |
    |S          |LAURA ABADSA| LAURA    |ABADSANTO|    L   |          |
 1         |
--|D              |80        |D           |0-0-1           |N
    |C          |KATHRYN E AB| KATHRYN  |AB       |    L   |    E     |
 3         |
--|D              |90        |D           |0-0-1           |N
```

```
    |C          |ANNA ABAYEV | ANNA     |ABAYEV   |    L   |          |
 5          |
-+---------+-----+-------+---------+--------+------+------+-----+-----+------+-------+
```

Intraflow Match

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Intraflow Match to maintain the order of rows
 while creating groups. This is a UDF (User Defined Function) and
associates an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION intraMatch as
'com.pb.bdq.amm.process.hive.intraflow.IntraMatchUDAF';
-- Intra Flow is implemented as a UDAF (User Defined Aggregation
function). It processes one group of rows at a time and generates the
result for that group of rows

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.match.rule'
set pb.bdq.match.rule={"type":"Parent",
"children":[{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0,
 "weight":0,
"algorithms":[{"name":"EditDistance", "weight":0, "options":null},
{"name":"Metaphone", "weight":0, "options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
"crossMatchField":[], "suspectField":"firstname", "candidateField":null},
{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0, "weight":0,
 "algorithms":[{"name":"KeyboardDistance", "weight":0, "options":null},
{"name":"Metaphone3", "weight":0, "options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
"crossMatchField":[], "suspectField":"lastname", "candidateField":null}],
"matchingMethod":"AllTrue", "scoringMethod":"Average",
"missingDataMethod":"IgnoreBlanks", "name":"NameData",
"matchWhenNotTrue":false, "threshold":100,"weight":0};
```

```
-- Set header(along with id field alias used in query) using
configuration property 'pb.bdq.match.header'
set pb.bdq.match.header=firstname,lastname,matchkey,middlename,id;

-- Set the express match column (optional)
set pb.bdq.match.express.column=matchkey;

-- Set sort field name to the alias used in the query, using the
configuration property 'pb.bdq.match.sort.field'
set pb.bdq.match.sort.field=id;

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.collectnumber.zero'
set pb.bdq.match.unique.collectnumber.zero=false;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
 reading.
-- Intra Match returns a list of map containing <key=value> pairs. Each
 map in the list corresponds to a row in the group. The below query
explodes that list of map and fetches fields from map by keys.

SELECT innerresult.record["MatchRecordType"],
 innerresult.record["MatchScore"],
 innerresult.record["CollectionNumber"],
 innerresult.record["ExpressMatched"],
 innerresult.record["firstname"],
 innerresult.record["lastname"],
 innerresult.record["matchkey"],
 innerresult.record["middlename"]
 FROM (
  SELECT  intraMatch(
    innerRowID.firstname,
    innerRowID.lastname,
    innerRowID.matchkey,
    innerRowID.middlename,
    innerRowID.id
  ) AS matchgroup
  FROM (
   SELECT  firstname, lastname, matchkey, middlename, rowid(*)
   AS id
   FROM customer_data
   ) innerRowID
  GROUP BY matchkey
 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) innerresult AS record ;

-- Query to dump output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/IntraFlow/' ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' collection items terminated by '||'
 map keys terminated by ':'
SELECT innerresult.record["MatchRecordType"],
```

```
 innerresult.record["MatchScore"],
 innerresult.record["CollectionNumber"],
 innerresult.record["ExpressMatched"],
 innerresult.record["firstname"],
 innerresult.record["lastname"],
 innerresult.record["matchkey"],
 innerresult.record["middlename"]
 FROM (
  SELECT  intraMatch(innerRowID.firstname,
    innerRowID.lastname,
    innerRowID.matchkey,
    innerRowID.middlename,
    innerRowID.id
  ) AS matchgroup
  FROM (
    SELECT  firstname, lastname, matchkey, middlename, rowid(*)
    AS id
    FROM customer_data
   ) innerRowID
  GROUP BY matchkey
  ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) innerresult AS record ;


--sample input data
--+-------------+-------------+---------------+--------------+
--| firstname   | lastname    | middlename    | matchkey     |
--+-------------+-------------+---------------+--------------+
--| Steven      | Aaen        | LYRIC         | AAE          |
--| DEBRA       | AALMO       | BOATMAN       | AAE          |

--| MARY        | AARON       | ROLLING MEADOW| AAE          |
--+-------------+-------------+---------------+--------------+

--sample output data
--+---------+------+---------+------+-----------+-----------+----------+-------+
--| firstname | lastname|middlename  |
matchkey|MatchRecordType|CollectionNumber|ExpressMatched|MatchScore|
--+---------+------+---------+------+-----------+-----------+----------+-------+
--| Steven    | Aaen    | LYRIC      | AAE     | S         |
0-0-1  |      Y         |     0    |
--| DEBRA     | AALMO   | BOATMAN    | AAE     |    D          |
  0-0-1  |      Y         |    100   |
--| MARY      | AARON   | ROLLING MEA| AAE     |    D           |
   0-0-1  |      Y        |    100   |
```

```
—+————————+————————+————————+————————+————————————+————————————+————————————+————————+
```

Transactional Match

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Transactional Match to maintain the order of
 rows while creating groups. This is a UDF (User Defined Function) and
 associates an incremental unique integer number to each row of the
data.

CREATE TEMPORARY FUNCTION transactionalMatch as
'com.pb.bdq.amm.process.hive.transactional.TransactionalMatchUDAF';

-- Transactional Match is implemented as a UDAF (User Defined Aggregation
 function). It processes one group of rows at a time and generates the
 result for that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.match.rule'
set pb.bdq.match.rule={"type":"Parent", "children":[{"type":"Child",
"matchWhenNotTrue":false, "threshold":80.0, "weight":0,
"algorithms":[{"name":"EditDistance", "weight":0, "options":null},
{"name":"Metaphone","weight":0,"options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
"crossMatchField":[], "suspectField":"firstname", "candidateField":null},
{"type":"Child", "matchWhenNotTrue":false, "threshold":80.0, "weight":0,
 "algorithms":[{"name":"KeyboardDistance","weight":0,"options":null},
{"name":"Metaphone3","weight":0,"options":null}],
"scoringMethod":"Maximum", "missingDataMethod":"IgnoreBlanks",
"crossMatchField":[], "suspectField":"lastname", "candidateField":null}],
 "matchingMethod":"AllTrue", "scoringMethod":"Average",
"missingDataMethod":"IgnoreBlanks", "name":"NameData",
"matchWhenNotTrue":false, "threshold":100, "weight":0};

-- Set header(along with id field alias used in query) using
```

```
configuration property 'pb.bdq.match.header'
set
pb.bdq.match.header=name,firstname,lastname,matchkey,middlename,recordid,id;

-- Set sort field name to the alias used in the query, using the
configuration property 'pb.bdq.match.sort.field'
set pb.bdq.match.sort.field=id;

-- Set sort collection number option for unique records using
configuration property 'pb.bdq.match.unique.candidate.return'. The
default value is false.
set pb.bdq.match.unique.candidate.return=true;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
 reading.
-- Transactional Match returns a list of map containing <key=value>
pairs. Each map in the list corresponds to a row in the group. The below
 query explodes that list of map and fetches fields from map by keys.

SELECT tmp2.record["MatchRecordType"],
 tmp2.record["MatchScore"],
 tmp2.record["HasDuplicate"],
 tmp2.record["name"],
 tmp2.record["firstname"],
 tmp2.record["lastname"],
 tmp2.record["matchkey"],
 tmp2.record["middlename"],
 tmp2.record["recordid"]
FROM (
 SELECT  transactionalMatch(innerRowID.name, innerRowID.firstname,
innerRowID.lastname, innerRowID.matchkey, innerRowID.middlename,
innerRowID.recordid, innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT name, firstname, lastname, matchkey, middlename, recordid,
rowid(name, firstname, lastname, matchkey, middlename, recordid) AS id
 FROM customer_data
  ) innerRowID
 GROUP BY matchkey
) As innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 as record ;


-- Query to dump output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/transmatch/' ROW FORMAT
 DELIMITED FIELDS TERMINATED BY ',' collection items terminated by '||'
 map keys terminated by ':'
SELECT tmp2.record["MatchRecordType"],
 tmp2.record["MatchScore"],
 tmp2.record["HasDuplicate"],
 tmp2.record["name"],
```

```
 tmp2.record["firstname"],
 tmp2.record["lastname"],
 tmp2.record["matchkey"],
 tmp2.record["middlename"],
 tmp2.record["recordid"]
FROM (
 SELECT  transactionalMatch(innerRowID.name,
   innerRowID.firstname,
   innerRowID.lastname,
   innerRowID.matchkey,
   innerRowID.middlename,
   innerRowID.recordid,
   innerRowID.id) as matchgroup
 FROM (
  SELECT name, firstname, lastname, matchkey, middlename, recordid,
rowid(name, firstname, lastname, matchkey, middlename, recordid) AS id

  FROM customer_data
  ) innerRowID
 GROUP BY matchkey ) As innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 as record ;



