pitney bowes

Spectrum™ Technology Platform

バージョン 2019.1.0

Data & Address Quality for Big Data SDK ガイド

目次

1 - はじめに

はじめに 5
この SDK の対象ユーザ 5
ワークフロー 6
モジュールとジョブ 7
レポート 8

2 - SDK のインストール

システム要件 10
オペレーティング システムのアップデート 10
インストーラに含まれるもの 10
Windows での SDK のインストール 11
Linux での SDK のインストール 12

3 - リファレンス データの使用

リファレンス データの概要 16
DNM、UNM ジョブのリファレンス データの展開 19
GAM ジョブのリファレンス データの抽出 20
UAM ジョブのリファレンス データの抽出 21

4 - Java API

SDK Java API のコンポーネント 28
ソフトウェア開発キット (SDK) の使用 29
コモン API エンティティ 31
Advanced Matching モジュールのジョブ 35
Data Integration モジュールのジョブ 101
Data Normalization モジュールのジョブ 113
Global Addressing モジュールのジョブ 134
Universal Addressing モジュールのジョブ 145
Universal Name モジュールのジョブ 185

5 - XML 設定ファイル

サンプル設定ファイル 196
Advanced Matching モジュール 198
Data Integration モジュール 227
Data Normalization モジュール 235
Global Addressing モジュール 247
Universal Addressing モジュール 255
Universal Name モジュール 275

6 - Hive ユーザ定義関数

はじめに 281
Advanced Matching モジュールの関数 289
Data Integration モジュールの機能 318
Data Normalization モジュールの関数 321
Global Addressing モジュールの関数 335
Universal Addressing モジュールの機能 358
Universal Name モジュールの関数 378

7 - レポート カウンタ

レポート カウンタ 386
Advanced Matching モジュール 386
Global Addressing モジュール 389
Universal Addressing モジュール 391
Universal Name モジュール 396

第 章 : 付録

付録 A :
例外 400
付録 B :
列挙体 402
付録 C :
ISO 国コードとモジュール サポート 416

1 - はじめに

このセクションの構成

はじめに 5
この SDK の対象ユーザ 5
ワークフロー 6
モジュールとジョブ 7
レポート 8

はじめに

Spectrum™ Data & Address Quality for Big Data SDK は、Hadoop プラットフォーム上での データ品質操作のための MapReduce ジョブ、Spark ジョブ、および Hive ユーザ定義関数の作成、設定、実行に役立ちます。

この SDK を使用すると、Hadoop プラットフォームで直接ジョブを作成して実行できるため、ネットワーク遅延をなくし、クラスタ内で分散されたデータ品質プロセスを実行することにより、パフォーマンスを著しく向上させることが可能になります。

> **注：** Amazon S3 ネイティブ ファイル システム (s3n) を Hadoop MapReduce および Spark ジョブの入出力として使用することもできます。

SDK の使用

この SDK は現在、Java API および Hive ユーザ定義関数 (UDF) を介して使用できます。

- Java API
  - MapReduce API
  - Spark API - SDK は RDD および データセットの両方をサポートします
- Hive ユーザ定義関数

この SDK の対象ユーザ

Spectrum™ Data & Address Quality for Big Data SDK は、次のようなユーザを対象としています。

1. Hadoop 上のデータのデータ品質をチェックしたいと考えている顧客。
2. MapReduce または Spark のプログラミングに精通しており、特定の使用事例に関するソリューションを作成したいと考えている Hadoop 開発者。
3. 既存のデータに対して *データ クレンジング*、*データ拡張*、*データの重複排除*、*データ統合*の操作を実行したいと考えている Hadoop 開発者。
4. MapReduce または Spark の複雑な部分には精通していないが、SQL と構文的に似ている Hive Query Language (HQL) は抵抗なく使用できる Hive ユーザ。

ワークフロー

SDK を使用するには、以下のコンポーネントが必要です。

Spectrum™ Data & Address Quality for Big Data SDK のインストール

Spectrum™ Data & Address Quality for Big Data SDK の JAR ファイルをシステムにインストールし、アプリケーションで使用できるようにする必要があります。

クライアント アプリケーション

SDK を使用して必要なデータ品質操作を呼び出して実行するために作成する必要がある Java アプリケーション。Spectrum™ Data & Address Quality for Big Data SDK の JAR ファイルを Java アプリケーションにインポートする必要があります。

Hadoop プラットフォーム

Spectrum™ Data & Address Quality for Big Data SDK を使用してジョブを実行する際、まず、設定済みの Hadoop プラットフォームからデータが読み込まれ、関連する処理が実行された後、出力データが Hadoop プラットフォームに書き出されます。このため、使用するマシンで Hadoop のアクセスの詳細情報を正しく設定しておく必要があります。詳細については、**Windows での SDK のインストール**（11ページ）および**Linux での SDK のインストール**（12ページ）を参照してください。

> **注：** Amazon S3 ネイティブ ファイル システム (s3n) を Hadoop MapReduce および Spark ジョブの入出力として使用することもできます。

リファレンス データ

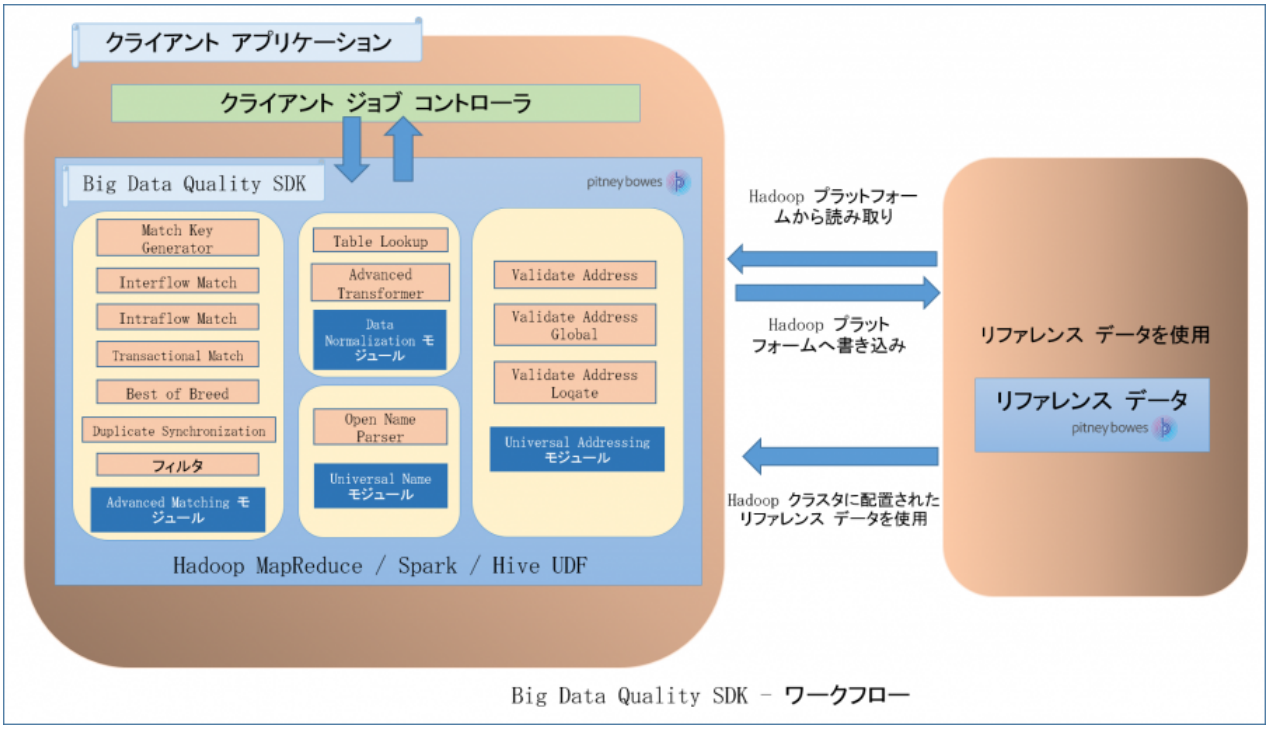
Spectrum™ Data & Address Quality for Big Data SDK で必要なリファレンス データは、Hadoop クラスタに配置されます。

Java API および Hive UDF、UDAF

Java API、Hive UDF または Hive UDAF を使用する場合は、リファレンス データをローカル データ ノードまたは Hadoop Distributed File System (HDFS) に配置できます。

- **ローカル データ ノード**: リファレンス データはクラスタ内の使用可能な全てのノードに配置されます。
- **Hadoop Distributed File System (HDFS)**リファレンス データは HDFS ディレクトリに配置され、ジョブの実行中に、データを分散キャッシュとしてダウンロードするか、ローカル ディレクトリにダウンロードするかを指定できます。詳細については、「**リファレンス データの配置および使用方法**（17ページ）」を参照してください。

注：また、この SDK では、パフォーマンス向上のための分散キャッシュを実行できます。



モジュールとジョブ

Spectrum™ Data & Address Quality for Big Data SDKでサポートされているモジュールと、これらの各モジュールで実行できるジョブは次のとおりです。

モジュール	**Java API および Hive UDF でサポートされるジョブ**
Advanced Matching モジュール	**Best of Breed ジョブ**（38ページ）
	Candidate Finder ジョブ（46ページ）
	Duplicate Synchronization ジョブ（52ページ）
	Filter ジョブ（59ページ）
	Intraflow ジョブ（76ページ）
	Interflow ジョブ（66ページ）
	Match Key Generator ジョブ（85ページ）
	Transactional Match ジョブ（92ページ）

モジュール	**Java API および Hive UDF でサポートされるジョブ**
Data Integration モジュール	**カスタム Groovy スクリプト ジョブ**（101ページ）
	Joiner ジョブ（107ページ） **注：** このジョブは Java API を介してのみ実行できます。
Data Normalization モジュール	**Advanced Transformer ジョブ**（114ページ）
	Open Parser ジョブ（122ページ）
	Table Lookup ジョブ（127ページ）
Global Addressing モジュール	**Global Address Validation**（134ページ）
Universal Addressing モジュール	**Validate Address**（146ページ）
	Validate Address Global（161ページ）
	Validate Address Loqate（174ページ）
Universal Name モジュール	**Open Name Parser**（186ページ）

レポート

これらのジョブに対してSpectrum™ Data & Address Quality for Big Data SDKを使用してレポートを生成できます。

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

さまざまなカウンタに基づいてレポートが生成されます。このレポートにより、実行したジョブの重複レコード数やユニーク レコード数など、有用なパラメータを追跡できます。レポートをコンソールで直接表示するか、ファイルとしてローカルに保存するかを選択できます。

2 - SDK のインストール

このセクションの構成

システム要件 10
オペレーティング システムのアップデート 10
インストーラに含まれるもの 10
Windows での SDK のインストール 11
Linux での SDK のインストール 12

システム要件

Hadoop Distributed File System (HDFS) の場合:

1. Java JDK 1.8 以降
2. Hadoop 2.6 以降
3. Spark 2.0.1 以降

Hive の場合:

1. Hive 1.2
2. 任意の Hive クライアント(例えば、Beeline など)

注： Spectrum™ Data and Address Quality for Big Data SDK は、Hadoop クラスタでのみ実行できます。

オペレーティング システムのアップデート

Spectrum™ Data & Address Quality for Big Data SDK をインストールする前に、お使いのオペレーティング システムに対して提供されているすべての最新製品アップデートを必ず適用してください。特に Java に関連する問題の修正は必須です。

インストーラに含まれるもの

SDK インストール ZIP ファイルには、次のコンポーネントが含まれています。

1. `Readme.txt`
2. `sdkinst.bin`: LINUX コンピュータ用のインストーラ。
3. `sdkinst.exe`: WINDOWS コンピュータ用のインストーラ。

Windows での SDK のインストール

Windows コンピュータ上に Spectrum™ Data & Address Quality for Big Data SDK をインストールするには、以下の手順を実行します。

1. ウェルカム メールまたはリリース通知メールに記載されているダウンロード手順に従って、Spectrum™ Data & Address Quality for Big Data SDK ZIP インストーラ ファイルをダウンロードします。

   注： 一般的なインストーラ ZIP ファイルの名前は、*BigDataSDK120F0101.zip*です。

2. Spectrum™ Data & Address Quality for Big Data SDK をインストールしたい場所で、ダウンロードした zip ファイルの内容を解凍します。
3. インストール ディレクトリに移動して、*sdkinst.exe* という名前のインストーラを探します。
4. `sdkinst.exe` をダブルクリックします。インストール ウィザードが表示されます。
5. **[次へ]** をクリックします。**[インストール フォルダを選択]** ウィンドウが表示されます。
   ここで、Spectrum™ Data & Address Quality for Big Data SDK をインストールするフォルダを指定できます。例えば、`C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK` です。
   a) **[選択]** ボタンをクリックして、必要なフォルダを選択します。
   b) デフォルト フォルダを選択する場合は、**[Restore Default Folder(デフォルト フォルダに戻す)]** ボタンをクリックします。
6. **[次へ]** をクリックします。
   [インストール前の注意事項] 画面で、インストール情報を確認します。
7. **[インストール]** をクリックします。
   Spectrum™ Data & Address Quality for Big Data SDK がコンピュータ上にインストールされます。
8. **[完了]** をクリックします。
9. SDKが正しくセットアップされていることを確認します。SDKをインストールした場所に移動します。例えば、`C:\Program Files\Pitney Bowes\Spectrum BigDataSDK\SDK` です。

コンピュータ上に SDK が正しくインストールされると、次のフォルダがインストール ディレクトリに追加されます。

- `API`
- `Documentation`

- `modules`
- `samples`
- `utilities`

注： Data Normalization モジュール、Global Addressing モジュール、Universal Name モジュールまたは Universal Addressing モジュールのジョブを使用するには、モジュールごとに個別のリファレンス データをインストールする必要があります。

Linux での SDK のインストール

Linux コンピュータ上にコマンド ラインで Spectrum™ Data & Address Quality for Big Data SDK をインストールするには、以下の手順を実行します。

1. ウェルカム メールまたはリリース通知メールに記載されているダウンロード手順に従って、Spectrum™ Data & Address Quality for Big Data SDK をダウンロードします。
2. Spectrum™ Data & Address Quality for Big Data SDK をインストールしたい場所で、ダウンロードしたファイルの内容を解凍します。
3. この場所にディレクトリを変更します。
4. 以下のコマンドを入力して、ファイルに対する `execute` 権限を取得します。
   ```
   chmod a+x sdkinst.bin
   ```
5. 次のコマンドを実行します。
   ```
   ./sdkinst.bin
   ```
   コマンド プロンプトの指示に従います。
6. 入力を求められたら、SDK をインストールするディレクトリを指定します。
   例えば、`/home/hadoop/BDQ_InstallPath` です。