--sample input data
-+----------+-------------+-------------+-------------+-------------+-+
--| name               | firstname               | lastname
 | matchkey              | middlename              | recordid
     |
-+----------+-------------+-------------+-------------+-------------+-+
--| ZORINA  ABDOOL     | ZORINA                  | ABDOOL
 | Z                     |                         | 12
     |
--| ZULFIQAR  ALI      | ZULFIQAR                | ALI
 | Z                     |                         | 116
     |
--| ZACHARY  BENNETT   | ZACHARY                 | BENNETT
 | Z                     |                         | 515
     |
--| ZOHAR  BUERGER     | ZOHAR                   | BUERGER
 | Z                     |                         | 889
     |
-+----------+-------------+-------------+-------------+-------------+-+

--sample output data
-+----------+-------------+-------+-------+-------+-----------+-------+---------+
--|name            |firstname | lastname | matchkey | middlename |
recordid  | MatchRecordType | MatchScore | HasDuplicate |
-+----------+-------+-------+-------+-------+-------+-----------+-------+---------+
--|ZORINA  ABDOOL  |ZORINA    | ABDOOL   | Z        |           | 12
       |  S     | 0  |  Y     |
--|ZULFIQAR  ALI   |ZULFIQAR  | ALI      | Z        |           | 116
       |      D          | 90  |  D    |
```

```
--|ZACHARY   BENNETT|ZACHARY    | BENNETT   | Z         |             | 515
        |       D           | 91   |   D     |
--|ZOHAR   BUERGER   |ZOHAR      | BUERGER   | Z         |             | 889
        |   D      | 91   |   D     |
--+------------+-------+---------------+-----------+---------+----------+
```

Best of Breed

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Best of Breed to maintain the order of rows
 while creating groups. This is a UDF (User Defined Function) and
associates an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION bestofbreed as
'com.pb.bdq.amm.process.hive.consolidation.bestofbreed.BestOfBreedUDAF';
-- Best of Breed is implemented as a UDAF (User Defined Aggregation
function). It processes one group of rows at a time and generates the
result for that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.consolidation.rule'

set pb.bdq.consolidation.rule={"consolidationConditions":[
{"consolidationRule":{"conditionClass":"conjoinedRule", "joinType":"AND",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"LONGEST", "fieldName":"c5", "value":null,
"valueNumeric":true, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"IS_NOT_EMPTY",
"fieldName":"c9", "value":null, "valueNumeric":false,
"valueFromField":false}]},
"actions":[{"accumulate":false, "copyFromField":true, "sourceData":"c2",
 "destinationFieldName":"c2"},
{"accumulate":false, "copyFromField":false, "sourceData":"Admin",
"destinationFieldName":"c4"}]},
{"consolidationRule":{"conditionClass":"conjoinedRule", "joinType":"AND",
```

```
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"LONGEST", "fieldName":"c5", "value":null,
"valueNumeric":true, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"IS_NOT_EMPTY",
"fieldName":"c9", "value":null, "valueNumeric":false,
"valueFromField":false}]},
"actions":[{"accumulate":false, "copyFromField":false,
"sourceData":"Changed", "destinationFieldName":"c10"},
{"accumulate":false, "copyFromField":true, "sourceData":"c5",
"destinationFieldName":"c6"},
{"accumulate":true, "copyFromField":true, "sourceData":"c10",
"destinationFieldName":"c10"}]}],
"keepOriginalRecords":true, "buildTemplateRecord":true,
"templateRules":[{"consolidationRule":{"conditionClass":"conjoinedRule",
 "joinType":"OR",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"CONTAINS", "fieldName":"c1", "value":"li",
"valueNumeric":false, "valueFromField":false},
{"conditionClass":"simpleRule", "operation":"LONGEST", "fieldName":"c5",
 "value":null, "valueNumeric":false, "valueFromField":false}]},
"actions":[]}]};

-- Set header (along with the id field alias used in the query) using
configuration property 'pb.bdq.consolidation.header'
set pb.bdq.consolidation.header=c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,id;

-- Set sort field name to the alias used in the query, using the
configuration property 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=id;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
 reading.
-- Best of Breed returns a list of map containing <key=value> pairs.
Each map in the list corresponds to a row in the group. The below query
 explodes that list of map and fetches fields from map by keys.

SELECT tmp2.record["c1"],
 tmp2.record["c2"],
 tmp2.record["c3"],
 tmp2.record["c4"],
 tmp2.record["c5"],
 tmp2.record["c6"],
 tmp2.record["c7"],
 tmp2.record["c8"],
 tmp2.record["c9"],
 tmp2.record["c10"],
 tmp2.record["CollectionRecordType"]
FROM (
 SELECT  bestofbreed(innerRowID.c1,
  innerRowID.c2,
  innerRowID.c3,
  innerRowID.c4,
```

```
  innerRowID.c5,
  innerRowID.c6,
  innerRowID.c7,
  innerRowID.c8,
  innerRowID.c9,
  innerRowID.c10,
  innerRowID.id) AS matchgroup
 FROM(
  SELECT c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, rowid(*) AS id FROM
databob
  ) innerRowID
 GROUP BY c3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;

-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/bestofbreed/' ROW FORMAT
 DELIMITED FIELDS TERMINATED BY ',' collection items terminated by '||'
 map keys terminated by ':'
SELECT tmp2.record["c1"],
 tmp2.record["c2"],
 tmp2.record["c3"],
 tmp2.record["c4"],
 tmp2.record["c5"],
 tmp2.record["c6"],
 tmp2.record["c7"],
 tmp2.record["c8"],
 tmp2.record["c9"],
 tmp2.record["c10"],
 tmp2.record["CollectionRecordType"]
FROM (
 SELECT bestofbreed(innerRowID.c1,
  innerRowID.c2,
  innerRowID.c3,
  innerRowID.c4,
  innerRowID.c5,
  innerRowID.c6,
  innerRowID.c7,
  innerRowID.c8,
  innerRowID.c9,
  innerRowID.c10,
  innerRowID.id) as matchgroup
 FROM(
  SELECT c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, rowid(*) AS id FROM
databob
  ) innerRowID
 GROUP BY c3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;


--sample input data
```

```
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
--| c1        |      c2   |      c3   |      c4   |      c5   |      c6   |
     c7   |      c8   | c9        | c10       |
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
--| Duplicate| 87        | 1         |           |ANNA ABNEY| ANNA      |
          | ABNEY     | A         | 18        |
--| Duplicate| 77        | 1         |           |ANNA A ANN| ANDREA    |
          | ANNAKAY   | A         | 196       |
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------+

--sample output data
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------------+
--| c1        | c2        | c3        | c4        | c5        | c6        |
c7        | c8        | c9        | c10       |CollectionRecordType|
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------------+
--| Duplicate| 87        | 1         |           |ANNA ABNEY| ANNA      |
          | ABNEY     | A         | 18        |   Primary           |
--| Duplicate| 77        | 1         |           |ANNA A ANN| ANDREA    |
 ARANOW   | ANNAKAY   | A         | 196       |   Secondary         |
--| Duplicate| 87        | 1         |           |ANNA ABNEY| ANNA      |
 ARANOW   | ABNEY     | A         | 18        |   BestOfBreed       |
--+-------+-------+-------+-------+-------+-------+-------+-------+-------+-------------+
```

Duplicate Synchronization

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Duplicate Synchronization to maintain the
order of rows while creating groups. This is a UDF (User Defined
Function) and associates an incremental unique integer number to each
row of the data.

CREATE TEMPORARY FUNCTION dupsync as
'com.pb.bdq.amm.process.hive.consolidation.duplicatesync.DuplicateSyncUDAF';

-- Duplicate Sync is implemented as a UDAF (User Defined Aggregation
function). It processes one group of rows at a time and generates the
result for that group of rows.
```

```
-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.consolidation.rule'

set pb.bdq.consolidation.rule={"consolidationConditions":
[{"consolidationRule":
{"conditionClass":"conjoinedRule", "joinType":"AND",
"consolidationRules":[{"conditionClass":"simpleRule",
"operation":"HIGHEST", "fieldName":"column2", "value":null,
"valueFromField":false, "valueNumeric":true}]},
"actions":[{"accumulate":false, "copyFromField":true,
"sourceData":"column5", "destinationFieldName":"column5"}]}]};

-- Set header (along with the id field alias used in the query) using
configuration property 'pb.bdq.consolidation.header'
set
pb.bdq.consolidation.header=column1,column2,column3,column4,column5,id;

-- Set sort field name to alias used in query using configuration
property 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=id;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
 reading.
-- Duplicate Sync returns a list of map containing <key=value> pairs.
Each map in the list corresponds to a row in the group. The below query
 explodes that list of map and fetches fields from map by keys.

SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  dupsync (innerRowID.column1,
  innerRowID.column2,
  innerRowID.column3,
  innerRowID.column4,
  innerRowID.column5,
  innerRowID.id
  ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM databob
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;
```

```
-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/dupsync/' ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' collection items terminated by '||'
 map keys terminated by ':'
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT dupsync( innerRowID.column1,
  innerRowID.column2,
  innerRowID.column3,
  innerRowID.column4,
  innerRowID.column5,
  innerRowID.id
  ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM databob
  ) innerRowID
 GROUP BY column3 ) AS innerResult
 LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;



--sample input data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 87       | 1        |          |ANNA ABNEY|
--| Duplicate| 77       | 1        |          |ANNA A ANN|
--| Suspect  |          | 1        |          |ANNA A ABN|
--+----------+----------+----------+----------+----------+

--sample output data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 87       | 1        |          |ANNA ABNEY|
--| Duplicate| 77       | 1        |          |ANNA A ANN|
```

```
--| Suspect  |          | 1        |          |ANNA ABNEY|
--+----------+----------+----------+----------+----------+
```

Filtre

Exemple de script Hive

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.

CREATE TEMPORARY FUNCTION rowid as
'com.pb.bdq.hive.common.RowIDGeneratorUDF';

-- This rowid is needed by Filter to maintain the order of rows while
creating groups. This is a UDF (User Defined Function) and associates
an incremental unique integer number to each row of the data.

CREATE TEMPORARY FUNCTION filter as
'com.pb.bdq.amm.process.hive.consolidation.filter.FilterUDAF';

-- Filter is implemented as a UDAF (User Defined Aggregation function).
 It processes one group of rows at a time and generates the result for
 that group of rows.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'pb.bdq.consolidation.rule'
set pb.bdq.consolidation.rule={"consolidationConditions":
[{"consolidationRule":{"conditionClass":"simpleRule",
"operation":"HIGHEST", "fieldName":"column2", "value":null,
"valueFromField":false, "valueNumeric":true}, "actions":[]}],
"removeDuplicates":true};

-- Set header (along with the id field alias used in the query) using
configuration property 'pb.bdq.consolidation.header'
set
pb.bdq.consolidation.header=column1,column2,column3,column4,column5,id;

-- Set sort field name to alias used in query using configuration
property 'pb.bdq.consolidation.sort.field'
set pb.bdq.consolidation.sort.field=id;

-- Execute Query on the desired table. The query uses a UDF rowid, which
 must be present in the query to maintain the ordering of the data while
```

```
 reading.

SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;

-- Query to dump the output to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/HiveUDF/filter/'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
collection items terminated by '||' map keys terminated by ':'
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (innerRowID.column1,
   innerRowID.column2,
   innerRowID.column3,
   innerRowID.column4,
   innerRowID.column5,
   innerRowID.id
 ) AS matchgroup
 FROM (
  SELECT column1, column2, column3, column4, column5, rowid(*)
  AS id
  FROM data
  ) innerRowID
 GROUP BY column3
 ) AS innerResult
LATERAL VIEW explode(innerResult.matchgroup) tmp2 AS record ;
```

```
--sample input data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Duplicate| 80       | 98       |          | EUNICE L |
--| Suspect  |          | 98       |          | ERIC L BR|
--+----------+----------+----------+----------+----------+

--sample output data
--+----------+----------+----------+----------+----------+
--| column1  | column2  | column3  | column4  | column5  |
--+----------+----------+----------+----------+----------+
--| Suspect  |          | 98       |          | ERIC L BR|
--+----------+----------+----------+----------+----------+
```

Fonctions du module Data Normalization

Table Lookup

Exemple de script Hive

```
-- Register Data Normalization Modue [dnm] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
-- Table Lookup is implemented as a UDF (User Defined function). Hence
 it processes one row at a time and generates a map of key value pairs
 for each row.
CREATE TEMPORARY FUNCTION tablelookup as
'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';

-- Set rule
set rule='{"rules":[{"action":"Standardize", "source":"CityCode",
"tableName":"State Name Abbreviations", "lookupMultipleWordTerms":false,
 "lookupIndividualTermsWithinField":false, "destination":"CityCode"}]}';

-- Set Reference Directory. This must be a local path on cluster machines
 and must be present on each node of the cluster at the same path.
set refdir='/home/hadoop/reference';

-- set header
set header ='AccountDescription,Address,ApartmentNumber,CityCode';
```

```
-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.

SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/TableLookup/' row format
 delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
 TEXTFILE
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;

--Sample input data
--+-------------------------+---------------+-----------------------+----------------+
--| citizen_data.accountdescription  | citizen_data.address  |
citizen_data.apartmentnumber  | citizen_data.citycode |
--+-------------------------+---------------+-----------------------+----------------+
--|                          | 400 E M0 St Apt 1405  |
            | NY                          |
--|                                       | 190 E 72nd St         |
                        | NY                            |
--|                                       | 1381 3rd Ave Apt 4    | 4
                        | TTYYY                         |
--+-------------------------+---------------+-----------------------+----------------+


--sample output data
```

```
-+----------------------+-----------------+---------------+---------------+-------+

--|StandardizationTermIdentified |  accountdescription   | address
  | apartmentnumber    | citycode|
-+----------------------+-----------------+---------------+---------------+-------+

--|    yes     |              | 400 E M0 St Apt 1405  |
       | NEW YORK |
--|    yes          |                            | 190 E 72nd St
|                       | NEW YORK |
--|    yes     |                           | 1381 3rd Ave Apt 4    | 4
                     | NEW YORK |
-+----------------------+-----------------+---------------+---------------+-------+
```

Advanced Transformer

Exemple de script Hive

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
-- Advanced Transformer is implemented as a UDF (User Defined function).
 Hence it processes one row at a time and generates a map of key value
 pairs for each row.
CREATE TEMPORARY FUNCTION advanceTransform as
'com.pb.bdq.dnm.process.hive.advancetransformer.AdvanceTransformerUDF';

-- Set rule
set rule='{"rules":[{"extractionType":"TableData", "source":"address",
 "nonExtractedData":"address_1", "extractedData":"address_2",
"tokenizationCharacters":"", "tableName":"Street Suffix Abbreviations",
 "multipleTermLookup":false, "tokenize":true, "extract":"ExtractTerm",
 "includeTermWith":"ExtractedData", "wordsToExtract":2}]}';


-- Set Reference Directory. This must be a local path on cluster machines
 and must be present on each node of the cluster at the same path.
set refdir='/home/hadoop/reference/';

-- set header
set header ='AccountDescription,Address';

-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.
```

```
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;


-- Query to dump output data to a file

INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/AdvXformer/' row format
 delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
 TEXTFILE
SELECT bar.ret["AdvancedTransformTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["address_1"]
FROM (
 SELECT advanceTransform(${hiveconf:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address)
 AS ret
 FROM advxformX
 ) bar;

--sample input data
+--------------------------------+--------------------+----------------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |
+--------------------------------+--------------------+----------------------+
| Yes                                |            | 400 E M0 St Apt 1405
 |
| Yes                                |                      | 190 E 72nd
St          |
+--------------------------------+--------------------+----------------------+

--sample output data
+-----------------------------+-----------------+-------------------+-----------------+
| AdvancedTransformTermIdentified  | accountdescription  |      address
          |     address_1      |
+-----------------------------+-----------------+-------------------+-----------------+
| Yes                                |            | 400 E M0 St Apt 1405
 | 400 E M0 Apt 1405  |
| Yes                                |                      | 190 E 72nd
St          | 190 E 72nd          |
```

```
+-----------------------------+-----------------+-----------------+-----------------+
```

Fonctions du module Universal Addressing

Validate Address

Avertissement : Avant de créer et d'exécuter le premier job Validate Address, assurez-vous que le service Acushare est en cours d'exécution. Pour connaître les étapes, reportez-vous à la section **Arrêt du service acushare** à la page 12.

Exemple de script Hive

```
-- Register Universal Addressing Module [UAM-Global] BDQ Hive UDAF Jar

ADD JAR <Directory
path>/uam.universaladdress.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION uamvalidation as
'com.pb.bdq.uam.process.hive.universaladdress.UAMUSAddressingUDAF';

-- set LD_LIBRARY_PATH(path to modules lib, runtime/lib and runtime/bin),
 G1RTS(path containing COBOL runtime) and ACU_RUNCBL_JNI_ONLOAD_DISABLE
 in this configuration
set mapreduce.admin.user.env =
LD_LIBRARY_PATH=/home/hduser/~/runtime/lib:
/home/hduser/~/runtime/bin:/home/hduser/~/server/modules/universaladdress/lib,
 ACU_RUNCBL_JNI_ONLOAD_DISABLE=1, G1RTS=/home/hduser/~/ ;

set hive.map.aggr = false;

-- set engine configuration
set pb.bdq.uam.universaladdress.engine.configurations={ "referenceData":{

"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/tapan/addressquality/",
"modulesDir":"/home/hduser/tapan/addressquality/modules",
"dpvDbPath":null, "suiteLinkDBPath":null, "ewsDBPath":null,
"rdiDBPath":null, "lacsDBPath":null};
```