   注： インストール ディレクトリとしてデフォルト以外のフォルダを選択する場合は、絶対インストール パスの長さが 34 文字を超えないようにしてください。

   デフォルトのインストール パスは 27 文字で許容範囲内に収まっています。

   ```
   /root/PBSpectrum_BigDataSDK
   ```

   インストール前の注意事項が表示されます。
7. 注意事項を確認し、`ENTER` キーを押してインストールを続行します。

8. インストール ログ ファイルを参照して、Spectrum™ Data & Address Quality for Big Data SDK が正しくインストールされたことを確認します。
9. 確認を終えたら、ENTER キーを押してインストーラを終了します。

コンピュータ上に SDK が正しくインストールされると、次のフォルダがインストール ディレクトリに追加されます。

- API
- Documentation
- modules
- samples
- utilities

> **注：** Data Normalization モジュール、Global Addressing モジュール、Universal Name モジュールまたは Universal Addressing モジュールのジョブを使用するには、モジュールごとに個別のリファレンス データをインストールする必要があります。

Acushare サービスの実行

最初の *Validate Address* ジョブを作成および実行する前に、Hadoop/Spark クラスタの各ノードで Acushare サービスを実行する必要があります。

> **注：** これは、*Validate Address* ジョブの実行前に一度だけ実行する必要がある操作です。

クラスタの各ノードで、以下の操作を行います。

1. Acushare セットアップ スクリプト `sdkrts.bin` を Spectrum™ Data & Address Quality for Big Data SDK インストール パスからノード上の任意の場所にコピーします。

   > **注：** SDK サーバーでは、Acushare セットアップ スクリプト `sdkrts.bin` は、`<BDQ SDK_InstallPath>/SDK/utilities/dbloader/aq/runtime/bin` にあります。

2. `installer.properties` ファイル (Acushare セットアップ スクリプトと同じディレクトリにあります) で、`USER_INSTALL_DIR` の詳細を指定します。これは Acushare サービスをインストールするパスです。

   重要： パスの長さは 60 文字以内にする必要があります。

   Acushare サービスは、インストールが正常に完了すると自動的に起動します。
3. Acushare サービスは、ノード上で手動で起動することもできます。`<Acushare runtime path>/runtime` に移動し、スクリプト ファイル `startrts.sh` を引数 `<Acushare runtime path>/runtime` 付きで実行します。

Acushare サービスの停止	ノード上の Acushare サービスを停止するには、`<Acushare runtime path>/runtime` に移動し、スクリプト ファイル `stoprts.sh` を引数 `<Acushare runtime path>/runtime` 付きで実行します。
Acushare サービスのアンインストール	Acushare サービスをノードからアンインストールするには、スクリプト ファイル `Uninstall_SDKRTS.sh` を `<Acushare runtime path>/Uninstall` から実行します。

3 - リファレンス データの使用

このセクションの構成

リファレンス データの概要 16
DNM、UNM ジョブのリファレンス データの展開 19
GAM ジョブのリファレンス データの抽出 20
UAM ジョブのリファレンス データの抽出 21

リファレンス データの概要

Pitney Bowes リファレンス データは、データ品質を保証するためにシステム内の他のデータ フィールドによって使用される許容値の集合を定義します。これによって、データの妥当性、正確性、一貫性が高まります。データからさらなる価値を引き出し、ビッグ データ システムから信頼できるデータを取得することができるようになります。

例えば、Data Normalization モジュールでリファレンス データを使用すると、企業全体にわたって単一の顧客の同一性を確立することができます。顧客情報の適切な定義は、運用効率を改善するための第一歩です。

リファレンス データは eStore で定期的に更新されるので、そこから取得して、次のいずれかの場所にインストールする必要があります。

- Hadoop クラスタのすべてのデータ ノード
- Hadoop Distributed File System (HDFS)

リファレンス データをインストールするためのスクリプト

リファレンス データをインストールするためのスクリプトは、SDK インストール ディレクトリの `Utilities/dbloader` フォルダにあります。この場所には次の子フォルダがあります。

dataquality このフォルダの内容は、以下のモジュールの参照データをインストールするための JAR やスクリプトです。

- Data Normalization モジュール
- Universal Name モジュール

> **注:** 詳細については、**インタラクティブ ユーティリティによる解凍**（23ページ）および**サイレント スクリプトを使用した解凍**（24ページ）を参照してください。

aq

- リファレンス データをインストールするための `scripts/server/installdb_unc.sh` および `scripts/server/silentInstalldb_unc.sh` スクリプト。このデータをインストールまたは展開するには、このスクリプトを実行する必要があります。
- `runtime` Universal Addressing モジュールの *Validate Address* ジョブのための Acushare サービス セットアップ情報が含まれているフォルダ。

注 : 詳細については、**インタラクティブ ユーティリティによる解凍**（23ページ）および**サイレント スクリプトを使用した解凍**（24ページ）を参照してください。

リファレンス データの配置および使用方法

リファレンス データは、以下の場所のいずれかに置くことができます。

- Hadoop クラスタのすべてのデータ ノード
- Hadoop Distributed File System (HDFS): リファレンス データが HDFS 上にある場合は、このデータの管理用オプションとして以下の 2 つがジョブの実行時に提供されます。
  - **現在の作業ディレクトリにダウンロードする**

    リファレンス データは、一時ファイルとして作業ディレクトリにダウンロードされます。これらのファイルは、ジョブが完了するたびに作業ディレクトリから削除され、各ジョブに必須のリファレンス データは新たにダウンロードされます。
  - **データをローカル パスにダウンロードする**

    リファレンス データは、指定したローカル データ パスにダウンロードされ、HDFS 上でリフレッシュされるまではすべてのジョブで使用できる状態が続きます。

HDFS でのリファレンス データの管理

指定したローカル パスにリファレンス データを HDFS から適切にダウンロードし、そのリファレンス データを使用してジョブを実行できるようにするには、以下の点を確認する必要があります。

- ジョブを実行中のユーザに、各データ ノード上のローカル ドライブへの書き込みアクセス権がある。
- データ ノードのそれぞれに、HDFS からデータをダウンロードするのに十分なハードディスク空き領域がある。

データをローカル パスにダウンロードする利点

- リファレンス データがすべてのノードにコピーまたは配置される場合 (Localtodatanodes オプション) のように、ノードのそれぞれにデータを配置する必要はありません。
- 所定のどのデータ ノード上でも、同じバージョンのデータが一度だけダウンロードされます。
- リファレンス データのダウンロードに制限はありません。

ジョブで指定するプロパティ

以下に、選択したリファレンス データの戦略とパスを示すために、ジョブ設定ファイルで指定する必要があるプロパティを示します。

- **リファレンス データを Hadoop クラスタのすべてのデータ ノード上に配置する**:

  JSON 文字列で、以下の詳細を指定します。

  - *referenceDataPathLocation*: `LocaltoDataNodes`
  - *dataDir*: リファレンス データが配置されるパス。

```
<property>
        <name>pb.bdq.reference.data</name>
        <value>{"referenceDataPathLocation":"LocaltoDataNodes",
               "dataDir":"/home/data/referenceData"}</value>
        <description>Pass reference data details as JSON
format.</description>
</property>
```

- **リファレンス データを Hadoop Distributed File System (HDFS) 上に配置し、分散キャッシュ モードを使用する:** JSON 文字列で、以下の詳細を指定します。

  - *referenceDataPathLocation*: `HDFS`
  - *dataDir*: HDFS 上のリファレンス データのパス
  - *dataDownloader*: `DC`

```
 <property>
        <name>pb.bdq.reference.data</name>
        <value>{"referenceDataPathLocation":"HDFS",
        "dataDir":"./referenceData",
        "dataDownloader":{"dataDownloader":"DC"}}</value>
        <description>Pass reference data details as JSON format.
        Pass above format for DATA DOWNLOADER when data is in
HDFS</description>
 </property>
```

- リファレンス データを Hadoop Distributed File System (HDFS) 上に配置し、ローカル パスにダウンロードする:

  JSON 文字列で、以下の詳細を指定します。

  - *referenceDataPathLocation*: `HDFS`
  - *dataDir*: HDFS 上のリファレンス データのパス
  - *dataDownloader*: `HDFS`
  - *localFSRepository*: リファレンス データをローカルにダウンロードする必要があるパス。

```
<property>
        <name>pb.bdq.reference.data</name>
```

```
        <value>{"referenceDataPathLocation":"HDFS",
         "dataDir":"/home/data/dm/referenceData",
         "dataDownloader":{"dataDownloader":"HDFS",
         "localFSRepository":"/local/download"}}</value>
        <description>Pass reference data details as JSON
format.</description>
</property>
```

DNM、UNM ジョブのリファレンス データの展開

Data Normalization モジュールおよび **Universal Name** モジュールジョブのリファレンス データを展開するには、データ ローダー スクリプト (`installdb_dnm` など) を実行する必要があります。

スクリプト ファイル (`installerdb_dnm` など) と JAR ファイルが、同じフォルダに存在することを確認してください。

1. コンピュータにログインします。
2. SDK をインストールした場所にディレクトリを変更します。

   コンピュータ上に Spectrum™ Data & Address Quality for Big Data SDK を正しくインストールすると、リファレンス データ ローダーがディレクトリ `BDQ_InstallPath/SDK/utilities/dbloader/unix/bin` に配置されるはずです。

3. リファレンス データ ローダー スクリプトを実行します例えば、`installdb_dnm` です。ステージのリストが番号付きで表示され、ステージを選択するように求められます。
4. データをロードするステージに対応する番号を入力します。
5. ダウンロード後にリファレンス データ セットを展開して配置する場所のパスを指定します。

   リファレンス データ入力は、Data Normalization モジュールおよび Universal Name モジュールのジョブの実行に必要な、Data Normalization モジュールの基本テーブルや Core Name データベースです。

6. 出力ディレクトリのパスを指定します。
7. ログファイルを表示するかどうかを尋ねるプロンプトが表示されます。選択を適宜行います。
8. データのロードが開始します。データは、指定した出力ディレクトリに展開されます。
9. 各ステージに対して、この手順を繰り返します。

これで、リファレンス データを使用できるようになります。

注： ジョブを実行するには、リファレンス データ パスに対する `write` 権限が必要です。

ユーザ定義のリファレンス データ

入力データ ファイルを、付属の Pitney Bowes リファレンス データとともにリファレンス データとして使用することができます。入力ファイルの各行には、**セミコロン**で区切られたキーと値を指定する必要があります。入力ファイルを Pitney Bowes リファレンス データ形式に変換するには、次のコマンドをコンソールで実行します。

```
java -cp [dataquality-12.2-jar-with-dependencies.jar] UserLibraryCreater
[-input <arg>] [-output <arg>]
```

ここで、`-input <arg>` は入力ファイルを配置したパス、`-output <arg>` は変換された入力ファイルが配置されるパスです。

> **注 :** 変換した入力ファイルの名前の末尾には *-user* を付け (例: *cdq-AdvTransformer-user.db*、*cdq-AdvTransformer-user.lg*)、DNM および UNM のリファレンス データが以前に抽出されたパスに配置する必要があります。

GAM ジョブのリファレンス データの抽出

このセクションでは、**Global Addressing** モジュール ジョブのリファレンス データ ソースを取得して抽出する手順について説明します。リファレンス データ ファイルは、eStore に SPD 形式で置かれています。

1. eStore から SPD ファイルを取得します。各 SPD ファイルには、個別の国または地域用のリファレンス データがあります。GAV リファレンス データの例を以下に示します。

SPD File Name	SPD Description	
GAC062017.spd	China GAV data	GAV_CHINA_JUN2017
GAA062017.spd	Americas GAV data	GAV_AMERICAS_JUN2017
GAE062017.spd	EMEA GAV data	GAV_EMEA_JUN2017
GEP062017.spd	EMEA Premium GAV data	GAV_EMEA2_JUN2017
GAP062017.spd	APAC GAV data	GAV_APAC_JUN2017
GAW062017.spd	World GAV data	GAV_WORLD_JUN2017
GGR062017.spd	Germany AddressPoint GAV data	GAV_GERMANY_ADDRESSPOINT_JUN2017
GES062017.spd	Spanish TT GAV data	GAV_SPANISH_JUN2017
GBR062017.spd	Great Britain GAV data	GAV_GREATBRITAIN_JUN2017
GGB062017.spd	GBR AB Subscription GAV data	GAV_GBR_AB_JUN2017
GIN062017.spd	India (Domestic) GAV data	GAV_INDIA_JUN2017
GMX062017.spd	Mexico NAVTEQ GAV data	GAV_MEXICO_NAVTEQ_JUN2017
GSE062017.spd	Sweden GAV data	GAV_SWEDEN_JUN2017
GAU062017.spd	Australia GNAF GAV data	GAV_AUSTRALIA_GNAF_JUN2017

> **注 :** SPD ファイル名の数字は、リファレンス データの月と年を示しています。例えば `GAC062017` の場合は、`06` が 6 月を示し、`2017` が年を示しています。

2. HDFS でリファレンス データを使用するには、SPD ファイルを HDFS に配置し、ジョブの実行中にパスを使用します。
3. Hadoop クラスタのローカル データ ノードでリファレンス データを使用するには、コマンドを使ってローカル ディレクトリに SPD ファイルを抽出します。

```
unzip <spd file name> -d <directory to extract>
```

例を次に示します。

```
unzip GAC062017 -d /home/hadoop/hduser/GAM_Feb_2018_DB/databases
```

`GAC062017` は SPD ファイル名です。`/home/hadoop/hduser/GAM_Feb_2018_DB/databases` は、リファレンス データの抽出先となるディレクトリです。

UAM ジョブのリファレンス データの抽出

このセクションでは、Validate Address ジョブのために各種のリファレンス データ ソースを取得して解凍する方法を説明します。リファレンス データは、eStore から取得します。

> **注：** Validate Address および Validate Address Global ジョブでは、Hadoop クラスタのすべてのデータ ノードまたは Hadoop Distributed File System (HDFS) にリファレンス データが配置されている必要があります。Validate Address Loqate ジョブでは、リファレンス データが 1 つのノードに配置され、さらにそのデータがその他すべてのデータ ノードにマウントされている必要があります。

Validate Address Loqate

1. eStore からリファレンス データを取得します。
2. *zip* ファイルの内容を解凍します。
3. 解凍したファイルを 1 つのノードに配置し、さらに他のすべてのデータ ノードにマウントします。

これで、これらのファイルをさまざまな MapReduce ジョブや Spark ジョブ、ユーザ定義関数で使うことができます。

Validate Address Global - Address Doctor

> **注：** Validate Address Global - Address Doctor は、SPD 形式のリファレンス データをサポートしていません。

1. eStore からリファレンス データを取得します。*Validate Address Global* の場合、取得するリファレンス データは次の 6 個のデータ バンドルに含まれています。
   - UAM - Enhanced International - Americas - Bundle Data: 南北アメリカ大陸のデータを含みます。
   - UAM - Enhanced International - EMEA - Bundle Data: 欧州、中東、およびアフリカのデータを含みます。
   - UAM - Enhanced International - APAC - Bundle Data: アジア太平洋地区のデータを含みます。
2. 以上の *zip* ファイルをダウンロードし、HDFS に配置します。HDFS には *zip* ファイルを解凍して配置しないでください。

   注: リファレンス データをローカル ノードに配置するには、zip ファイルを解凍して、すべてのデータ ノードに配置します。すべてのデータ ノードでパスが同じになるように設定する必要があります。

これで、これらのファイルをさまざまな MapReduce ジョブや Spark ジョブ、ユーザ定義関数で使うことができます。

Validate Address - C1P

1. eStore から以下のリファレンス データ バンドルを取得します。
   - US_SUB
   - DPV
   - EWS
   - LACS
   - SUITE
2. データの抽出
   - スクリプト `installdb_unc.sh` を使用したインタラクティブ ユーティリティによる解凍 (**インタラクティブ ユーティリティによる解凍**（23ページ）を参照)
   - サイレント スクリプト `silentInstalldb_unc.sh`。(**サイレント スクリプトを使用した解凍**（24ページ）を参照)

データはがローカルまたはエッジ ノードに解凍され、ここから HDFS にプッシュして MapReduce ジョブや Spark ジョブ、Hive ユーザ定義関数で使うことができます。

インタラクティブ ユーティリティによる解凍

注： `execute` 権限が `aq` フォルダに付与されていることを確認します。

1. 管理者権限で、または root ユーザとして、ログインします。
2. ディレクトリを `<BDQ_Installation>/SDK/utilities/dbloader/aq/scripts/server` に変更します。
3. 次のコマンドを使用して、スクリプト `installdb_unc` を実行します。

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

   このコマンドは、Acushare サービスが実行中かどうかの確認も行い、まだ実行中でない場合はこのサービスを起動します。以下のオプションも表示されます。

   - **US サブスクリプション**: 1 を押すと、使用可能なデータ ロードのタイプがリストされます。
   - **終了**: 99 を押すと終了します。

4. ロードするデータのタイプを表す特定の数値を入力します。

   ```
   1. Subscription Database
   2. Delivery Point Validation
   3. Residential Delivery Indicator
   4. Early Warning System
   5. LACSLink Database
   6. SuiteLink Database

   99. Exit


   Enter the number of the type of data you want to load
   and then press enter:
   ```

5. eStore からリファレンス データを取得し、解凍して、フォルダ内に配置します。このフォルダまでのパスが入力パスとなります。例えば、`/home/hadoop/hduser/UAM_DEC_2017_DB/databases` です。
6. 入力パスを指定します。
   ユーティリティによってデフォルト出力パスが表示されます。
7. そのまま続行する場合は `c`、デフォルト パスを変更する場合は `m`、終了する場合は `q` を入力します。

```
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/

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

   ===>
```

入力データが、指定した出力ファイルの場所に解凍されます。

8. 新しい RDI ファイルの場所が正しいかどうかを確認するように求められます。`y` または `n` を入力します。

```
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.

   ===>
```

データのロードが開始します。データは、指定した出力ディレクトリに展開されます。

注： ロードするデータのタイプごとにこの手順を繰り返します。

サイレント スクリプトを使用した解凍

インタラクティブな操作を行わずにリファレンス データを解凍するには、`silentInstalldb_unc.sh` スクリプトを使用します。このスクリプトは引数を一度に受け取り、ダイアログを表示せずにデータベースをマシン上に解凍します。

注： `execute` 権限が `aq` フォルダに付与されていることを確認します。

1. 管理者権限で、または root ユーザとして、ログインします。
2. ディレクトリを `<BDQ_Installation>/SDK/utilities/dbloader/aq/scripts/server` に変更します。
3. 次のコマンドを使用して、スクリプト `silentInstalldb_unc.sh` を実行します。

```
./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path>
```

このコマンドでは、Acushare サービスが実行中かどうかも確認されます。実行中でなければ、このコマンドによってサービスが起動されます。

4. 以下の表に、コマンドの説明を示します。

必須	引数	説明
はい	`-i` ***-input***	解凍する入力データのパス。入力データが存在するフォルダまでのパスを指定する必要があります。例: `/home/hadoop/hduser/UAM_DEC_2017_DB/databases`
はい	`-o` ***-output***	解凍したデータの出力場所のパス
はい	`-d` ***-database***	解凍するデータベース タイプ値は次のとおりです。 • US_SUB • RDI • EWS • LACS • DPV • SUITE
はい	`-a` ***-acushare***	インストールされている acushare サービスのパス
いいえ	`-z` ***-zip***	出力ファイルを `tar.gz` 形式のファイルに格納します。 圧縮した `tar.gz` ファイルを作成するには、値 `Y` を渡します。デフォルトは `N` です。
いいえ	***-override***	出力場所をオーバーライドします。出力場所をオーバーライドしない場合は、値 `N` を渡します。デフォルトは `Y` です。
いいえ	***-optionaldb***	解凍時にオプションの EOT ファイルをロードします。オプションの EOT ファイルをロードしない場合は、値 `N` を渡します。デフォルトは `Y` です。 **注 :** US_SUB データベースでのみ有効です。

必須	引数	説明
いいえ	*-ewsfile*	EWS ファイルの名前を指定します。デフォルトは OUT です。 **注：** EWS データベースでのみ有効です。

解凍するデータベース タイプごとに個別のコマンドがあります。

- US_SUB: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -optionaldb N -zip Y`
- DPV: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`
- EWS: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -ewsfile <name of file> -zip Y`
- LACS: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`
- SUITE: `./silentInstalldb_unc.sh -input <input database path> -output <output database path> -database <database type> -acushare <acushare installation path> -override N -zip Y`

注： 必須およびオプションのパラメータの詳細を表示するには、ヘルプ コマンド `./silentInstalldb_unc.sh -help` を実行します。

4 - Java API

このセクションの構成

SDK Java API のコンポーネント 28
ソフトウェア開発キット (SDK) の使用 29
コモン API エンティティ 31
Advanced Matching モジュールのジョブ 35
Data Integration モジュールのジョブ 101
Data Normalization モジュールのジョブ 113
Global Addressing モジュールのジョブ 134
Universal Addressing モジュールのジョブ 145
Universal Name モジュールのジョブ 185

SDK Java API のコンポーネント

Java API を使って Spectrum™ Data & Address Quality for Big Data SDK ジョブを実行するための主要コンポーネントは、以下のとおりです。

JAR ファイル

1. Hadoop JAR ファイル。
2. 以下の表に示された、必要な Spectrum™ Data & Address Quality for Big Data SDK ジョブが属するモジュールの JAR ファイル。

モジュール	ジョブ	JAR ファイル
Advanced Matching モジュール	すべての AMM ジョブ	*amm.core-12.2.jar*
Data Integration モジュール	すべての DIM ジョブ	*dim.core-12.2.jar*
Data Normalization モジュール	すべての DNM ジョブ	*dnm.core-12.2.jar*
Global Addressing モジュール	Global Address Validation	*gam-global addressvalidation.core-12.2.jar*
Universal Addressing モジュール	Validate Address	*uam-universaladdress.core-12.2.jar*
Universal Addressing モジュール	Validate Address Global	*uam-global.core-12.2.jar*
Universal Addressing モジュール	Validate Address Loqate	*uam-loqate.core-12.2.jar*
Universal Name モジュール	すべての UNM ジョブ	*unm.core-12.2.jar*

設定ファイル	マッチルール、入力ファイル詳細情報、出力ファイル詳細情報、MapReduce または Spark 詳細設定など、ジョブの実行に必要なすべてのパラメータと値を含む、XML 形式のファイル。 XML 設定ファイルのサンプルは、`<Big Data Quality bundle>\samples\configuration` にあります。
クライアント Java アプリケーション	API を使用して、Java API で提供される必要な Spectrum™ Data & Address Quality for Big Data SDK ジョブを作成および実行する Java アプリケーション。
Hadoop プラットフォーム	作成されたジョブは、入力データにアクセスし、出力データをファイルに書き出すために、設定済みの Hadoop プラットフォームにアクセスします。

ソフトウェア開発キット (SDK) の使用

SDK は、次の 2 つのいずれかの方法で Spectrum™ Data & Address Quality for Big Data SDK ジョブの実行に使用できます。

1. コンソール上で、モジュール固有の JAR ファイルを直接実行し、XML 形式の各種設定プロパティ ファイルをコマンドの引数として渡します。

   MapReduce ジョブの場合は `hadoop` コマンド、Spark ジョブの場合は `submit-spark` コマンドを実行します。

   手順については、**設定プロパティ ファイルの使用**（196ページ）を参照してください。

2. 関連する Spectrum™ Data & Address Quality for Big Data SDK モジュールの JAR ファイルをインポートすることによって独自の Java クライアント プロジェクトを作成し、クライアント プロジェクト内で、対象のジョブに必要なすべてのジョブ設定を指定して、実行します。

   手順については、**Java アプリケーションの作成**（29ページ）を参照してください。

Java アプリケーションの作成

Spectrum™ Data & Address Quality for Big Data SDK がコンピュータ上にインストールされていることを確認します。

SDK を使用するには

1. 必要に応じて、次の方法のいずれかを使用して、SDK を使用するための Java プロジェクトを作成します。
   a) 必要なデータ品質操作を実行するための、特定の Java プロジェクトを作成する。
      この方法では、実行するデータ品質ジョブごとに、個別の Java プロジェクトを作成する必要があります。
   b) 必要なあらゆるデータ品質操作を実行するための共通の Java プロジェクトを、対応する実行時引数を使用して作成する。
      この方法では、必要なデータ品質操作に対応する実行時引数を受け入れる Java プロジェクトを、1 つのみ作成する必要があります。
2. SDK を使用するプロジェクトに Spectrum™ Data & Address Quality for Big Data SDK モジュール固有の JAR ファイルをインポートします。モジュール固有の JAR ファイルの一覧については、**SDK Java API のコンポーネント**（28ページ）を参照してください。
3. 必要な Hadoop JAR ファイルをプロジェクトにインポートします。
4. 必要なデータ品質ジョブを実行するためのアプリケーションを、適切な設定を用いて作成します。
5. Maven や Ant など、任意のビルド ツールを使用してプロジェクトを構築します。
   これにより、プロジェクトの JAR ファイルが作成されます。

   例えば、`MatchKeyGeneratorClient-with-dependencies.jar` が作成されます。
6. プロジェクトの JAR ファイルを Hadoop プラットフォームに配置します。
7. Hadoop プラットフォーム上で、コマンド プロンプトを使用して、JAR ファイルを置いた場所のパスにディレクトリを移動します。
8. 次のコマンドを使用して、プロジェクトの JAR を実行します。

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

   例:

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

Hadoop プラットフォーム上で必要なジョブが作成され、実行されました。

Java アプリケーションは、Hadoop プラットフォーム上の指定されたパスからの入力データにアクセスし、Hadoop プラットフォーム上にジョブを作成して実行します。ジョブの出力は、Hadoop プラットフォーム上の指定された出力パスのファイルに書き出されます。

コモン API エンティティ

ConjoinedRule

目的
統合ルールの種類。複数のルールが AND および OR 演算子で結合される場合に使用されます。結合ルール (ConjoinedRule) は、シンプル ルール (SimpleRule) をコンポーネントとして含むことができます。**SimpleRule**（35ページ） を参照してください。

このクラスによって、**Advanced Matching** モジュールと **Data Normalization** モジュールのジョブのルールを定義できます。

ConsolidationCondition

目的
Advanced Matching モジュールと **Data Normalization** モジュールのジョブの統合ルールと対応するアクションを指定します。

ConsolidationRule

目的
レコードに対してアクションが必要かどうかの判断基準となる、統合ルールを指定します。

このクラスによって、**Advanced Matching** モジュールと **Data Normalization** モジュールのジョブの統合ルールを定義できます。

ConsolidationAction

目的
特定の統合条件のために、グループ内の他のレコードにコピーする必要のあるフィールドを指定します。

このクラスによって、**Advanced Matching** モジュールと **Data Normalization** モジュールのジョブの統合アクションを定義できます。

FilePath

目的

ジョブを実行するための入力および出力テキスト ファイルの詳細を指定します。

表 1 : FilePath サブクラスのパラメータおよび値

パラメータ	値	データ タイプ

パス	入力および出力テキスト ファイルのパス
	ファイルが Amazon S3 Native FileSystem ファイル システム (s3n) の場合は、次の形式でパスを入力する必要があります。
	`s3n://AWS Access key ID: AWS Secret access key@<Bucket name followed by input file path>`
	`s3n:// AWS Access key ID: AWS Secret access key@<Bucket name followed by output file path>`
		Path

recordSeparator	レコード区切り文字	文字列
fieldSeperator	フィールド区切り文字	文字列
fileHeader	ヘッダー フィールド	文字列

JobConfig<T extends ProcessType>

目的

ジョブに対して Hadoop 設定を指定するインターフェイス。

MRJobConfig

目的

MapReduce ジョブに対して Hadoop 設定を指定します。

SparkJobConfig

目的
すべての Spark ジョブに対して Hadoop 設定を指定します。

> **注：** デフォルトで、RDD データ構造はすべての Spark ジョブで使用されます。DataSet API を有効にするには、**pb.bdq.spark.job.dataset.enable** を true に設定します。