```
-- set input configuration
set
pb.bdq.uam.universaladdress.input.configuration={"outputStandardAddress":true,
 "outputPostalData":false, "outputParsedInput":false,
"outputAddressBlocks":true, "performUSProcessing":true,
"performCanadianProcessing":false,
"performInternationalProcessing":false, "outputFormattedOnFail":false,
 "outputCasing":"MIXED", "outputPostalCodeSeparator":true,
"outputMultinationalCharacters":false, "performDPV":false,
"performRDI":false, "performESM":false, "performASM":false,
"performEWS":false, "performLACSLink":false, "performLOT":false,
"failOnCMRAMatch":false, "extractFirm":false, "extractUrb":false,
"outputReport3553":false, "outputReportSERP":false,
"outputReportSummary":true, "outputCASSDetail":false,
"outputFieldLevelReturnCodes":false, "keepMultimatch":false,
"maximumResults":10,
"standardAddressFormat":"STANDARD_ADDRESS_FORMAT_COMBINED_UNIT",
"standardAddressPMBLine":"STANDARD_ADDRESS_PMB_LINE_NONE",
"cityNameFormat":"CITY_FORMAT_STANDARD", "vanityCityFormatLong":true,
"outputCountryFormat":"ENGLISH", "homeCountry":"United States",
"streetMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"firmMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"directionalMatchingStrictness":"MATCHING_STRICTNESS_MEDIUM",
"dualAddressLogic":"DUAL_NORMAL", "dpvSuccessfulStatusCondition":"A",
"reportListFileName":"", "reportlistProcessorName":"",
"reportlistNumber":1, "reportMailerAddress":"", "reportMailerName":"",
 "reportMailerCityLine":"", "canReportMailerCPCNumber":"",
"canReportMailerAddress":"", "canReportMailerName":"",
"canReportMailerCityLine":"", "internationalCityStreetSearching":100,
"addressLineSearchOnFail":true, "outputStreetAlias":true,
"outputVeriMoveBlock":false, "dpvDetermineNoStat":false,
"dpvDetermineVacancy":false, "outputAbbreviatedAlias":false,
"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4",
"performSuiteLink":false, "suppressZplusPhantomCarrierR777":false,
"canStandardAddressFormat":"D", "canEnglishApartmentLabel":"APT",
"canFrenchApartmentLabel":"APP", "canFrenchFormat":"C",
"canOutputCityFormat":"D", "canOutputCityAlias":true,
"canDualAddressLogic":"D", "canPreferHouseNum":false,
"canSSLVRFLG":false, "canRuralRouteFormat":"A", "canNonCivicFormat":"A",
 "canDeliveryOfficeFormat":"I", "canEnableSERP":false,
"canSwitchManagedPostalCodeConfidence":false, "stats":null,
"counts":null, "z3seg":null, "serpStats":null, "dpvSeedList":null,
"lacsSeedList":null, "zipInputSet":null, "reportName":null,
"currentUser":null, "jobName":null, "jobId":null, "jobRequest":false,
"properties":{"DPVDetermineVacancy":"N", "DualAddressLogic":"N",
"ExtractUrb":"N", "CanFrenchFormat":"C", "AddressLineSearchOnFail":"Y",
 "OutputFieldLevelReturnCodes":"N", "OutputFormattedOnFail":"N",
"OutputStreetNameAlias":"Y", "OutputReportSERP":"N",
"OutputAddressBlocks":"Y", "ExtractFirm":"N",
"CanEnglishApartmentLabel":"APT", "OutputPreferredCity":"Z",
"FirmMatchingStrictness":"M", "CanFrenchApartmentLabel":"APP",
"KeepMultimatch":"N", "StandardAddressPMBLine":"N",
```

```
"PerformSuiteLink":"N", "CanStandardAddressFormat":"D",
"DPVSuccessfulStatusCondition":"A", "PerformLACSLink":"N",
"PerformUSProcessing":"Y", "PerformEWS":"N", "StandardAddressFormat":"C",
 "SuppressZplusPhantomCarrierR777":"N", "HomeCountry":"United States",
 "ReportMailerAddress":"", "OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N", "CanDeliveryOfficeFormat":"I",
"OutputAbbreviatedAlias":"N", "PerformCanadianProcessing":"N",
"PerformDPV":"N", "PerformInternationalProcessing":"N",
"CanSSLVRFlg":"N", "StreetMatchingStrictness":"M",
"InternationalCityStreetSearching":"100",
"canSwitchManagedPostalCodeConfidence":"N", "CanDualAddressLogic":"D",
 "PerformASM":"N", "OutputCasing":"M", "ReportListFileName":"",
"CanReportMailerAddress":"", "ReportMailerCityLine":"",
"CanReportMailerCPCNumber":"", "ReportListProcessorName":"",
"CanOutputCityAlias":"Y", "DirectionalMatchingStrictness":"M",
"CanRuralRouteFormat":"A", "CanOutputCityFormat":"D",
"ReportListNumber":"1", "CanReportMailerCityLine":"",
"OutputMultinationalCharacters":"N", "EnableSERP":"N",
"CanNonCivicFormat":"A", "OutputShortCityName":"S",
"OutputPostalCodeSeparator":"Y", "FailOnCMRAMatch":"N", "PerformLOT":"N",
 "OutputCountryFormat":"E", "CanPreferHouseNum":"N",
"CanReportMailerName":"", "PerformRDI":"N", "ReportMailerName":"",
"PerformESM":"N", "OutputReportSummary":"Y",
"OutputVanityCityFormatLong":"Y", "OutputPreferredAlias":"N",
"DPVDetermineNoStat":"N", "MaximumResults":"10"}}};

-- set general configuration
set pb.bdq.uam.universaladdress.general.configuration =
{"dFileType":"SPLIT", "dMemoryModel":"MEDIUM",
"lacsLinkMemoryModel":"MEDIUM", "suiteLinkMemoryModel":"MEDIUM"};

-- set reference path
set pb.bdq.reference.data.local.location=/media/New
Volume/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15;

-- set process type
set pb.bdq.uam.universaladdress.process.type=VALIDATE;

-- set header
set pb.bdq.header=InputKeyValue,FirmName,AddressLine1,AddressLine2,City,
StateProvince,PostalCode,Text;

-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.
SELECT tmp2.record["Confidence"], tmp2.record["AddressLine1"] FROM (
select uamvalidation(inputkeyvalue, firmname, addressline1, addressline2,
 city, stateprovince, postalcode, text) from uam_us) as addressgroup
LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/GlobalAddressing/' row
format delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED
```

```
 AS TEXTFILE
SELECT tmp2.record["Confidence"], tmp2.record["AddressLine1"] FROM (
select uamvalidation(inputkeyvalue, firmname, addressline1, addressline2,
 city, stateprovince, postalcode, text) from uam_us) as addressgroup
LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;
```

```
+------------+--------------------+----------+---------------+-------------+------------+
| address.recordid  |     address.addressline1     |  address.city   |
 address.stateprovince  | address.postalcode  | address.country  |
+------------+--------------------+----------+---------------+-------------+------------+
| 1                  | 18 Merivale St                | South Brisbane  |
 QLD                     | 4101                 | AUS              |
| 2                  | 19 Serpentine Rd              | Albany          |
 WA                      | 6330                 | AUS              |
| 3                  | 317 VICTORIA ST GR            | BRUNSWICK       |
 VIC                     | 3056                 | AUS              |
| 4                  | DUPLEX 6/16-18 O'CONNELL ST   | AINSLIE         |
 ACT                     | 2602                 | AUS              |
| 5                  | LOT 154 470 BRYGON CREEK DR   | UPPER COOMERA   |
 QLD                     | 4209                 | AUS              |
| 6                  | 16 GREENE ST                  | WARRAWONG       |
 ACT                     | 2502                 | AUS              |
| 7                  | UNIT 47/16 BLAIRMOUNT ST      | PARKINSON       |
 QLD                     | 4115                 | AUS              |
| 8                  | 13-15 FRANCESCO CRES          | BELLA VISTA     |
 NSW                     | 2153                 | AUS              |
| 9                  | 4 RYANS LANE                  | HEATHCOTE       |
 VIC                     | 3523                 | AUS              |
| 10                 | 1 CHRISTMAS LN                | NORTH POLE      |
 VIC                     | 1111                 | AUS              |
+------------+--------------------+----------+---------------+-------------+------------+
```

```
+----------+-------------+-------+-------------------------------+-------------+
|Confidence |StreetName     |HouseNumber |     AddressLine1
  | AddressType   |
+----------+-------------+----------+-------------------------+-------------+
| 100.00   | MERIVALE      | 18    | 18 MERIVALE ST                | S
      |
| 99.42    | SERPENTINE    | 19       | 19 SERPENTINE RD E           |
 S       |
| 97.95    | VICTORIA      | 317      | 317 VICTORIA ST              |
 S       |
| 100.00   | O'CONNELL     | 16-18    | DUP 6 16-18 O'CONNELL ST     |
 S       |
| 0.00     | BRYGON CREEK  | 470      | LOT 154 470 BRYGON CREEK DR  |
 U       |
| 76.99    | GREENE        | 16        | 16 GREENE ST
| S       |
| 100.00   | BLAIRMOUNT    | 16       | U 47 16 BLAIRMOUNT ST        |
 S       |
| 100.00   | FRANCESCO     | 13-15    | 13-15 FRANCESCO CRES         |
```

```
 S       |
| 100.00   | RYANS          | 4         | 4 RYANS LANE                 |
 S       |
| 0.00     | CHRISTMAS      | 1         | 1 CHRISTMAS LN               |
 U       |
+----------+--------------+-----------+----------------------------+--------------+
```

Validate Address Global

Exemple de script Hive

```
-- Register Universal Addressing Module [UAM-Global] BDQ Hive UDAF Jar

ADD JAR <Directory path>/uam.global.hive.${project.version}.jar;

ADD FILE <Directory path>/libAddressDoctor5.so;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION globalvalidation as
'com.pb.bdq.uam.process.hive.global.GlobalAddressingUDAF';

set hive.map.aggr = false;