JobDetail<T extends ProcessType>

目的
ジョブの作成に必要な基本情報を保管します。

JobFactory

目的
ジョブ インスタンスの作成を指定し、作成されるジョブの詳細を指定する、ベース インターフェイス。

JobPath

目的
ジョブの入力ソースおよび出力デスティネーションの詳細を指定する親クラス。

OrcFilePath

目的

ジョブを実行する ORC 形式ファイルの入力パスまたは出力パスを指定します。

`path` パラメータは、入力および出力ファイル パスを値に取ります。

ParquetFilePath

目的

ジョブを実行する Parquet 形式ファイルの入力パスまたは出力パスを指定します。

`path` パラメータは、入力および出力ファイル パスを値に取ります。

ProcessType

目的

MapReduce や Spark など、サポートされるすべてのプロセス タイプに使用される、親マークアップ インターフェイス。

MRProcessType

目的

ジョブに対し、MapReduce プロセス タイプを指定します。

SparkProcessType

目的

ジョブに対し、Spark プロセス タイプを指定します。

ReferenceDataPath

目的

ジョブのリファレンス データのパスを指定します。

ReportManager

目的

ジョブのレポート統計を取得するためのインターフェイス。

SimpleRule

目的
統合ルールの種類。シンプル ルール (SimpleRule) は単体、または結合ルール (ConjoinedRule) のコンポーネントとして使用できます。**ConjoinedRule**（31ページ） を参照してください。

例外

JobException

目的
適切なメッセージを表示して、ジョブ固有の例外を処理します。

Advanced Matching モジュールのジョブ

コモンモジュール API

AdvanceMatchDetail<T extends ProcessType>

目的
Advanced Matching モジュールのジョブの詳細を指定します。

AdvanceMatchFactory

目的
Advanced Matching モジュールのジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

GroupbyOption<T extends ProcessType>

目的
Advanced Matching ジョブに対してグループ化を行う列を指定します。

GroupbyMROption

目的
Advanced Matching MapReduce ジョブに対してグループ化を行う列を指定します。

GroupbySparkOption

目的
Advanced Matching Spark ジョブに対してグループ化を行う列を指定します。

MatchKeySettings

目的
Match Key Generator ジョブのマッチ キーの `List` を保持します。

MatchRule

目的
Advanced Matching ジョブのマッチング ルールが作成できます。

親ノードと子ノードの階層を定義することによって、これを行います。各ノードは、マッチングされる入力フィールドの 1 つに対応します。

ChildMatchRule

目的
フィールドを特定のアルゴリズムやその他のプロパティにマッピングする、マッチ ルールの子ノードを指定します。

ParentMatchRule

目的
他の親ノードと子ノードの論理的なグループ分けである、マッチ ルールの親ノードを指定します。

特殊なシナリオ

Group-By 列が空白のレコード

Group-By の値が空白のレコードはすべて、形式に誤りがあるとみなされ、出力 HDFS フォルダへ別のファイルに出力されます。

形式に誤りがあるこれらのファイルは、次のように名前が付けられます。

候補ファイル内の形式誤りレコード	候補ファイル内の Group-By 列が空白のレコードは、形式に誤りがあるとして破棄され、`malformedRecordsCandidate-m-<5 digit numeral>` という命名規則に従って名前が付けられたファイルに挿入されます。 例えば、`malformedRecordsCandidate-m-00000`、`malformedRecordsCandidate-m-00001` です。 これは、Interflow Match ジョブに適用されます。
サスペクト ファイル内の形式誤りレコード	サスペクト ファイル内の Group-By 列が空白のレコードは、形式に誤りがあるとして破棄され、`malformedRecordsSuspect-m-<5 digit numeral>` という命名規則に従って名前が付けられたファイルに挿入されます。 例えば、`malformedRecordsSuspect-m-00000`、`malformedRecordsSuspect-m-00001` です。 これは、Interflow Match ジョブに適用されます。
入力ファイル内の形式誤りレコード	入力ファイル内の Group-By 列が空白のレコードは、形式に誤りがあるとして破棄され、`malformedRecords-m-<5 digit numeral>` という命名規則に従って名前が付けられたファイルに挿入されます。 例えば、`malformedRecords-m-00000`、`malformedRecords-m-00001` です。 これは、Intraflow Match、Transactional Match、Best of Breed、Duplicate Synchronization、Filter のジョブに適用されます。

形式誤りレコードのカウンタ

1 回のジョブ実行における形式誤りレコードの数は、次のカウンタに保存されます。

- MALFORMED_CANDIDATE_RECORDS
- MALFORMED_SUSPECT_RECORDS
- MALFORMED_RECORDS

注: これらのカウンタの値には、`AdvanceMatchFactory` インスタンスの `getCounters()` メソッドを呼び出すことによってアクセスできます。

Best of Breed ジョブ

Best of Breed は、重複レコードのコレクションから選択する最良のデータを使用して新しい統合レコードを作成することで、重複レコードを統合します。この "スーパー" レコードは、最良の組み合わせレコードと呼ばれます。処理対象レコードの選択で使用するルールを定義します。処理が完了すると、最良の組み合わせレコードがシステムに保持されます。

API エンティティ

BestOfBreedConfiguration

Best of Breed 統合ジョブを実行するための統合ルールとテンプレート ルールを指定します。

BestofBreedDetail

目的

Best of Breed 統合ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	類似レコードのグループを結合することによって 1 つの Best of Breed レコードを作成する際に使用するフィールドを指定します。Best of Breed レコードは、各レコード グループに対して作成されます。 *MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。

パラメータ	説明
Best of Breed Configuration	類似レコードの各コレクションに対して Best of Breed レコードを作成する際に使用する、統合ルールとテンプレート ルールを定義します。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注: `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

入力列に加えて、Best of Breed ジョブの出力生成時に以下の列が追加されます。

パラメータ	説明	出力値
Collection Record Type	重複するレコードのコレクションからテンプレートと Best of Breed レコードを識別します。	テンプレート レコードが定義されている場合、とり得る値は次のとおりです。 **Primary** レコードが、コレクションの選択されたテンプレート レコードである場合。 **Secondary** レコードが、コレクションの選択されたテンプレート レコードでない場合。 **BestOfBreed** レコードが、コレクションの新しく作成された Best of Breed レコードである場合。 テンプレート レコードが定義されていない場合、とり得る値は **BestOfBreed** のみです。

注：[Collection Record Type] 以外の出力列は、Best of Breed 環境設定用の統合条件の作成時に定義された場合のみ表示されます。

Best of Breed MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Best of Breed ジョブの入力と出力の詳細を指定します。以下の手順に従って、`BestofBreedDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。

a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

b) `BestOfBreedConfiguration` のインスタンスを作成することによって、ジョブの統合ルールとテンプレート ルールを生成します。このインスタンスの中で、次の操作を行います。

1. `ConsolidationCondition` のインスタンスを使用して、テンプレート レコードを定義します。このインスタンスは、`ConsolidationRule` インスタンスで構成されます。
2. `ConsolidationCondition` のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

   `ConsolidationCondition` の各インスタンスは、`ConsolidationRule` インスタンスとそれに対応する `ConsolidationAction` インスタンスを使用して定義されます。

   注： `ConsolidationRule` の各インスタンスは、`SimpleRule` の 1 つのインスタンスによって定義するか、または、子の `SimpleRule` インスタンスとネストされた `ConjoinedRule` インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

c) `BestofBreedDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `BestOfBreedConfiguration` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `BestofBreedDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `BestofBreedDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath`のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `BestofBreedDetail` フィールドを使用して、ジョブの名前を設定します。

g) `BestofBreedDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `BestofBreedDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Best of Breed Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Best of Breed ジョブの入力と出力の詳細を指定します。以下の手順に従って、`BestofBreedDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

   GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

   b) `BestOfBreedConfiguration` のインスタンスを作成することによって、ジョブの統合ルールとテンプレート ルールを生成します。このインスタンスの中で、次の操作を行います。

1. `ConsolidationCondition` のインスタンスを使用して、テンプレート レコードを定義します。このインスタンスは、`ConsolidationRule` インスタンスで構成されます。
2. `ConsolidationCondition` のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

   `ConsolidationCondition` の各インスタンスは、`ConsolidationRule` インスタンスとそれに対応する `ConsolidationAction` インスタンスを使用して定義されます。

   注： `ConsolidationRule` の各インスタンスは、`SimpleRule` の 1 つのインスタンスによって定義するか、または、子の `SimpleRule` インスタンスとネストされた `ConjoinedRule` インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

c) `BestofBreedDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `BestOfBreedConfiguration` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `BestofBreedDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `BestofBreedDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `BestofBreedDetail` フィールドを使用して、ジョブの名前を設定します。

g) `BestofBreedDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `BestofBreedDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Candidate Finder ジョブ

Candidate Finder は、一連の潜在的なマッチを形成する候補レコードを取得します。検索インデックス検索は Transactional Match とは別に機能します。また、Candidate Finder では、データの書式によっては、サスペクト レコード、候補レコード、またはその両方のレコードの名前や住所のパーシングが必要となる場合もあります。

また、Candidate Finder ではフルテキスト インデックス検索も可能で、さまざまな検索タイプ (数値、範囲、すべて含む、いずれも含まない) と条件 (すべて真、いずれかが真) を使用して、文字やテキストの高度な検索条件を容易に定義できます。

> **注：** 検索インデックスを保存するには、クラスタで *HBase NoSQL* データベースが利用でき、アクセス可能であることが必要です。

API エンティティ

CandidateFinderDetail

Candidate Finder ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** 入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注：** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。
Index Name	検索インデックスの名前。
Complex Search Query	選択された検索クエリのタイプ。 **注：** Complex Search Query のみがサポートされます。
Index Output Fields	検索インデックス出力のフィールド名。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 注： `FilePath` の適切なコンストラクタを呼び出します。 **レコード区切り文字** 出力ファイル内で使用されるレコード区切り文字。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Job Name	ジョブの名前。
フェッチ バッチ サイズ	結果を処理するバッチのサイズ。デフォルトは 10000 です。
結果の最大数	インデックス検索で返す応答の最大数。デフォルトは 10 です。
開始レコード	検索結果の最初のレコード番号。デフォルトは 1 です。

出力列

入力列に加えて、Candidate Finder ジョブの出力生成時に以下の列が追加されます。

パラメータ	説明/有効値
CandidateCount	このフィールドは、処理中に返された候補の総数を表します。
CandidateGroup	このフィールドは、サスペクト レコードとその候補のグループを表します。各レコードには、CandidateGroup 番号が与えられます。そのレコードの候補には、同じ CandidateGroup 番号が与えられます。
HasDuplicates	候補が検出されたかどうかを識別します。有効な値を次に示します。 **Y** - サスペクト レコードで候補がある **N** - サスペクト レコードで候補がない **D** - 候補レコードである
TransactionRecordType	とり得る値は、**S** (サスペクト レコード)、 **D** (重複レコード)、および **U** (ユニーク レコード) です。

Candidate Finder MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Candidate Finder ジョブの入力と出力の詳細を指定します。以下の手順に従って、`ProcessType` を指定する `CandidateFinderDetail` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) インスタンス **MRJobConfig**（32ページ） で、`hbase_zookeeper_quorum` および `hbase_zookeeper_property_clientPort` の値を設定します。
   b) `ComplexSearchQuery` のインスタンスを作成することによって、ジョブのクエリを生成します。このインスタンスの中で、次の操作を行います。
      1. `QueryName`、`IndexFieldName`、`IndexFieldType` などのプロパティを設定します。`Numeric`、`Range`、`Contains All`、`Contains None` などの検索クエリを使用できます。
      2. 検索クエリ プロパティを設定し、`AND` や `OR` などの論理演算子を使用してこれらを接続します。

         注： `ComplexSearchQuery` の各インスタンスは、1 つのインスタンス、子のインスタンスの階層、または論理演算子を使用して結合されネストされたインスタンス

を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

c) `inputPath` インスタンスの `CandidateFinderDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

d) `CandidateFinderDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `jobName` インスタンスの `CandidateFinderDetail` フィールドを使用して、ジョブの名前を設定します。

f) `CandidateFinderDetail` インスタンスの `FetchBatchSize` フィールドを設定します。デフォルトは `10000` です。

g) `CandidateFinderDetail` インスタンスの `MaximumResults` フィールドを設定します。デフォルトは `10` です。

h) `CandidateFinderDetail` インスタンスの `StartingRecord` フィールドを設定します。デフォルトは `1` です。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `CandidateFinderDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Candidate Finder Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Candidate Finder ジョブの入力と出力の詳細を指定します。以下の手順に従って、`ProcessType` を指定する `CandidateFinderDetail` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。
   a) インスタンス **SparkJobConfig**（33ページ） で、`hbase_zookeeper_quorum` および `hbase_zookeeper_property_clientPort` の値を設定します。
   b) `ComplexSearchQuery` のインスタンスを作成することによって、ジョブのクエリを生成します。このインスタンスの中で、次の操作を行います。
      1. `QueryName`、`IndexFieldName`、`IndexFieldType` などのプロパティを設定します。`Numeric`、`Range`、`Contains All`、`Contains None` などの検索クエリを使用できます。
      2. 検索クエリ プロパティを設定し、`AND` や `OR` などの論理演算子を使用してこれらを接続します。

         注： `ComplexSearchQuery` の各インスタンスは、1 つのインスタンス、子のインスタンスの階層、または論理演算子を使用して結合されネストされたインスタンスを使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

   c) `inputPath` インスタンスの `CandidateFinderDetail` フィールドを使用して、入力ファイルの詳細を設定します。
      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。
   d) `CandidateFinderDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。
      - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `jobName` インスタンスの `CandidateFinderDetail` フィールドを使用して、ジョブの名前を設定します。

f) `CandidateFinderDetail` インスタンスの `FetchBatchSize` フィールドを設定します。デフォルトは `10000` です。

g) `CandidateFinderDetail` インスタンスの `MaximumResults` フィールドを設定します。デフォルトは `10` です。

h) `CandidateFinderDetail` インスタンスの `StartingRecord` フィールドを設定します。デフォルトは `1` です。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `CandidateFinderDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Duplicate Synchronization ジョブ

Duplicate Synchronization は、レコードのコレクションから、そのコレクション内のすべてのレコードの対応するフィールドにコピーするフィールドを指定します。フィールド データをコレクション内の別のレコードにコピーするときに従う必要があるルールを指定できます。処理が完了すると、コレクション内のレコードがすべて保持されます。

API エンティティ

DuplicateSynchronizationConfiguration

Duplicate Synchronization 統合ジョブを実行するための統合ルールを指定します。

DuplicateSyncDetail

目的
Duplicate Synchronization 統合ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	同期するレコードのグループを作成するのに使用するフィールドを指定します。 *MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合、Group-By オプションを作成するため次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **注 :** 入力にグループがない場合は、このパラメータに null を設定します。その場合は、データ全体が 1 つのグループであるとみなされます。
Duplicate Synchronization Configuration	あるレコードのフィールドをコレクションの他のレコードにコピーするときに適用されるルール。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

[Duplicate Synchronization Configuration] 入力パラメータで定義された統合条件に基づき、必要に応じて、入力列以外の列が出力に追加される場合があります。

Duplicate Synchronization MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Duplicate Synchronization ジョブの入力と出力の詳細を指定します。以下の手順に従って、`DuplicateSyncDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

      GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

   b) `DuplicateSynchronizationConfiguration` のインスタンスを作成することによって、ジョブの統合条件を生成します。このインスタンスの中で、`ConsolidationCondition` のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

      `ConsolidationCondition` の各インスタンスは、`ConsolidationRule` インスタンスとそれに対応する `ConsolidationAction` インスタンスを使用して定義されます。

      > **注：** `ConsolidationRule` の各インスタンスは、`SimpleRule` の 1 つのインスタンスによって定義するか、または、子の `SimpleRule` インスタンスとネストされた `ConjoinedRule` インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

   c) `DuplicateSyncDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `DuplicateSynchronizationConfiguration` インスタンスを、コンストラクタの引数として渡します。

      `JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

   d) `inputPath` インスタンスの `DuplicateSyncDetail` フィールドを使用して、入力ファイルの詳細を設定します。

      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `DuplicateSyncDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `DuplicateSyncDetail` フィールドを使用して、ジョブの名前を設定します。

g) `DuplicateSyncDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. 先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出し、ジョブを作成します。ここで、上の `DuplicateSyncDetail` のインスタンスを引数として渡します。
`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Duplicate Synchronization Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Duplicate Synchronization ジョブの入力と出力の詳細を指定します。以下の手順に従って、`DuplicateSyncDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

b) `DuplicateSynchronizationConfiguration` のインスタンスを作成することによって、ジョブの統合条件を生成します。このインスタンスの中で、

`ConsolidationCondition` のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

`ConsolidationCondition` の各インスタンスは、`ConsolidationRule` インスタンスとそれに対応する `ConsolidationAction` インスタンスを使用して定義されます。

> **注：** `ConsolidationRule` の各インスタンスは、`SimpleRule` の 1 つのインスタンスによって定義するか、または、子の `SimpleRule` インスタンスとネストされた `ConjoinedRule` インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

c) `DuplicateSyncDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `DuplicateSynchronizationConfiguration` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `DuplicateSyncDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `DuplicateSyncDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `DuplicateSyncDetail` フィールドを使用して、ジョブの名前を設定します。

g) `DuplicateSyncDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `DuplicateSyncDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。
4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Filter ジョブ

Filter ステージでは、指定したルールに基づいて、レコードをレコードのグループに保持または削除します。

API エンティティ

FilterConfiguration

Filter 統合ジョブを実行するための統合ルールを指定します。

FilterDetail

目的
Filter 統合ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	フィルタリングするレコードのグループを作成するのに使用するフィールドを指定します。Filter ジョブは、各グループからのレコードを 1 つ以上保持します。 *MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合、Group-By オプションを作成するため次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **注 :** 入力にグループがない場合は、このパラメータに null を設定します。その場合は、データ全体が 1 つのグループであるとみなされます。
Filter Configuration	統合条件を定義します。この条件に基づいて、ジョブは各グループからのレコードを 1 つ以上保持します。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

出力列は入力列と同じです。出力に追加されるその他の列はありません。

Filter MapReduce ジョブの使用

1. AdvanceMatchFactory のインスタンスを、その静的メソッド getInstance() を使用して作成します。
2. Filter ジョブの入力と出力の詳細を指定します。以下の手順に従って、FilterDetail を指定する ProcessType のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) GroupbyOption のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

      GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

   b) FilterConfiguration のインスタンスを作成することによって、ジョブの統合ルールを生成します。このインスタンスの中で、ConsolidationCondition のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

      ConsolidationCondition の各インスタンスは、ConsolidationRule インスタンスとそれに対応する ConsolidationAction インスタンスを使用して定義されます。

      > **注：** ConsolidationRule の各インスタンスは、SimpleRule の 1 つのインスタンスによって定義するか、または、子の SimpleRule インスタンスとネストされた ConjoinedRule インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

   c) FilterDetail のインスタンスを作成します。JobConfig タイプのインスタンスと、上で作成した GroupbyOption インスタンスおよび FilterConfiguration インスタンスを、コンストラクタの引数として渡します。

      JobConfig パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

   d) inputPath インスタンスの FilterDetail フィールドを使用して、入力ファイルの詳細を設定します。

      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して FilePath のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して OrcFilePath のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して ParquetFilePath のインスタンスを作成します。

   e) FilterDetail インスタンスの outputPath フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath`のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `FilterDetail` フィールドを使用して、ジョブの名前を設定します。

g) `FilterDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. 先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出し、ジョブを作成します。ここで、上の `FilterDetail` のインスタンスを引数として渡します。
   `createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Filter Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Filter ジョブの入力と出力の詳細を指定します。以下の手順に従って、`FilterDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

   GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

   b) `FilterConfiguration` のインスタンスを作成することによって、ジョブの統合ルールを生成します。このインスタンスの中で、`ConsolidationCondition` のインスタンスを使用して統合条件を定義し、論理演算子を使用して条件を組み合わせます。

   `ConsolidationCondition` の各インスタンスは、`ConsolidationRule` インスタンスとそれに対応する `ConsolidationAction` インスタンスを使用して定義されます。

   > **注：** `ConsolidationRule` の各インスタンスは、`SimpleRule` の 1 つのインスタンスによって定義するか、または、子の `SimpleRule` インスタンスとネストされ

た `ConjoinedRule` インスタンスが、論理演算子で結合された階層を使用して定義できます。**列挙 JoinType**（405ページ）、および**列挙 Operation**（404ページ）を参照してください。

c) `FilterDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `FilterConfiguration` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `FilterDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `FilterDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `FilterDetail` フィールドを使用して、ジョブの名前を設定します。

g) `FilterDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `FilterDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Interflow ジョブ

Interflow Match は、2 つの入力レコード ストリーム内の類似するデータ レコード間でマッチを検出します。最初のレコード ストリームはサスペクト レコードのソースで、2 番目のストリームは候補レコードのソースです。

Interflow Match では、マッチ グループ条件 (マッチ キー等) を使用して、特定のサスペクト レコードと重複する可能性があるレコードのグループを識別します。

レポート

Interflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクト レコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Expressマッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。
AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。
UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATE_CANDIDATES	検出された重複候補の総数。

TOTAL_DUPLICATE_SCORE すべての重複の合計マッチ スコア。

API エンティティ

InterMatchDetail

目的
Interflow Match ジョブの詳細を指定します。

InterMatchComparisonOption

目的
Interflow Match ジョブを定義する際の比較オプション (サスペクト レコードを、すべての候補レコードと比較する必要があるか、または選択された候補レコードと比較するか) を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	*MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合は、Group-By オプションを作成するために次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。
Match Rule	親ルールと子ルールを、`MatchRule` オブジェクトの作成に必要な数だけ定義します。 詳細については、「**MatchRule**（36ページ）」を参照してください。

パラメータ	説明
Candidate File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の候補テキスト ファイルのパス。 **レコード区切り文字** 候補ファイル内で使用されるレコード区切り文字。 **Field Separator** 候補ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 候補ファイルのヘッダー フィールドの配列。 **Skip First Row** サスペクト ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 **重要:** サスペクト ファイルと候補ファイルは、同じファイル形式である必要があります。テキスト ファイルまたは ORC 形式のファイル。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Suspect File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上のサスペクト テキスト ファイルのパス。 **レコード区切り文字** サスペクト ファイル内で使用されるレコード区切り文字。 **Field Separator** サスペクト ファイルで、レコード内の連続する2つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** サスペクト ファイルのヘッダーフィールドの配列。 **Skip First Row** サスペクト ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注 :** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ）である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Match Key Settings	マッチングの実行に必要なマッチ キーの生成に適用する、列とアルゴリズムの組み合わせ。 **注 :** マッチ キーを 1 つだけ指定します。 **注 :** マッチ キー設定は、マッチングを実行する前にマッチ キーを生成する場合のみ、設定します。

パラメータ	説明
Job Name	ジョブの名前。
Express Match Column	レコードの Express マッチに使用する列名。
Setting Collection Number Zero to Unique Records	ユニーク レコードのコレクション番号を 0 (ゼロ) に設定する場合は、これを `true` にします。
Comparison Option	次の 2 つのオプションのいずれかを選択できます。 • Compare the Suspect record to all Candidate records: ユニーク レコードを出力に返す必要があるかどうかを指定します。 • Compare the Suspect record to the selected Candidate record only: 検索して返す重複レコードの最大数を指定します。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

入力列に加えて、Interflow Match ジョブの出力生成時に以下の列が追加されます。

列	説明	出力値
コレクション番号	重複レコードのコレクションを識別します。	とり得る値は、`0-0-1`、`0-0-2`、などです。
Express Match Identified	マッチの取得に Express マッチ キーが使われたかどうかを示します。	1. Express マッチ キーを使用して重複する候補レコードがマッチした場合は、出力値は `Y` になります。 2. 重複する候補レコードがマッチしたが、Express マッチ キーは使用されなかった場合は、出力値は空白になります。 3. Express マッチ キーを使用してユニークな候補レコードがマッチした場合は、出力値は `N`.になります。 4. Express マッチ キーを使用してサスペクト レコードがマッチした場合は、出力値は空白になります。

列	説明	出力値
Interflow Source Type	入力レコードがサスペクト レコードであるか、候補レコードであるかを示します。	値は、サスペクト レコードの場合は S、候補レコードの場合は C になります。
Match Record Type	コレクションに含まれるマッチ レコードのタイプを識別します。	とり得る値は、S (サスペクト レコード)、D (重複レコード)、および U (ユニーク レコード) です。
マッチ スコア	2 つのレコードの全体的なスコアを識別します。	とり得る値は、重複レコードとユニーク レコードの場合は 0 (ゼロ) ～ 100 で、0 は精度の低いマッチ、100 は非常に精度の高いマッチを意味します。 **注：** サスペクト レコードの場合、この値は 0 です。

Interflow Match MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Interflow Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`InterMatchDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

      GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

   b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。
   c) `InterMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

      `JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

   d) `candidateFilePath` インスタンスの `InterMatchDetail` フィールドを使用して、候補ファイルの詳細を設定します。

テキスト候補ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な候補ファイル情報を指定して`FilePath` のインスタンスを作成します。ORC 候補ファイルの場合、ORC 候補ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

e) `suspectFilePath` インスタンスの `InterMatchDetail` フィールドを使用して、サスペクト ファイルの詳細を設定します。

テキスト サスペクト ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細なサスペクトファイル情報を指定して`FilePath`のインスタンスを作成します。ORC サスペクト ファイルの場合、ORC サスペクト ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。PARQUET サスペクト ファイルの場合、PARQUET サスペクト ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

重要: サスペクト ファイルと候補ファイルは、同じファイル形式である必要があります。テキスト ファイルまたは ORC 形式のファイル。

f) `InterMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

g) `jobName` インスタンスの `InterMatchDetail` フィールドを使用して、ジョブの名前を設定します。

h) 必要に応じて、`expressMatchColumn` インスタンスの `InterMatchDetail` フィールドを使用して、Express マッチ列を設定します。

i) ユニーク レコードにコレクション番号 0 (ゼロ) を割り当てる場合は、`collectionNumberZerotoUniqueRecords` インスタンスの `InterMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

ユニーク レコードにコレクション番号 0 を割り当てたくない場合は、このフラグに `false` を設定します。

j) `comparisonOption` インスタンスの `InterMatchDetail` フィールドを使用して、比較オプションを設定します。このフィールドで、クラス **InterMatchComparisonOption** （67ページ） を使用して必要な値を設定することにより、以下の 2 つのオプションのいずれかを選択します。

- **[Compare the Suspect record to all Candidate records]**: ユニーク レコードを出力に返す必要があるかどうかを指定します。
- **[Compare the Suspect record to the selected Candidate record only]**: 検索して返す重複レコードの最大数を指定します。

k) `InterMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

l) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Interflow Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Interflow マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `InterMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `InterMatchDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Interflow Match Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Interflow Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`InterMatchDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

   GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。

c) `InterMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `candidateFilePath` インスタンスの `InterMatchDetail` フィールドを使用して、候補ファイルの詳細を設定します。

テキスト候補ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な候補ファイル情報を指定して`FilePath` のインスタンスを作成します。ORC 候補ファイルの場合、ORC 候補ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

e) `suspectFilePath` インスタンスの `InterMatchDetail` フィールドを使用して、サスペクト ファイルの詳細を設定します。

テキスト サスペクト ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細なサスペクト ファイル情報を指定して`FilePath` のインスタンスを作成します。ORC サスペクト ファイルの場合、ORC サスペクト ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。PARQUET サスペクト ファイルの場合、PARQUET サスペクト ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

重要: サスペクト ファイルと候補ファイルは、同じファイル形式である必要があります。テキスト ファイルまたは ORC 形式のファイル。

f) `InterMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

g) `jobName` インスタンスの `InterMatchDetail` フィールドを使用して、ジョブの名前を設定します。

h) 必要に応じて、`expressMatchColumn` インスタンスの `InterMatchDetail` フィールドを使用して、Express マッチ列を設定します。

i) ユニーク レコードにコレクション番号 0 (ゼロ) を割り当てる場合は、`collectionNumberZerotoUniqueRecords` インスタンスの `InterMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

ユニーク レコードにコレクション番号 0 を割り当てたくない場合は、このフラグに `false` を設定します。

j) `comparisonOption` インスタンスの `InterMatchDetail` フィールドを使用して、比較オプションを設定します。このフィールドで、クラス **InterMatchComparisonOption** （67ページ） を使用して必要な値を設定することにより、以下の 2 つのオプションのいずれかを選択します。

- **[Compare the Suspect record to all Candidate records]**: ユニーク レコードを出力に返す必要があるかどうかを指定します。
- **[Compare the Suspect record to the selected Candidate record only]**: 検索して返す重複レコードの最大数を指定します。

k) `InterMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

l) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Interflow Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Interflow マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `InterMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `InterMatchDetail` のインスタンスを引数として渡します。
`runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。
4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Intraflow ジョブ

Intraflow Match は、単一の入力ストリーム内の類似するデータ レコード間でマッチを検出します。データフローの別のコンポーネントで定義または作成したフィールドに基づいて、階層的なルールを作成できます。