-- set engine configuration
set pb.bdq.uam.global.engine.configurations=[{ "referenceData":
{"dataDir":"/media/New Volume/hduser/resources/uam/addressDoctor/5.8.0/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"databaseType":"BATCH_INTERACTIVE", "preloadingType":"NONE",
"allCountries":false, "supportedCountries":"CAN,USA,AUS"}];

-- set input configuration
set
pb.bdq.uam.global.input.configuration={"resultStateProvinceType":"COUNTRY_STANDARD",
 "processMatchingScope":"ALL", "processEnrichmentAMAS":false,
"inputForceCountryISO3":"AUS", "inputDefaultCountryISO3":"AUS",
"inputFormatDelimiter":"CRLF", "resultFormatDelimiter":"CRLF",
"resultIncludeInputs":false, "resultCountryType":"NAME_EN",
"processOptimizationLevel":"STANDARD",
"resultPreferredLanguage":"DATABASE", "processMode":"BATCH",
"resultPreferredScript":"DATABASE", "resultMaximumResults":1,
"resultCasing":"NATIVE",
"properties":{"Result.StateProvinceType":"COUNTRY_STANDARD",
"Process.MatchingScope":"ALL", "Process.EnrichmentAMAS":"false",
"Input.ForceCountryISO3":"AUS", "Input.FormatDelimiter":"CRLF",
"Result.FormatDelimiter":"CRLF", "Input.DefaultCountryISO3":"AUS",
"Result.IncludeInputs":"false", "Result.CountryType":"NAME_EN",
```

```
"Process.OptimizationLevel":"STANDARD",
"Result.PreferredLanguage":"DATABASE", "Process.Mode":"BATCH",
"Result.PreferredScript":"DATABASE", "Result.MaximumResults":"1",
"Result.Casing":"NATIVE", "Database.AddressGlobal":"Database"}};

-- set general configuration
set pb.bdq.uam.global.general.configuration={"cacheSize":"LARGE",
"maxThreadCount":8, "maxAddressObjectCount":8, "rangesToExpand":"NONE",
 "flexibleRangeExpansion":"ON", "enableTransactionLogging":false,
"maxMemoryUsageMB":1024};

-- set unlock codec
set pb.bdq.uam.global.unlockCode=<Insert your Unlock Code here>;

-- set header
set
pb.bdq.header=recordid,AddressLine1,City,StateProvince,PostalCode,Country;

-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.
SELECT tmp2.record["HouseNumber"], tmp2.record["Confidence"],
tmp2.record["AddressLine1"], tmp2.record["StreetName"],
tmp2.record["PostalCode"], tmp2.record["ElementInputStatus"],
tmp2.record["MailabilityScore"] FROM ( SELECT  globalvalidation(recordid,
 addressline1, city, stateprovince, postalcode, country) as mygp from
address) as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2
as record ;


-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/GlobalAddressing/' row
format delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED
 AS TEXTFILE
SELECT tmp2.record["HouseNumber"], tmp2.record["Confidence"],
tmp2.record["AddressLine1"], tmp2.record["StreetName"],
tmp2.record["PostalCode"], tmp2.record["ElementInputStatus"],
tmp2.record["MailabilityScore"] FROM ( SELECT  globalvalidation(recordid,
 addressline1, city, stateprovince, postalcode, country) as mygp from
address) as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2
as record ;

+------------+--------------------+-----------+---------------+-------------+------------+
| address.recordid  |     address.addressline1     |  address.city   |
 address.stateprovince  | address.postalcode  | address.country  |
+------------+--------------------+-----------+---------------+-------------+------------+
| 1                 | 18 Merivale St               | South Brisbane  |
 QLD                    | 4101                | AUS              |
| 2                 | 19 Serpentine Rd             | Albany          |
 WA                     | 6330                | AUS              |
| 3                 | 317 VICTORIA ST GR           | BRUNSWICK       |
 VIC                    | 3056                | AUS              |
| 4                 | DUPLEX 6/16-18 O'CONNELL ST  | AINSLIE         |
```

```
 ACT                    | 2602                 | AUS                |
| 5                   | LOT 154 470 BRYGON CREEK DR  | UPPER COOMERA    |
 QLD                    | 4209                 | AUS                |
| 6                   | 16 GREENE ST                 | WARRAWONG        |
 ACT                    | 2502                 | AUS                |
| 7                   | UNIT 47/16 BLAIRMOUNT ST     | PARKINSON        |
 QLD                    | 4115                 | AUS                |
| 8                   | 13-15 FRANCESCO CRES         | BELLA VISTA      |
 NSW                    | 2153                 | AUS                |
| 9                   | 4 RYANS LANE                 | HEATHCOTE        |
 VIC                    | 3523                 | AUS                |
| 10                  | 1 CHRISTMAS LN               | NORTH POLE       |
 VIC                    | 1111                 | AUS                |
+------------+--------------------+-----------+----------------+-------------+------------+



+----------+-------------+-------+-----------------------------+-------------+
|Confidence |StreetName     |HouseNumber |     AddressLine1
  | AddressType   |
+----------+-------------+-------+-----------------------------+-------------+
| 100.00   | MERIVALE      | 18    | 18 MERIVALE ST               | S
      |
| 99.42    | SERPENTINE    | 19       | 19 SERPENTINE RD E           |
 S       |
| 97.95    | VICTORIA      | 317      | 317 VICTORIA ST              |
 S       |
| 100.00   | O'CONNELL     | 16-18    | DUP 6 16-18 O'CONNELL ST     |
 S       |
| 0.00     | BRYGON CREEK  | 470      | LOT 154 470 BRYGON CREEK DR  |
 U       |
| 76.99    | GREENE        | 16        | 16 GREENE ST
| S       |
| 100.00   | BLAIRMOUNT    | 16       | U 47 16 BLAIRMOUNT ST        |
 S       |
| 100.00   | FRANCESCO     | 13-15    | 13-15 FRANCESCO CRES         |
 S       |
| 100.00   | RYANS         | 4        | 4 RYANS LANE                 |
 S       |
| 0.00     | CHRISTMAS     | 1        | 1 CHRISTMAS LN               |
 U       |
```

```
+----------+-------------+-----------+---------------------------+-------------+
```

Validate Address Loqate

Exemple de script Hive

```
-- Register Universal Address Module [UAM] BDQ Hive Loqate UDAF Jar
ADD JAR <Directory path>/uam.loqate.hive.${project.version}.jar;

-- Provide alias to UDAF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION loqatevalidation as
'com.pb.bdq.uam.process.hive.loqate.LoqateAddressingUDAF';

-- Adding required files to distributed cache.
ADD FILES <Directory Path>/loqate-core.car;
ADD FILES <Directory Path>/LoqateVerificationLevel.csv;
ADD FILES <Directory Path>/Loqate.csv;
ADD FILES <Directory Path>/countryTables.csv;
ADD FILES <Directory Path>/countryNameTables.csv;

set hive.map.aggr = false;

-- set process configuration
set pb.bdq.uam.loqate.process.configuration={"processType":"VALIDATE",
 "includeMatchedAddressElements":true,
"standardizedInputAddressElements":true, "returnAddressDataBlocks":true,
 "casing":"Mixed", "outputReportSummary":false,
"returnMultipleAddresses":false, "failedOnMultiMatchFound":false,
"countryFormat":"ENGLISH", "defaultCountry":"USA",
"scriptAlphabet":"Native", "returnGeocodedAddressFields":true,
"acceptanceLevel":"Level0", "minimumMatchScore":0,
"formatDataUsingAMASConventions":false,
"singleFieldDuplicateHandling":false,
"multiFieldDuplicateHandling":false,
"nonStandardFieldDuplicateHandling":false,
"outputFieldDuplicateHandling":false, "includeStandardAddress":true,
"duplicateHandling":false, "returnMultipleAddressCount":10};

-- set general configuration
set pb.bdq.uam.loqate.general.configuration={"maxIdle":null,
"minIdle":16, "maxActive":16, "maxWait":null, "whenExhaustedAction":null,
 "testOnBorrow":null, "testOnReturn":null, "testWhileIdle":null,
"timeBetweenEvictionRunsMillis":null, "numTestsPerEvictionRun":null,
"minEvictableIdleTimeMillis":null};

-- set engine configuration
```

```
set pb.bdq.uam.loqate.engine.configuration={"verbose":true,
"toolInfo":true, "outputAddressFormat":false, "logInput":false,
"logOutput":false, "logFileName":null, "matchScoreAbsoluteThreshold":60,
 "matchScoreThresholdFactor":95, "postalCodeMaxResults":10,
"strictReferenceMatch":false};

-- set reference directory path
set pb.bdq.referencedata.dir=/media/New
Volume/hduser/resources/uam/loqate/Linux;

-- set process type
set pb.bdq.uam.loqate.process.type=VALIDATE;

-- set input header
set pb.bdq.header='InputKeyValue,AddressLine1,AddressLine2,AddressLine3,
AddressLine4,City,StateProvince,PostalCode,Country,FirmName';

select SELECT tmp2.record["HouseNumber"], tmp2.record["Confidence"],
tmp2.record["AddressLine1"], tmp2.record["StreetName"],
tmp2.record["PostalCode"], tmp2.record["DPID"], tmp2.record["Barcode"]
 FROM ( SELECT  loqatevalidation(recordid, addressline1, city,
stateprovince, postalcode, country) as mygp from address) as <TABLE_NAME>
 LATERAL VIEW explode(addressgroup.mygp) tmp2 as record ;

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/loqate/' row format
delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS
TEXTFILE SELECT * FROM ( SELECT tmp2.record["HouseNumber"],
tmp2.record["Confidence"], tmp2.record["AddressLine1"],
tmp2.record["StreetName"], tmp2.record["PostalCode"],
tmp2.record["DPID"], tmp2.record["Barcode"] FROM ( SELECT
loqatevalidation(recordid, addressline1, city, stateprovince, postalcode,
 country) as mygp from address) as <TABLE_NAME> LATERAL VIEW
explode(addressgroup.mygp) tmp2 as record ;



--Sample Input
+-------------+-------------------------------+-------------+-----------+-------------+
| inputkeyvalue  |            addressline1            | stateprovince
 | postalcode  |    country    |
+-------------+-------------------------------+-------------+-----------+-------------+
| 1              | 80 Quan Su                          |
 |             | Vietnam       |
| 2              | Final Av. Panteón Foro Libertador  |
 | 1010        | Venezuela     |
| 3              | P O Box 834                         |
 |             | St Vincent    |
| 4              | Colonia 2066                        |
 |             | Uruguay       |
| 5              | Ave de la Resistance BP127          |
 |             | Burkina Faso  |
| 6              | Buyuk Turon Street, 41              |
```

```
 |             | Uzbekistan    |
| 7              | Empire State Building              | NY
 | 10118       | US            |
| 8              | 3 Leontovycha St                   |
 |             | Ukraine       |
| 9              |                                    | Ceredigion
 |             | Wales         |
| 10             | 5 Main Street                      | Ballindalloch
 |             | Scotland      |
+------------+-------------------------+------------+---------+------------+


-- Sample Output
+-----------+--------------+--------+---+-----------------------------------+
|Match Score|StreetName     |HouseNumber |            addressline1
        |
+-----------+--------------+-----------+-----------------------------------+
| 100.00    | MERIVALE      | 80     | 80 Quan Su
   |
| 100.00     | SERPENTINE    |         | Final Av. Panteón Foro Libertador
   |
| 0.00     | VICTORIA      | 0        | P O Box 834
    |
| 75.00    | O'CONNELL     | 2066     | Colonia 2066
    |
| 83.33      | BRYGON CREEK   | 470       | Ave de la Resistance BP127
       |
| 100.00     | GREENE         |          | Buyuk Turon Street, 41
      |
| 96.8254    | BLAIRMOUNT     | 41        | Empire State Building
       |
| 83.950    | FRANCESCO      | 350      | 3 Leontovycha St
     |
| 50.00    | RYANS          | 3         |
     |
| 100      | CHRISTMAS      | 5         | 5 Main Street
     |
+-----------+--------------+-----------+-----------------------------------+

!quit
```

Fonctions du module Universal Name

Open Name Parser

Exemple de script Hive

```
-- Register Universal Name Module [UNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/unm.hive.${project.version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
-- Open Name Parser is implemented as a UDF (User Defined function).
Hence it processes one row at a time and generates a map of key value
pairs for each row.
CREATE TEMPORARY FUNCTION opennameparser as
'com.pb.bdq.unm.process.hive.opennameparser.OpenNameParserUDF';

-- set rule
set rule='{"name":"name", "culture":"", "splitConjoinedNames":false,
"shortcutThreshold":0, "parseNaturalOrderPersonalNames":false,
"naturalOrderPersonalNamesPriority":1,
"parseReverseOrderPersonalNames":false,
"reverseOrderPersonalNamesPriority":2, "parseConjoinedNames":false,
"naturalOrderConjoinedPersonalNamesPriority":3,
"reverseOrderConjoinedPersonalNamesPriority":4,
"parseBusinessNames":false, "businessNamesPriority":5}';

-- Set Reference Directory. This must be a local path on cluster machines
 and must be present at the same path on each node of the cluster.
set refdir='/home/hadoop/reference/';

-- set header
set header='inputrecordid,Name,nametype';

-- Execute Query on the desired table, to display the job output on
console. This query returns a map of key value pairs containing output
 fields for each row.
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"] from (select opennameparser(${hiveconf:rule},
 ${hiveconf:refdir}, ${hiveconf:header}, inputrecordid, name, nametype)
 as tmp1 from nameparser) as tmp LATERAL VIEW explode(tmp1) adTable AS
 adid;
```

```
-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '/home/hadoop/opennameparser/' row
format delimited FIELDS TERMINATED BY ',' lines terminated by '\n' STORED
 AS TEXTFILE
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"] from (select opennameparser(${hiveconf:rule},
 ${hiveconf:refdir}, ${hiveconf:header}, inputrecordid, name, nametype)
 as tmp1 from nameparser) as tmp LATERAL VIEW explode(tmp1) adTable AS
 adid;



--sample input data
+-------------------------------+------------------------+----------------------+
| inputrecordid        | name           |      nametype          |
+-------------------------------+------------------------+----------------------+
| 1                                  |JOHN VAN DER LINDEN-JONES |
Simple Name    |
| 2                                  |RYAN JOHN SMITH           |  Simple
 Name      |
+-------------------------------+------------------------+----------------------+