レポート

Intraflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

INPUT_RECORDS	マッチング ソート実行前のマッチング ステージにおけるレコードの数。
DUPLICATE_RECORDS	マッチ グループ内の重複レコード (サスペクト レコードまたは候補レコード) の数。
UNIQUE_RECORDS	各マッチ グループで他のレコードにマッチしないサスペクトまたは候補レコードの数。 マッチ グループ内に 1 つしか存在していないレコードであれば、サスペクトは自動的にユニーク レコードとなります。
MATCH_GROUPS	(グループ化) マッチ キーでグループ化されたレコード。
DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクトレコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Express マッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。
AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
TOTAL_DUPLICATES	検出された重複の総数。
TOTAL_SCORE	すべての重複の合計マッチ スコア。

API エンティティ

IntraMatchDetail

目的

Intraflow Match ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	*MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合は、Group-By オプションを作成するために次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。
Match Rule	親ルールと子ルールを、`MatchRule` オブジェクトの作成に必要な数だけ定義します。 詳細については、「**MatchRule**（36ページ）」を参照してください。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ）である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Job Name	ジョブの名前。
Express Match Column	レコードの Express マッチに使用する列名。
Setting Collection Number Zero to Unique Records	ユニーク レコードのコレクション番号を 0 (ゼロ) に設定する場合は、これを `true` にします。

パラメータ	説明
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。
Match Key Settings	マッチングの実行に必要なマッチ キーの生成に適用する、列とアルゴリズムの組み合わせ。 **注 :** マッチ キーを 1 つだけ指定します。 **注 :** マッチ キー設定は、マッチングを実行する前にマッチ キーを生成する場合のみ、設定します。

出力列

入力列に加えて、Intraflow Match ジョブの出力生成時に以下の列が追加されます。

列	説明	出力値
コレクション番号	重複レコードのコレクションを識別します。	とり得る値は、`0-0-1`、`0-0-2`、などです。
Express Match Identified	マッチの取得に Express マッチ キーが使われたかどうかを示します。	1. Express マッチ キーを使用して重複する候補レコードがマッチした場合は、出力値は `Y` になります。 2. 重複する候補レコードがマッチしたが、Express マッチ キーは使用されなかった場合は、出力値は空白になります。 3. Express マッチ キーを使用してユニークな候補レコードがマッチした場合は、出力値は空白になります。 4. Express マッチ キーを使用してサスペクト レコードがマッチした場合は、出力値は空白になります。
Match Record Type	コレクションに含まれるマッチ レコードのタイプを識別します。	とり得る値は、`S` (サスペクト レコード)、`D` (重複レコード)、および `U` (ユニーク レコード) です。

列	説明	出力値
マッチ スコア	2 つのレコードの全体的なスコアを識別します。	とり得る値は、重複レコードとユニーク レコードの場合は 0 (ゼロ) ～ 100 で、0 は精度の低いマッチ、100 は非常に精度の高いマッチを意味します。 **注：** サスペクト レコードの場合、この値は 0 です。

Intraflow Match MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Intraflow Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`IntraMatchDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

   GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

   b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。
   c) `IntraMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

   `JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。
   d) `inputPath` インスタンスの `IntraMatchDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `IntraMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `IntraMatchDetail` フィールドを使用して、ジョブの名前を設定します。

g) 必要に応じて、`expressMatchColumn` インスタンスの `IntraMatchDetail` フィールドを使用して、Express マッチ列を設定します。

h) ユニーク レコードにコレクション番号 0 (ゼロ) を割り当てる場合は、`collectionNumberZerotoUniqueRecords` インスタンスの `IntraMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

ユニーク レコードにコレクション番号 0 を割り当てたくない場合は、このフラグに `false` を設定します。

i) `IntraMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

j) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Intraflow Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Intraflow マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `IntraMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `IntraMatchDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。

5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Intraflow Match Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Intraflow Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`IntraMatchDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。
   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

      GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

   b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。
   c) `IntraMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

      `JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

   d) `inputPath` インスタンスの `IntraMatchDetail` フィールドを使用して、入力ファイルの詳細を設定します。

      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   e) `IntraMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

      - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `IntraMatchDetail` フィールドを使用して、ジョブの名前を設定します。

g) 必要に応じて、`expressMatchColumn` インスタンスの `IntraMatchDetail` フィールドを使用して、Express マッチ列を設定します。

h) ユニーク レコードにコレクション番号 0 (ゼロ) を割り当てる場合は、`collectionNumberZerotoUniqueRecords` インスタンスの `IntraMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

ユニーク レコードにコレクション番号 0 を割り当てたくない場合は、このフラグに `false` を設定します。

i) `IntraMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

j) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Intraflow Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Intraflow マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `IntraMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `IntraMatchDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Match Key Generator ジョブ

Match Key Generator ジョブでは、マッチ キーが生成できます。

注： データのマッチ キーを生成するには、他のジョブを実行する前に、一度 Match Key Generator ジョブを実行する必要があります。

Match Key Generator は、レコードごとに非ユニーク キーを作成します。この非ユニーク キーは、潜在的な重複レコードのグループを特定するためにマッチング ステージで使用できます。マッチ キーを使用すると、レコードをマッチ キー別にグループ化し、各グループ内でのみレコードを比較できるので、マッチング プロセスが促進されます。

マッチ キーは、定義したルールを使用して作成され、入力フィールドから構成されます。指定する入力フィールドごとに、そのフィールドで実行されるアルゴリズムが選択されます。その後、各アルゴリズムの結果を連結して、単一のマッチ キー フィールドが作成されます。

マッチ キーの作成に加え、後のデータフローの Intraflow Match ステージまたは Interflow Match ステージで使用する Express マッチ キーも作成できます。

複数のマッチ キーおよび Express マッチ キーを作成できます。

例えば、次のような入力レコードがあり、

名 - Fred
姓 - Mertz
郵便番号 - 21114-1687
性別コード - M

次のようなレコードのデータを組み合わせてマッチ キーを生成するマッチ キー ルールを定義したとします。

入力フィールド	開始位置	長さ
郵便番号	1	5
郵便番号	7	4
姓	1	5
名	1	5
性別コード	1	1

次のようなキーになります。

211141687MertzFredM

API エンティティ

MatchKeyGeneratorDetail

目的
Match Key Generator ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注：** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Match Key Settings	マッチングの実行に必要なマッチ キーの生成に適用する、列とアルゴリズムの組み合わせ。 **注:** 少なくとも 1 つのマッチ キーを指定する必要があります。必要に応じて複数のマッチ キーを指定できます。
Job Name	ジョブの名前。

出力列

入力列に加えて、Match Key Generator ジョブの出力生成時に以下の列が追加されます。

列	説明	出力値
MatchKey	レコードを識別するために生成されるキー。	マッチ キーの生成に選択された列とアルゴリズムに基づいて生成されたキーです。 **注：** 出力で生成される、ユーザが名前を付けたマッチ キー列の数は、ジョブ設定によって異なります。

Match Key Generator MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Match Key Generator ジョブの入力と出力の詳細を指定します。以下の手順に従って、`MatchKeyGeneratorDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `MatchKeySettings` のインスタンスを作成および設定することによって、マッチングを実行するためのマッチ キー設定を指定します。詳細については、関連するコード サンプルを参照してください。
   b) `MatchKeyGeneratorDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `MatchKeySettings` インスタンスを、コンストラクタの引数として渡します。

      `JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。
   c) `inputPath` インスタンスの `MatchKeyGeneratorDetail` フィールドを使用して、入力ファイルの詳細を設定します。
      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

d) `MatchKeyGeneratorDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `jobName` インスタンスの `MatchKeyGeneratorDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `MatchKeyGeneratorDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。

Match Key Generator Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Match Key Generator ジョブの入力と出力の詳細を指定します。`ProcessType` を指定する `MatchKeyGeneratorDetail` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

   a) `MatchKeySettings` のインスタンスを作成および設定することによって、マッチングを実行するためのマッチ キー設定を指定します。詳細については、関連するコード サンプルを参照してください。

   b) `MatchKeyGeneratorDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `MatchKeySettings` インスタンスを、コンストラクタの引数として渡します。

   `JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

   c) `inputPath` インスタンスの `MatchKeyGeneratorDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。

- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

d) `MatchKeyGeneratorDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `jobName` インスタンスの `MatchKeyGeneratorDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `MatchKeyGeneratorDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

Transactional Match ジョブ

Transactional Match ジョブにより、サスペクト レコードをレコード グループに含まれる候補レコードに一致させて、重複を特定することができます。

Transactional Match は、重複を特定するため、サスペクト レコードと、あるグループのレコードを照合します。これらのレコードはまず、選択した列によりグループ化され、最初のレコードがサスペクト レコードとしてマークされます。グループの残りすべてのレコードは候補レコードと呼ばれ、サスペクト レコードと照合されます。

候補レコードが重複の場合は、コレクション番号が割り当てられ、そのマッチ レコード タイプに重複が設定され、その候補レコードが書き出されます。グループ内のマッチしない候補にはコレクション番号 0 が割り当てられ、そのラベルにユニークが設定され、その候補が書き出されます。

レポート

Transactional Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。
UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATES_SCORE	すべての重複の合計マッチ スコア。
TOTAL_DUPLICATES	検出された重複の総数。

API エンティティ

TransactionalMatchDetail

目的
Transactional Match ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Group-By Option	*MapReduce* ジョブの場合は、次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。 **Number of Reducer Tasks** レコードのグループ化に必要なリデューサー タスクの数。 *Spark* ジョブの場合は、Group-By オプションを作成するために次の引数を渡します。 **GroupBy Column** レコードのグループ化に使用する列の名前。

パラメータ	説明
Match Rule	親ルールと子ルールを、`MatchRule` オブジェクトの作成に必要な数だけ定義します。 詳細については、「**MatchRule**（36ページ）」を参照してください。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注：`FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。
Return Unique Candidates	ユニークな候補を出力として返さなければならないかどうかを示すフラグ。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

パラメータ	説明
Match Key Settings	マッチングの実行に必要なマッチ キーの生成に適用する、列とアルゴリズムの組み合わせ。 **注 :** マッチ キーを 1 つだけ指定します。 **注 :** マッチ キー設定は、マッチングを実行する前にマッチ キーを生成する場合のみ、設定します。

出力列

入力列に加えて、Transactional Match ジョブの出力生成時に以下の列が追加されます。

パラメータ	説明	出力値
Match Record Type	コレクションに含まれるマッチ レコードのタイプを識別します。	とり得る値は、S (サスペクト レコード)、D (重複レコード)、および U (ユニーク レコード) です。
マッチ スコア	2 つのレコードの全体的なスコアを識別します。	とり得る値は、重複レコードとユニーク レコードの場合は 0 (ゼロ) ～ 100 で、0 は精度の低いマッチ、100 は非常に精度の高いマッチを意味します。 **注 :** サスペクト レコードの場合、この値は 0 です。
Has Duplicates	サスペクト レコードに重複があるかどうかを示します。	サスペクト レコードの場合、出力値は次のいずれかです。 • Y (重複が存在します) • N (重複は存在しません) 重複レコードの場合、出力値は D です。 ユニーク レコードの場合、出力値は U です。

Transactional Match MapReduce ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Transactional Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`TransactionalMatchDetail` を指定する `ProcessType` のインスタンスを作成することに

よって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。

a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

GroupbyMROption（36ページ） のインスタンスを使用して、必要な Group-By 列とリデューサー数を指定します。

b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。

c) `TransactionalMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `TransactionalMatchDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `TransactionalMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `TransactionalMatchDetail` フィールドを使用して、ジョブの名前を設定します。

g) ユニークな候補レコードを出力に返す場合は、`returnUniqueCandidates` インスタンスの `TransactionalMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

h) `TransactionalMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

i) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Transactional Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Transactional マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `TransactionalMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. MapReduce ジョブを作成するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `TransactionalMatchDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Transactional Match Spark ジョブの使用

1. `AdvanceMatchFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Transactional Match ジョブの入力と出力の詳細を指定します。以下の手順に従って、`TransactionalMatchDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

   a) `GroupbyOption` のインスタンスを作成することによって、レコードのグループ化に使用する列を指定します。

   GroupbySparkOption（36ページ） のインスタンスを使用して、Group-By 列を指定します。

   b) `MatchRule` のインスタンスを作成することによって、ジョブのマッチング ルールを生成します。

   c) `TransactionalMatchDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `GroupbyOption` インスタンスおよび `MatchRule` インスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `TransactionalMatchDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `TransactionalMatchDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `TransactionalMatchDetail` フィールドを使用して、ジョブの名前を設定します。

g) ユニークな候補レコードを出力に返す場合は、`returnUniqueCandidates` インスタンスの `TransactionalMatchDetail` フラグに `true` を設定します。デフォルトは `true` です。

h) `TransactionalMatchDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

i) 入力データにマッチ キーがない場合は、マッチ キー設定を指定して、Transactional Match ジョブを実行する前にまず、Match Key Generator ジョブを実行してマッチ キーを生成する必要があります。

入力データのマッチ キーを生成するには、`MatchKeySettings` のインスタンスを作成および設定することによってマッチ キー設定を指定し、Transactional マッチングを実行する前にマッチ キーを生成します。`matchKeySettings` インスタンスの `TransactionalMatchDetail` フィールドを使用して、このインスタンスを設定します。

注： マッチ キー設定方法については、コード サンプルを参照してください。

3. Spark ジョブを作成して実行するには、先ほど作成した `AdvanceMatchFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `TransactionalMatchDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。
4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Data Integration モジュールのジョブ

コモンモジュール API

DataIntegrationFactory

目的
Data Integration モジュールのジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

カスタム Groovy スクリプト

カスタム Groovy スクリプト ジョブ

このジョブは、定義された Groovy スクリプトに基づいて入力フィールドを変換します。

API エンティティ

CustomGroovyScriptConfiguration

これらの詳細を指定するには、次の手順を実行します。

- Groovy スクリプト ファイル
- 入力フィールド
- 出力フィールド

CustomGroovyScriptDetail

目的

`Custom Groovy Script Detail` ジョブに対して以下を指定します。

- 入力ファイル
- 出力ファイル
- ジョブの名前
- `M/d/yy` としての日付パターン
- `M/d/yy h:mm a` としての日時パターン
- `h:mm a` としての時刻パターン

入力パラメータ

パラメータ	説明
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注: `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

出力列は入力列と同じで、Groovy スクリプトによって変換された値が格納されます。

Groovy スクリプト MapReduce ジョブの使用

1. 静的メソッド `getInstance()` を使用して、`DataIntegrationFactory` のインスタンスを作成します。

2. Groovy スクリプト ジョブの入力と出力の詳細を指定します。以下の手順に従って、`ProcessType` を指定する `CustomGroovyScriptDetail` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。`CustomGroovyScriptDetail` インスタンスを作成および設定するには、次の手順を実行します。

   a) `ProcessType` に **MRProcessType**（34ページ） を指定して、`CustomGroovyScriptDetail` のインスタンスを作成します。このインスタンスに次の詳細を設定します。