--sample output data
+----------------------------+--------------+----------------------+
| Name           | NameScore    |      CultureCode    |
+----------------------------+--------------+----------------------+
| JOHN VAN DER LINDEN-JONES  |   75         |       True           |
| RYAN JOHN SMITH            |   100        |       True           |
+----------------------------+-----+--------------------------------+
```

Annexe

In this section

A - Exceptions

In this section

Messages d'exception

Exceptions - QPI Java

- `<Classname>.<Member>` est vide ou null.
- Les valeurs minimales de `GroupbyMROption.numReduceTasks = 0` doivent être 1.
- Les valeurs minimales de `maxNumOfDuplicates` = 0 doivent être 1.
- Aucun fichier n'est disponible au niveau du chemin d'accès spécifié.
- Impossible d'identifier le fichier d'entrée comme fichier Suspect ou Candidat.
- ExpressMatchKey défini, mais pas disponibles pour record\t.
- Impossible d'obtenir le nom de fichier de InputSplit.
- Impossible d'initialiser le moteur.
- Erreur lors du traitement des enregistrements consolidés :

Exceptions - Fonctions définies par l'utilisateur Hive

- _FUNC_ doit comporter les arguments minimaux.
- Impossible d'initialiser le moteur. Règle transmise : `<Rule used>`
- Type d'argument attendu : chaîne. Type d'argument reçu : `<Mismatched Type>`
- Exception : configuration `<Header string>` manquante.
- Erreur lors du traitement des enregistrements consolidés : `<Exception details>`
- Exception : colonne de champ de tri `<column name>` manquante dans la configuration du job.

B - Énumérations

In this section

Énumérations courantes

Énumération `MatchingAlgorithm`

Package : `com.pb.bdq.api.matcher`

Classe : `Algorithm`

1. Acronyme
2. CharacterFrequency
3. DaitchMokotoffSoundex
4. Date
5. DoubleMetaphone
6. EditDistance
7. EuclideanDistance
8. ExactMatch
9. Initiales
10. JaroWinklerDistance
11. KeyboardDistance
12. Koeln
13. KullbackLeiblerDistance
14. Metaphone
15. SpanishMetaphone
16. Metaphone3
17. NGramDistance
18. NGramSimilarity
19. NumericString
20. Nysiis
21. Phonix
22. Soundex
23. Sous-chaîne
24. SyllableAlignment

Énumération `Algorithm`

Package : `com.pb.bdq.api.matchkeygenerator`

Classe : `MatchKeyRule`

1. Soundex
2. Metaphone
3. SpanishMetaphone
4. DoubleMetaphone
5. Nysiis

6. Phonix
7. Metaphone3
8. Koeln
9. Consonne
10. Sous-chaîne

Énumération `RecordSeparator`
Package : `com.pb.bdq.common.job`

Classe : `FilePath`

1. WINDOWS
2. LINUX
3. MACINTOSH

Énumération `ReferenceDataPathLocation`
Package : `com.pb.bdq.common.job`

Constante d'énumération	Description
HDFS	Les données de références sont placées dans un répertoire HDFS.
LocaltoDataNodes	Les données de référence sont placées sur tous les nœuds de données disponibles dans le cluster.

Énumération `Operation`
Package : `com.pb.bdq.api.consolidation`

1. CONTAINS
2. HIGHEST
3. LOWEST
4. NOT_EQUAL
5. GREATER
6. LESSER
7. EQUAL
8. GREATER_THAN_EQUAL_TO
9. LESS_THAN_EQUAL_TO
10. IS_EMPTY
11. IS_NOT_EMPTY
12. MOST_COMMON
13. LONGEST
14. SHORTEST

Énumération `MatchingMethod`
Package : `com.pb.bdq.api.matcher`

Classe : `ParentMatchRule`

1. AllTrue
2. AnyTrue
3. BasedOnThreshold

Énumération `ScoringMethod`
Package : `com.pb.bdq.api.matcher`

Classe : `MatchRule`

1. Minimum
2. Maximum
3. Moyenne
4. WeightedAverage
5. VectorSummation

Énumération `MissingDataMethod`
Package : `com.pb.bdq.api.matcher`

Classe : `MatchRule`

1. IgnoreBlanks
2. CountAs100
3. CountAs0
4. CompareBlanks

Énumération `JoinType`
Package : `com.pb.bdq.api.consolidation`

Classe : `ConjoinedRule`

1. OR
2. AND

Énumération `IncludeTerm`
Package : `com.pb.bdq.api.advtransformer`

Classe : `TableDataExtraction`

1. ExtractedData
2. NonExtractedData
3. TermNeither

Énumération `Extract`
Package : `com.pb.bdq.api.advtransformer`

Classe : `TableDataExtraction`

1. ExtractTerm
2. ExtractNWordsLeft
3. ExtractNWordsRight

Énumération `AdvTransformerExtractionType`

Package : `com.pb.bdq.api.advtransformer`

Classe : `AbstractAdvancedTransformerRules`

1. TableData
2. RegularExpression

Énumération `MatchRuleType`

Package : `com.pb.bdq.api.matcher`

Classe : `MatchRule`

1. Parent
2. Child

Énumération `SortInput`

Package : `com.pb.bdq.api.matcher`

Classe : `MatchRule`

1. CHARS
2. TERMS

Énumération `TableLookupAction`

Package : `com.pb.bdq.api.tablelookup`

Classe : `AbstractTableLookupRule`

1. Normaliser
2. Catégoriser
3. Identifier

Énumérations Universal Addressing

Énumération `DatabaseType`

Package : `com.pb.bdq.api.uam.global`

Classe : `GlobalAddressingEngineConfiguration`

1. BATCH_INTERACTIVE
2. FASTCOMPLETION

3. CERTIFIED

Énumération `PreloadingType`
Package : `com.pb.bdq.api.uam.global`

Classe : `GlobalAddressingEngineConfiguration`

1. NONE
2. FULL
3. PARTIAL

Énumération `CountryCodes`
Package : `com.pb.bdq.api.uam`

Description : Codes par ordre alphabétique affectés à tous les pays pris en charge.

Énumération `StateProvinceType`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. COUNTRY_STANDARD
2. ABBREVIATION
3. EXTENDED

Énumération `CountryType`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. ISO2
2. ISO3
3. ISO_NUMBER
4. NAME_CN
5. NAME_DA
6. NAME_DE
7. NAME_EN
8. NAME_ES
9. NAME_FI
10. NAME_FR
11. NAME_GR
12. NAME_HU
13. NAME_IT
14. NAME_JP
15. NAME_KR
16. NAME_NL
17. NAME_PL

18. NAME_PT
19. NAME_RU
20. NAME_SA
21. NAME_SE

Énumération `PreferredScript`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. DATABASE
2. POSTAL_ADMIN_PREF
3. POSTAL_ADMIN_ALT
4. LATIN
5. LATIN_ALT
6. ASCII_SIMPLIFIED
7. ASCII_EXTENDED

Énumération `PreferredLanguage`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. DATABASE
2. ANGLAIS

Énumération `Casing`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. NATIF
2. UPPER
3. LOWER
4. MIXED
5. NOCHANGE

Énumération `OptimizationLevel`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. NARROW
2. STANDARD
3. WIDE

Énumération `Mode`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. BATCH
2. CERTIFIED
3. FASTCOMPLETION
4. INTERACTIF
5. PARSE

Énumération `MatchingScope`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. TOUT
2. LOCALITY_LEVEL
3. STREET_LEVEL
4. DELIVERYPOINT_LEVEL

Énumération `FormatDelimiter`
Package : `com.pb.bdq.api.uam.global`

Interface :`GlobalAddressingInputOption`

1. RETOUR CHARIOT SAUT DE LIGNE (CRLF)
2. SAUT DE LIGNE (LF)
3. RETOUR CHARIOT (CR)
4. POINT-VIRGULE
5. VIRGULE
6. TABULATION (TAB)
7. BARRE VERTICALE
8. ESPACE

Énumération `ExhaustedAction`
Package : `com.pb.bdq.api.uam.loqate`

Classe : `LoqateAddressingGeneralConfiguration`

1. GROW
2. BLOCK
3. FAIL

Énumération `AcceptanceLevel`
Package : `com.pb.bdq.api.uam.loqate.validate`

Classe : `LoqateAddressingValidateConfiguration`

1. Level0
2. Level1

3. Level2
4. Level3
5. Level4
6. Level5

Énumération `OutputCasing`
Package : `com.pb.bdq.api.uam.loqate.validate`

Classe : `LoqateAddressingValidateConfiguration`

1. Mixte
2. Majuscules

Énumération `CountryFormat`
Package : `com.pb.bdq.api.uam.loqate.validate`

Classe : `LoqateAddressingValidateConfiguration`

1. ANGLAIS
2. ISO
3. UPU

Énumération `ScriptAlphabet`
Package : `com.pb.bdq.api.uam.loqate.validate`

Classe : `LoqateAddressingValidateConfiguration`

1. InputScript
2. Native
3. Latin_English

Énumération `CacheSize`
Package : `com.pb.bdq.api.uam.global`

Classe : `GlobalAddressingGeneralConfiguration`

1. NONE
2. SMALL
3. LARGE

Énumération `RangesToExpand`
Package : `com.pb.bdq.api.uam.global`

Classe : `GlobalAddressingGeneralConfiguration`

1. NONE
2. ONLY_WITH_VALID_ITEMS

Énumération `FlexibleRangeExpansion`
Package : `com.pb.bdq.api.uam.global`

Classe : `GlobalAddressingGeneralConfiguration`

1. ON
2. Désactivé

Énumération `CasingType`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. MIXED
2. UPPER

Énumération `CityNameFormat`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. CITY_FORMAT_LONG
2. CITY_FORMAT_SHORT
3. CITY_FORMAT_STANDARD

Énumération `OutputCountryFormat`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. ANGLAIS
2. FRANÇAIS
3. ALLEMAND
4. ESPAGNOL
5. ISO
6. UPU

Énumération `DualAddressLogic`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. DUAL_NORMAL
2. DUAL_PO_BOX
3. DUAL_STREET

Énumération `StandardAddressFormat`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_FORMAT_COMBINED_UNIT
2. STANDARD_ADDRESS_FORMAT_SEPARATE_UNIT

3. STANDARD_ADDRESS_FORMAT_SEPARATE_DUAL

Énumération `StreetMatchingStrictness`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

Énumération `FirmMatchingStrictness`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

Énumération `DirectionalMatchingStrictness`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

Énumération `StandardAddressPMBLine`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_PMB_LINE_NONE
2. STANDARD_ADDRESS_PMB_LINE_1
3. STANDARD_ADDRESS_PMB_LINE_2

Énumération `PreferredCity`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. CITY_OVERRIDE_NAME_ZIP4
2. CITY_USPS_STATE_FILE
3. CITY_PRIMARY_NAME

Énumération `DPVFileType`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressGeneralConfiguration`

1. SPLIT
2. FULL
3. FLAT

Énumération `DPVMemoryModel`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

Énumération `LacsLinkMemoryModel`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

Énumération `SuiteLinkMemoryModel`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

Énumération `DPVSuccessStatusCondition`
Package : `com.pb.bdq.api.universaladdress`

Classe : `UniversalAddressInputConfiguration`

1. DPV_CONDITON_FULL
2. DPV_CONDITON_PARTIAL
3. DPV_CONDITON_ALWAYS

Énumération `UAMCASSReportType`

Package : `com.pb.bdq.uam.common`

1. CASS_3553
2. CASS_DETAIL
3. CASS_DETAIL2
4. CASS_DETAIL3

C - Prise en charge du module et des codes ISO de pays

In this section

Prise en charge du module et des codes ISO de pays

Le tableau répertorie les codes ISO à deux et trois chiffres de chaque pays.

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Afghanistan	AF	AFG
Aland, îles	AX	ALA
Albanie	AL	ALB
Algérie	DZ	DZA
Samoa américaines	AS	ASM
Andorre	AD	AND
Angola	AO	AGO
Anguilla	AI	AIA
Antarctique	AQ	ATA
Antigua et Barbuda	AG	ATG
Argentine	AR	ARG
Arménie	AM	ARM

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Aruba	AW	ABW
Australie	AU	AUS
Autriche	AT	AUT
Azerbaïdjan	AZ	AZE
Bahamas	BS	BHS
Bahreïn	BH	BHR
Bangladesh	BD	BGD
Barbade	BB	BRB
Biélorussie	BY	BLR
Belgique	BE	BEL
Bélize	BZ	BLZ
Bénin	BJ	BEN
Bermudes	BM	BMU
Bhoutan	BT	BTN

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Bolivie, État plurinational de	BO	BOL
Bonaire, Saint-Eustache et Saba	BQ	BES
Bosnie-Herzégovine	BA	BIH
Botswana	BW	BWA
Île Bouvet	BV	BVT
Brésil	BR	BRA
Territoire britannique de l'Océan indien	IO	IOT
Brunéi Darussalam	BN	BRN
Bulgarie	BG	BGR
Burkina Faso	BF	BFA
Burundi	BI	BDI
Cambodge	KH	KHM
Cameroun	CM	CMR
Canada	CA	CAN

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Cap-Vert	CV	CPV
Îles Caïmans	KY	CYM
République centrafricaine	CF	CAF
Tchad	TD	TCD
Chili	CL	CHL
Chine	CN	CHN
Île Christmas	CX	CXR
Îles Cocos (Keeling)	CC	CCK
Colombie	CO	COL
Comores	KM	COM
Congo	CG	COG
Congo, République démocratique du	CD	COD
Îles Cook	CK	COK
Costa Rica	RETOUR CHARIOT (CR)	CRI

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Côte d'Ivoire	CI	CIV
Croatie	HR	HRV
Cuba	CU	CUB
Curaçao	CW	CUW
Chypre	CY	CYP
République tchèque	CZ	CZE
Danemark	DK	DNK
Djibouti	DJ	DJI
Dominique	DM	DMA
République dominicaine	DO	DOM
Équateur	EC	ECU
Égypte	EG	EGY
El Salvador	SV	SLV
Guinée équatoriale	GQ	GNQ

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Erythrée	ER	ERI
Estonie	EE	EST
Éthiopie	ET	ETH
Falkland, îles (Malvinas)	FK	FLK
Îles Féroé	FO	FRO
Fiji	FJ	FJI
Finlande	FI	FIN
France	FR	FRA
Guyane française	GF	GUF
Polynésie française	PF	PYF
Terres australes françaises	TF	ATF
Gabon	GA	GAB
Gambie	GM	GMB
Géorgie	GE	GEO

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Allemagne	DE	DEU
Ghana	GH	GHA
Gibraltar	GI	GIB
Grèce	GR	GRC
Groenland	GL	GRL
Grenade	GD	GRD
Guadeloupe	GP	GLP
Guam	GU	GUM
Guatemala	GT	GTM
Guernesey	GG	GGY
Guinée	GN	GIN
Guinée-Bissau	GW	GNB
Guyane	GY	GUY
Haïti	HT	HTI

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Îles Heard-et-MacDonald	HM	HMD
Saint-Siège (État de la cité du Vatican)	VA	VAT
Honduras	HN	HND
Hong Kong	HK	HKG
Hongrie	HU	HUN
Islande	IS	ISL
Inde	IN	IND
Indonésie	ID	IDN
Iran, République islamique d'	IR	IRN
Irak	IQ	IRQ
Irlande	IE	IRL
ïle de Man	IM	IMN
Israël	IL	ISR
Italie	IT	ITA

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Jamaïque	JM	JAM
Japon	JP	JPN
Jersey	JE	JEY
Jordanie	JO	JOR
Kazakhstan	KZ	KAZ
Kenya	KE	KEN
Kiribati	KI	KIR
Corée, République populaire démocratique de	KP	PRK
Corée, République de	KR	KOR
Kosovo	KS	KOS
Koweït	KW	KWT
Kirghizistan	KG	KGZ
République démocratique populaire Lao	LA	LAO
Lettonie	LV	LVA

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Liban	LB	LBN
Lesotho	LS	LSO
Libéria	LR	LBR
Libyenne, Jamahiriya arabe	LY	LBY
Liechtenstein	LI	LIE
Lituanie	LT	LTU
Luxembourg	LU	LUX
Macao	MO	MAC
Macédoine, ancienne République yougoslave de	MK	MKD
Madagascar	MG	MDG
Malawi	MW	MWI
Malaisie	MY	MYS
Maldives	MV	MDV
Mali	ML	MLI

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Malte	MT	MLT
Îles Marshall	MH	MHL
Martinique	MQ	MTQ
Mauritanie	MR	MRT
Maurice	MU	MUS
Mayotte	YT	MYT
Mexique	MX	MEX
Micronésie, États fédérés de	FM	FSM
Moldova, République de	MD	MDA
Monaco	MC	MCO
Mongolie	MN	MNG
Monténégro	ME	MNE
Montserrat	MS	MSR
Maroc	MA	MAR

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Mozambique	MZ	MOZ
Myanmar	MM	MMR
Namibie	NA	NAM
Nauru	NR	NRU
Népal	NP	NPL
Pays-Bas	NL	NLD
Nouvelle-Calédonie	NC	NCL
Nouvelle-Zélande	NZ	NZL
Nicaragua	NI	NIC
Niger	NE	NER
Nigéria	NG	NGA
Niué	NU	NIU
Île Norfolk	NF	NFK
Îles Mariannes du Nord	MP	MNP

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Norvège	NO	NOR
Oman	OM	OMN
Pakistan	PK	PAK
Palaos	PW	PLW
Palestine, territoire occupé	PS	PSE
Panama	PA	PAN
Papouasie - Nouvelle-Guinée	PG	PNG
Paraguay	PY	PRY
Pérou	PE	PER
Philippines	PH	PHL
Pitcairn	PN	PCN
Pologne	PL	POL
Portugal	PT	PRT
Porto Rico	PR	PRI

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Qatar	QA	QAT
Réunion	RE	REU
Roumanie	RO	ROU
Russie, Fédération de	RU	RUS
Rwanda	RW	RWA
Saint-Barthélemy	BL	BLM
Sainte-Hélène, Ascension et Tristan da Cunha	SH	SHE
Saint-Christophe-et-Niévès	KN	KNA
Sainte Lucie	LC	LCA
Saint-Martin (partie française)	MF	MAF
Saint-Pierre-et-Miquelon	PM	SPM
Saint-Vincent-et-les-Grenadines	VC	VCT
Samoa	WS	WSM
Saint-Marin	SM	SMR

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Sao Tomé-et-Principe	ST	STP
Arabie saoudite	SA	SAU
Sénégal	SN	SEN
Serbie	RS	SRB
Seychelles	SC	SYC
Sierra Leone	SL	SLE
Singapour	SG	SGP
Saint-Martin (partie hollandaise)	SX	SXM
Slovaquie	SK	SVK
Slovénie	SI	SVN
Îles Salomon	SB	SLB
Somalie	SO	SOM
Afrique du Sud	ZA	ZAF
Géorgie du Sud et les îles Sandwich du Sud	GS	SGS

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Soudan du Sud	SS	SSD
Espagne	ES	ESP
Sri Lanka	LK	LKA
Soudan	SD	SDN
Surinam	SR	SUR
Svalbard et Jan Mayen	SJ	SJM
Swaziland	SZ	SWZ
Suède	SE	SWE
Suisse	CH	CHE
République arabe syrienne	SY	SYR
Taïwan, province de Chine	TW	TWN
Tadjikistan	TJ	TJK
Tanzanie, République unie de	TZ	TZA
Thaïlande	th	THA

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
Timor-Leste	TL	TLS
Togo	TG	TGO
Tokelau	TK	TKL
Tonga	TO	TON
Trinité-et-Tobago	TT	TTO
Tunisie	TN	TUN
Turquie	TR	TUR
Turkménistan	TM	TKM
Îles Turques-et-Caïques	TC	TCA
Tuvalu	TV	TUV
Ouganda	UG	UGA
Ukraine	UA	UKR
Émirats Arabes Unis	AE	ARE
Royaume-Uni	Go	GBR

Nom de pays ISO	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3
États-Unis	US	États-Unis
Îles mineures éloignées des États-Unis	UM	UMI
Uruguay	UY	URY
Ouzbékistan	UZ	UZB
Vanuatu	UV	VUT
Venezuela (République bolivarienne du)	VE	VEN
Vietnam	VN	VNM
Îles Vierges britanniques	VG	VGB
Îles Vierges des États-Unis	VI	VIR
Wallis-et-Futuna	WF	WLF
Sahara occidental	EH	ESH
Yémen	YE	YEM
Zambie	ZM	ZMB
Zimbabwe	ZW	ZWE

Notices

Avis USPS®

Pitney Bowes Inc. détient une licence non exclusive pour la publication et la vente de bases de données ZIP + 4® sur des supports optiques et magnétiques. Les marques de commerce suivantes appartiennent à United States Postal Service : CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS^{Link}, NCOA^{Link}, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite^{Link} , United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code et ZIP + 4. Cette liste de marques de commerce appartenant à U.S. Postal Service n'est pas exhaustive.

Pitney Bowes Inc. détient une licence non exclusive de USPS® pour le traitement NCOA^{Link}®.

Les prix des produits, des options et des services de Pitney Bowes Software ne sont pas établis, contrôlés ni approuvés par USPS® ni par le gouvernement des États-Unis. Lors de l'utilisation de données RDI™ pour déterminer les frais d'expédition de colis, le choix commercial de l'entreprise de distribution de colis à utiliser n'est pas fait par USPS® ni par le gouvernement des État-Unis.

Fournisseur de données et avis associés

Les produits de données contenus sur ce support et utilisés au sein des applications Pitney Bowes Software sont protégés par différentes marques de commerce et par un ou plusieurs des copyrights suivants :

Le jeu de données Geocode Address World contient des données distribuées sous licence de GeoNames Project (**www.geonames.org**) fournies sous la licence Creative Commons Attribution License (« Attribution License ») à l'adresse : **http://creativecommons.org/licenses/by/3.0/legalcode**. Votre utilisation des données GeoNames (décrites dans le Manuel de l'utilisateur Spectrum™ Technology Platform) est régie par les conditions de la licence Attribution License et tout conflit entre votre accord avec Pitney Bowes Software, Inc. et la licence Attribution License sera résolu en faveur de la licence Attribution License uniquement s'il concerne votre utilisation des données GeoNames.

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com