   - 入力ファイル: `inputPath` フィールドを使用

     注：

     - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
     - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
     - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   - 出力ファイル: `outputPath` フィールドを使用

     注：

     - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
     - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
     - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   - ジョブの名前: `jobName` フィールドを使用
   - 日付パターン: `M/d/yy`
   - 日時パターン: `M/d/yy h:mm a`
   - 時刻パターン: `h:mm a`

   b) `CustomGroovyScriptConfiguration` のインスタンスを作成し、次の詳細を設定します。

   - The groovyScriptFile
   - InputFields
   - OutputFields

c) `getScriptTransformerConfiguration()` メソッドを使用して `configuration` を作成します。このメソッドは、上で作成および設定した `CustomGroovyScriptConfiguration` のインスタンスのリストを呼び出します。

3. MapReduce ジョブを作成するには、先ほど作成した `DataIntegrationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `CustomGroovyScriptDetail` のインスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。

Groovy スクリプト Spark ジョブの使用

1. 静的メソッド `getInstance()` を使用して、`DataIntegrationFactory` のインスタンスを作成します。
2. Groovy スクリプト ジョブの入力と出力の詳細を指定します。以下の手順に従って、`ProcessType` を指定する `CustomGroovyScriptDetail` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ）を使用する必要があります。`CustomGroovyScriptDetail` インスタンスを作成および設定するには、次の手順を実行します。

   a) `ProcessType` に **SparkProcessType**（34ページ） を指定して、`CustomGroovyScriptDetail` のインスタンスを作成します。このインスタンスに次の詳細を設定します。

   - 入力ファイル: `inputPath` フィールドを使用

     注：

     - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath`のインスタンスを作成します。
     - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
     - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   - 出力ファイル: `outputPath` フィールドを使用

     注：

     - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath`のインスタンスを作成します。

- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

- ジョブの名前: `jobName` フィールドを使用
- 日付パターン: `M/d/yy`
- 日時パターン: `M/d/yy h:mm a`
- 時刻パターン: `h:mm a`

b) `CustomGroovyScriptConfiguration` のインスタンスを作成し、次の詳細を設定します。

- The groovyScriptFile
- InputFields
- OutputFields

c) `getScriptTransformerConfiguration()` メソッドを使用して `configuration` を作成します。このメソッドは、上で作成および設定した `CustomGroovyScriptConfiguration` のインスタンスのリストを呼び出します。

3. Spark ジョブを作成するには、先ほど作成した `DataIntegrationFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、`JoinDetail` インスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを作成し、`map` インスタンスの `ControlledJob` を返します。

Joiner

Joiner ジョブ

このジョブは、SQL 方式の JOIN 操作を実行し、複数のファイルのレコードを結合します。

API エンティティ

JoinDetail

目的

このクラスは、ファイルの入力パス、結合のタイプ (Inner、LeftOuter、Full)、結合される列、ジョブの出力パスなど、結合ジョブの詳細を指定します。

入力パラメータ

パラメータ	説明
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注: `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

結合された列が出力列に表示されます。

Joiner MapReduce ジョブの使用

1. 静的メソッド `getInstance()` を使用して、`DataIntegrationFactory` のインスタンスを作成します。

2. `JoinDetail` インスタンスでジョブの入力と出力の詳細を指定し、`ProcessType` を **MRProcessType**（34ページ） に指定します。`JoinDetail` インスタンスを作成および設定するには、次の手順を実行します。

   1. `ProcessType` を **MRProcessType**（34ページ） に指定し、デフォルトの設定を使用して `JoinDetail` のインスタンスを作成します。
   2. `FilePath` の別々のインスタンスを作成し、インスタンスごとに、`RecordSeparator` (**列挙 RecordSeparator**（404ページ） を使用)、`fieldSeperator`、`textQualifier`、`fileHeader` (最初の行をスキップするかどうかを指定) の各入力ファイルの詳細を設定します。

      注：

      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   3. 上の手順で作成した `JoinDetail` インスタンスで、以下の詳細を設定します。

      - `InputPaths`: 上で作成して設定した `FilePath` のインスタンスを渡します。
      - `LeftInput`: 結合操作の左側の入力を指定します。
      - `JobName`: ジョブの名前。
      - `JoinType`: **列挙 JoinDetail.JoinType**（405ページ） を使用して結合タイプを定義します。
      - `JoinColumns`:結合される入力列を指定します。値はカンマで区切って指定する必要があります。
      - `OutputPath`: `setOutputPath` メソッドを使用してジョブの出力パスを設定し、ファイルを上書きするかどうか、およびヘッダーを作成するかどうかを指定します。

3. `MapReduce` ジョブを作成するには、先ほど作成した `DataIntegrationFactory` のインスタンスを使用して、そのメソッド `createJob()` を呼び出します。ここで、`JoinDetail` インスタンスを引数として渡します。

   `createJob()` メソッドはジョブを作成し、`List` インスタンスの `ControlledJob` を返します。

4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。

Joiner Spark ジョブの使用

1. 静的メソッド `getInstance()` を使用して、`DataIntegrationFactory` のインスタンスを作成します。
2. `JoinDetail` インスタンスでジョブの入力と出力の詳細を指定し、`ProcessType` を **SparkProcessType**（34ページ） に指定します。`JoinDetail` インスタンスを作成および設定するには、次の手順を実行します。
   1. ProcessType を **SparkProcessType**（34ページ） に指定し、デフォルトの設定を使用して、JoinDetail のインスタンスを作成します。
   2. `FilePath` の別々のインスタンスを作成し、インスタンスごとに、`RecordSeparator` (**列挙 RecordSeparator**（404ページ） を使用)、`fieldSeperator`、`textQualifier`、`fileHeader` (最初の行をスキップするかどうかを指定) の各入力ファイルの詳細を設定します。

      注：

      - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
      - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
      - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   3. 上の手順で作成した `JoinDetail` インスタンスで、以下の詳細を設定します。
      - `InputPaths`: 上で作成して設定した `FilePath` のインスタンスを渡します。
      - `LeftInput`: 結合操作の左側の入力を指定します。
      - `JobName`: ジョブの名前。
      - `JoinType`: **列挙 JoinDetail.JoinType**（405ページ） を使用して結合タイプを定義します。
      - `JoinColumns`:結合される入力列を指定します。値はカンマで区切って指定する必要があります。
      - `OutputPath`: `setOutputPath` メソッドを使用してジョブの出力パスを設定し、ファイルを上書きするかどうか、およびヘッダーを作成するかどうかを指定します。
3. Spark ジョブを作成するには、先ほど作成した `DataIntegrationFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、`JoinDetail` インスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを作成し、`map` インスタンスの `ControlledJob` を返します。

Data Normalization モジュールのジョブ

コモンモジュール API

DataNormalizationDetail<T extends ProcessType>

目的
Data Normalization モジュールのジョブの詳細を指定します。

DataNormalizationFactory

目的
Data Normalization モジュールのジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

Advanced Transformer

Advanced Transformer ジョブは、テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。このコンポーネントは、特定の語、あるいは語の右側または左側から指定した数だけ単語を抽出します。抽出データと非抽出データを既存のフィールドまたは新しいフィールドに配置できます。

例えば、次の住所フィールドから一組の情報を抽出し、その情報を別のフィールドに配置するとします。

2300 BIRCH RD STE 100

これを行うには、語 STE と語 STE の右側にあるすべての単語を抽出する Advanced Transformer を作成することができます、分離されるフィールド：

2300 BIRCH RD

Advanced Transformer ジョブ

Advanced Transformer ジョブは、テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。このコンポーネントは、特定の語、あるいは語の右側または左側から指定した数だけ単語を抽出します。

API エンティティ

AbstractAdvancedTransformerRules

目的

Advanced Transformer ジョブのルールを指定する親クラス。

AdvancedTransformerDetail

目的

Advanced Transformer ジョブの詳細を指定します。

AdvancedTransformerConfiguration

目的

テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。

RegularExpressionExtraction

目的

正規表現を用いてデータを抽出するルールを指定します。

RegularExpressionGroupItem

目的

親の正規表現の一部を指定します。親の正規表現の各部分は、異なる出力フィールドに格納できます。

TableDataExtraction

目的

テーブルからデータを抽出するためのルールを定義します。

入力パラメータ

パラメータ	説明
Advanced Transformer Configuration	テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。 特定の語や、語の右側または左側から指定した数の単語を抽出できます。抽出データと非抽出データは、既存のフィールドまたは新しいフィールドに配置されます。 Advanced Transformer ルールは、`AdvancedTransformerConfiguration` タイプのインスタンスを用いて定義できます。このインスタンスは、`TableDataExtraction` または `RegularExpressionExtraction` のインスタンスである必要があります。
リファレンス データ パス	リファレンス データ パスの詳細を指定します。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

入力列に加えて、Advanced Transformer ジョブの出力生成時に以下の列が追加されます。

列	説明	出力値
Non-Extracted Data	この出力列は、入力には存在しない新しい列名が、Non-Extracted Data 列として指定された場合に追加されます。 列名は、ユーザが入力したとおりです。 **注：** Non-Extracted Data 列に対し、既存のソース列を選択するか、新しい列名を入力できます。	指定された語に基づく、各レコードの非抽出データ。
Extracted Data	この出力列は、入力には存在しない新しい列名が、Extracted Data 列として指定された場合に追加されます。 列名は、ユーザが入力したとおりです。 **注：** Extracted Data 列に対し、既存のソース列を選択するか、新しい列名を入力できます。	指定された語に基づく、各レコードの抽出データ。
Advanced Transform Term Identified	語が識別されたかどうかを示します。	値は `Yes` または `No` です。

Advanced Transformer MapReduce ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Advanced Transformer ジョブの入力と出力の詳細を指定します。以下の手順に従って、`AdvancedTransformerDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `AdvancedTransformerConfiguration` のインスタンスを作成することによって、Advanced Transformer ルールを設定します。
   このインスタンス内に、`AbstractAdvancedTransformerRules` タイプのインスタンスを追加します。この `AbstractAdvancedTransformerRules` インスタンスは、必要な Advanced Transformer ルール カテゴリに応じて `TableDataExtraction` または `RegularExpressionExtraction` のいずれかのクラスを用いて定義する必要があります。

b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。

c) `AdvancedTransformerDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `AdvancedTransformerConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `AdvancedTransformerDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `AdvancedTransformerDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `AdvancedTransformerDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `AdvancedTransformerDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Advanced Transformer Spark ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Advanced Transformer ジョブの入力と出力の詳細を指定します。以下の手順に従って、`AdvancedTransformerDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。
   a) `AdvancedTransformerConfiguration` のインスタンスを作成することによって、Advanced Transformer ルールを設定します。

   このインスタンス内に、`AbstractAdvancedTransformerRules` タイプのインスタンスを追加します。この `AbstractAdvancedTransformerRules` インスタンスは、必要な Advanced Transformer ルール カテゴリに応じて `TableDataExtraction` または `RegularExpressionExtraction` のいずれかのクラスを用いて定義する必要があります。

   b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。

   c) `AdvancedTransformerDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `AdvancedTransformerConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

   `JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

   d) `inputPath` インスタンスの `AdvancedTransformerDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

   e) `AdvancedTransformerDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `AdvancedTransformerDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `AdvancedTransformerDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Open Parser

Open Parser は、世界のさまざまなカルチャーからの入力データを、シンプルかつ強力なパーシング グラマーでパースします。このグラマーを使用すると、ドメイン パターンを表す一連の式を入力データのパース用に定義できます。また、Open Parser は、統計データを収集してパーシング マッチにスコアを付けるため、パーシング グラマーの効果を調べるのも容易です。

Open Parser を使用して、次のことができます。

- ドメインエディタで定義したドメイン固有およびカルチャー固有のパーシング グラマーを使用して、入力データをパースします。
- ドメインエディタで使用できるものと同じ、シンプルかつ強力なパーシング グラマーを使用して Open Parser で定義した、ドメインに依存しないパーシング グラマーを使用して、入力データをパースします。
- データフロー オプションで定義したドメインに依存しないパーシング グラマーを実行時に使用して、入力データをパースします。
- ターゲット入力データファイルを使用してジョブを実行する前に、パーシング グラマーをプレビューして、サンプル入力データがどのようにパースされるかをテストします。
- パーシング グラマー結果をトレースして、定義した式にトークンがどれくらいマッチしたか、またはマッチしなかったかを確認し、マッチング処理に関する理解を深めます。

Open Parser ジョブ

Open Parser ジョブは、定義されたパーシング グラマーに従って入力データの文字列をパースします。

API エンティティ

OpenParserDetail

目的

Open Parser ジョブの詳細を指定します。

OpenParserConfiguration

目的

入力テキストの文字列を、該当のコンポーネントに分割します。

入力パラメータ

パラメータ	説明
Open Parser 環境設定	入力文字列を構成要素となる該当のコンポーネントにパースします。
リファレンス データ パス	リファレンス データ パスの詳細を指定します。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

このジョブは、入力データの文字列を、パーシング グラマーの定義に基づいて該当の構成要素に分割し、出力列に表示します。また、`Yes` または `No` を表示して、レコードがパースされたかどうかを示します。

Open Parser MapReduce ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Open Parser ジョブの入力と出力の詳細を指定します。以下の手順に従って、`OpenParserDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `OpenParserConfiguration` のインスタンスを作成することによって、パーシング ルールを設定します。このインスタンスに、グラマー ファイル パスを設定します。
   b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。列挙 **ReferenceDataPathLocation**（404ページ）を参照してください。

c) `OpenParserDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `OpenParserConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) 作成した `OpenParserDetail` のインスタンス内で、`OpenParserDetail` インスタンスの `inputPath` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `OpenParserDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `OpenParserDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `OpenParserDetail` のインスタンスを引数として渡します。
`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。

Open Parser Spark ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Open Parser ジョブの入力と出力の詳細を指定します。以下の手順に従って、`OpenParserDetail` を指定する `ProcessType` のインスタンスを作成することによって、

これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

a) `OpenParserConfiguration` のインスタンスを作成することによって、パーシング ルールを設定します。このインスタンスに、グラマー ファイル パスを設定します。

b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。

a) `OpenParserDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `OpenParserConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

a) 作成した `OpenParserDetail` のインスタンス内で、`OpenParserDetail` インスタンスの `inputPath` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

b) `OpenParserDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

c) `jobName` インスタンスの `OpenParserDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `OpenParserDetail` のインスタンスを引数として渡します。

`runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

Table Lookup

Table Lookup ステージは、語とその語の妥当性確認済みの形式とを照合して正規化し、標準バージョンを適用します。この評価は、正規化する語をテーブルから検索して実行されます。

Table Lookup ジョブ

Table Lookup ジョブは、語とその語の妥当性確認済みの形式とを照合して正規化し、標準バージョンを適用します。

API エンティティ

AbstractTableLookupRule

目的

Table Lookup で使用するルールを指定します。

分類

目的

Table Lookup ジョブ用の分類 (Categorize) ルールを指定します。

特定

目的

Table Lookup ジョブ用の特定 (Identify) ルールを指定します。

正規化

目的

Table Lookup ジョブ用の正規化 (Standardize) ルールを指定します。

TableLookupDetail

目的

Table Lookup ジョブの詳細を指定します。

TableLookupConfiguration

目的

語を、その語の妥当性確認済みの形式と照合して正規化し、正規化したバージョンをすべてのレコードに適用します。

入力パラメータ

パラメータ	説明
Table Lookup Configuration	語を、その語の妥当性確認済みの形式と照合して正規化し、正規化したバージョンをすべてのレコードに適用します。 ルールのタイプは `Standardize`、`Categorize`、または `Identify` とすることができます。
リファレンス データ パス	リファレンス データ パスの詳細を指定します。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

入力列に加えて、Table Lookup ジョブの出力生成時に以下の列が追加されます。

列	説明	出力値
デスティネーション	ルール オプションが `Standardize` または `Categorize` の場合、この出力列は、入力には存在しない新しい列名が、Destination 列として指定された場合に追加されます。 列名は、ユーザが入力したとおりです。 **注：** Destination 列に対し、既存のソース列を選択するか、新しい列名を入力できます。	テーブル データに対してマッチングされる、ソース列の正規化された値。
Standardization Term Identified	正規化された語が識別されたかどうかを示します。	値は `Yes` または `No` です。

Table Lookup MapReduce ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Table Lookup ジョブの入力と出力の詳細を指定します。以下の手順に従って、`TableLookupDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `TableLookupConfiguration` のインスタンスを作成することによって、Table Lookup ルールを設定します。
   このインスタンス内に、`AbstractTableLookupRule` タイプのインスタンスを追加します。この `AbstractTableLookupRule` インスタンスは、必要な Table Lookup ルール カテゴリに応じて `Standardize`,`Categorize` または `Identify` のいずれかのクラスを用いて定義する必要があります。
   b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。
   c) `TableLookupDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `TableLookupConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。
   `JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `TableLookupDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `TableLookupDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `TableLookupDetail` フィールドを使用して、ジョブの名前を設定します。

g) `TableLookupDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. MapReduce ジョブを作成するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `TableLookupDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Table Lookup Spark ジョブの使用

1. `DataNormalizationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Table Lookup ジョブの入力と出力の詳細を指定します。以下の手順に従って、`TableLookupDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

a) `TableLookupConfiguration` のインスタンスを作成することによって、Table Lookup ルールを設定します。

このインスタンス内に、`AbstractTableLookupRule` タイプのインスタンスを追加します。この `AbstractTableLookupRule` インスタンスは、必要な Table Lookup ルールカテゴリに応じて `Standardize`,`Categorize` または `Identify` のいずれかのクラスを用いて定義する必要があります。

b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンスデータパスと場所のタイプの詳細を設定します。列挙 **ReferenceDataPathLocation**（404ページ）を参照してください。

c) `TableLookupDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `TableLookupConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `TableLookupDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して `FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `TableLookupDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して `FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `TableLookupDetail` フィールドを使用して、ジョブの名前を設定します。

g) `TableLookupDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. Spark ジョブを作成して実行するには、先ほど作成した `DataNormalizationFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `TableLookupDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。
4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Global Addressing モジュールのジョブ

Global Address Validation

Global Address Validation は、高度な住所の正規化および検証の機能を提供します。Global Address Validation は Global Addressing モジュールの一部です。

API エンティティ

AddressValidationDetail<T extends ProcessType>

目的
Global Address Validation ジョブの詳細を指定します。

AddressValidationEngineConfiguration

目的
Global Address Validation ジョブの `ProductDatabaseInfoList` を設定します。

これらは 1 回限りの設定です。

AddressValidationFactory

目的
Global Address Validation ジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

このインスタンスは、レポート カウンタを生成するために使われます。

ProductDatabaseInfo

目的
Global Address Validation ジョブを作成および実行するためのデータベース パス、国コード、プロセス タイプを設定します。

AddressValidationInputOption

目的
Global Address Validation ジョブを作成および実行するために入力の設定を構成する。これはルール設定であり、さまざまなオプションがあります。これらの設定はジョブごとに異なります。

入力パラメータ

パラメータ	説明
Address Validation のエンジン設定	製品データベース情報のリストを設定します。

パラメータ	説明
Address Validation の入力オプション	入力設定を指定します。 1. 出力の大文字と小文字の区別 2. マッチ モード 3. デフォルトの国 4. 結果の最大数 5. 入力アドレスを返す 6. Return Parsed Address 7. 精度コードを返す 8. マッチ スコアを返す 9. 住所番号が必ず一致 10. 通りが一致 11. 都市が一致 12. 地方が一致 13. 州が一致 14. 州/省が一致 15. 郵便番号が一致 16. 複数の一致を保持 17. 都市名よりも郵便番号を優先 18. 都市へのフォールバック 19. 郵便番号へのフォールバック 20. バリデーション レベル
製品データベース情報	以下を設定します。 1. 国コード 2. データベース パス 3. プロセス タイプ
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注 : `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

出力列

1. AddressBlock1-2

2. ApartmentLabel
3. ApartmentNumber
4. 建物
5. City
6. City.Matched
7. CitySubdivision
8. CitySubdivision.Matched
9. Confidence
10. Country
11. Country specific fields
12. FirmName
13. HouseNumber
14. Housenumber.Matched
15. LeadingDirectional
16. MatchOnAllStreetFields
17. MatchOnStreetDirectional
18. MultimatchCount
19. PostalCode
20. PostalCode.AddOn
21. Postalcode.Matched
22. Principality
23. ProcessedBy
24. StateProvince
25. StateProvince.Matched
26. StateProvinceSubdivision
27. StateProvinceSubdivision.Matched
28. StreetName
29. StreetName.Matched
30. StreetType
31. StreetType.Matched
32. TrailingDirectional
33. Vendor Code

注： フィールドの説明については、Spectrum™ Technology Platform の『*Addressing ガイド*』のトピック「*Global Address Validation*」を参照してください。

国固有の詳細情報 - 米国

米国の住所に対しては、以下の追加の詳細情報が表示されます。詳細については、Spectrum™ Technology Platform の『*Addressing ガイド*』の**レポート**を参照してください。

- USA.AddressLocation

- USA.AdvancedBarcode
- USA.Apartment1
- USA.Apartment2
- USA.BCCheckDigit
- USA.CarrierRouteCode
- USA.CASSAddressLine1
- USA.CASSAddressLine2
- USA.CASSCityName
- USA.DefaultMatch
- USA.DPV
- USA.DPV.CMRA
- USA.CongressionalDistrict
- USA.DPV.FalsePositive
- USA.DPV.Footnote
- USA.DPV.NoStat
- USA.DPV.PBSAFound
- USA.DPV.Vacant
- USA.FIPSCountyNumber
- USA.FiveDigitBarcode
- USA.FullCityName
- USA.LACS
- USA.LACS.Indicator
- USA.LACS.ReturnCode
- USA.LACS.SeedHit
- USA.LOTCode
- USA.LOTSequence
- USA.MatchLevel
- USA.POBoxOnly
- USA.PostalBarcode
- USA.PreferredCityName
- USA.PreferredState
- USA.RDI
- USA.Status
- USA.Status.Code
- USA.Status.Description
- USA.SuiteLink.Fidelity
- USA.SuiteLink.MatchCode
- USA.ZIPValid
- USA.ZIP4Valid

Global Address Validation MapReduce ジョブの使用

1. `AddressValidationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `AddressValidationDetail` のインスタンスを作成し、Global Address Validation ジョブの入力と出力の詳細を指定します。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `productDatabaseInfo` のインスタンスを作成し、次の詳細を設定します。
      1. ReferenceDataPath: 列挙 **ReferenceDataPathLocation**（404ページ） を使用します。
      2. CountryCode: 列挙 **CountryCodes**（407ページ） を使用します。
      3. ProcessType: **Enum AddressValidationProcessType**（415ページ） を使用します。
   b) 配列リスト クラス `ProductDatabaseInfoList` を作成し、add() メソッドを使用して `ProductDatabaseInfo` を挿入します。
   c) `AddressValidationEngineConfiguration` のインスタンスを作成し、このインスタンス内で `ProductDatabaseInfoList` を設定します。
   d) `AddressValidationInputOption` のインスタンスを作成し、この新しいインスタンスに次の詳細を設定します。

      注：次の列挙体を使用します: **Enum AddressValidationInputOption.MatchMode**（415ページ）、列挙 **CountryCodes**（407ページ）、列挙 **Casing**（409ページ）。

      - 大文字と小文字の区別
      - MatchMode
      - DefaultCountry
      - MaximumResults
      - ReturnInputAddress
      - ReturnParsedAddress
      - ReturnPrecisionCode
      - ReturnMatchScore
      - MustMatchAddressNumber
      - MustMatchStreet
      - MustMatchCity
      - MustMatchLocality
      - MustMatchState
      - MustMatchStateProvince
      - MustMatchPostCode
      - KeepMultiMatch

- PreferPostalOverCity
- CityFallback
- PostalFallback
- ValidationLevel

e) `AddressValidationDetail` のインスタンスを作成します。ジョブの環境設定と、上で作成した `addressValidationEngineConfiguration` と `inputOption` のインスタンスを引数として渡します。このインスタンスに次の詳細を設定します。

> **注：** `Config` パラメータは、**MRJobConfig**（32ページ） タイプ (MR ジョブ) および **SparkJobConfig**（33ページ） タイプ (Spark ジョブ) のインスタンスである必要があります。

1. `inputPath` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` フィールドを使用して、ジョブの名前を設定します。
4. `compressOutput` フラグを false に設定して、ジョブの出力が圧縮されないようにします。

3. MapReduce ジョブを作成するには、先ほど作成した `AddressValidationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `AddressValidationDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `AddressValidationFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Global Address Validation Spark ジョブの使用

1. `AddressValidationFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `AddressValidationDetail` のインスタンスを作成し、Global Address Validation ジョブの入力と出力の詳細を指定します。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `productDatabaseInfo` のインスタンスを作成し、次の詳細を設定します。
      1. ReferenceDataPath: 列挙 **ReferenceDataPathLocation**（404ページ） を使用します。
      2. CountryCode: 列挙 **CountryCodes**（407ページ） を使用します。
      3. ProcessType: **Enum AddressValidationProcessType**（415ページ） を使用します。

   b) 配列リスト クラス `ProductDatabaseInfoList` を作成し、add() メソッドを使用して `ProductDatabaseInfo` を挿入します。
   c) `AddressValidationEngineConfiguration` のインスタンスを作成し、このインスタンス内で `ProductDatabaseInfoList` を設定します。
   d) `AddressValidationInputOption` のインスタンスを作成し、この新しいインスタンスに次の詳細を設定します。

      注：次の列挙体を使用します: **Enum AddressValidationInputOption.MatchMode**（415ページ）、列挙 **CountryCodes**（407ページ）、列挙 **Casing**（409ページ）。

      - 大文字と小文字の区別
      - MatchMode
      - DefaultCountry
      - MaximumResults
      - ReturnInputAddress
      - ReturnParsedAddress

- ReturnPrecisionCode
- ReturnMatchScore
- MustMatchAddressNumber
- MustMatchStreet
- MustMatchCity
- MustMatchLocality
- MustMatchState
- MustMatchStateProvince
- MustMatchPostCode
- KeepMultiMatch
- PreferPostalOverCity
- CityFallback
- PostalFallback
- ValidationLevel

e) `AddressValidationDetail` のインスタンスを作成します。ジョブの環境設定と、上で作成した `addressValidationEngineConfiguration` と `inputOption` のインスタンスを引数として渡します。このインスタンスに次の詳細を設定します。

> **注：** `Config` パラメータは、**MRJobConfig**（32ページ） タイプ (MR ジョブ) および **SparkJobConfig**（33ページ） タイプ (Spark ジョブ) のインスタンスである必要があります。

1. `inputPath` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` フィールドを使用して、ジョブの名前を設定します。
4. `compressOutput` フラグを false に設定して、ジョブの出力が圧縮されないようにします。

3. Spark ジョブを作成するには、先ほど作成した `AddressValidationFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `AddressValidationDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。
4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Universal Addressing モジュールのジョブ

コモンモジュール APIs

UniversalAddressingDetail<T extends ProcessType>

目的
Universal Addressing モジュールのジョブの詳細を指定します。

UniversalAddressingFactory

目的
Universal Addressing モジュールのジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

Validate Address

Validate Address

Validate Address は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州または省名など、欠落している郵便情報を追加します。

> **注：** 現在、Validate Address では米国住所のみがサポートされています。

Validate Address は、住所の妥当性確認の有無、返された住所の確信レベル、住所の妥当性が確認できなかった場合の理由など、検証処理に関する結果インジケータも返します。

Validate Address は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを **Universal Addressing** モジュールの各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて*正規化*します。データベースにマッチしなかった場合、Validate Address は、オプションで入力住所の*書式を整えます*。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。

> **注：** Validate Address は CASS 認定™処理をサポートしており、これにより USPS® の郵便料金値引きのための条件を揃えることができます。

API エンティティ

UAMAddressingDetail<T extends ProcessType>

目的
Validate Address ジョブの詳細を指定します。

UniversalAddressEngineConfiguration

目的
Validate Address ジョブの作成および実行に必要な*リファレンス データ パス*や *COBOL ランタイム パス*等の各種設定を行います。

これらは 1 回限りの設定です。

UAMAddressingFactory

目的
Validate Address ジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

このインスタンスは、レポート カウンタと CASS レポートを生成するために使われます。

UniversalAddressGeneralConfiguration

目的

Validate Address ジョブを作成および実行するために必要な JVM 構成を設定します。

UniversalAddressValidateInputConfiguration

目的

Validate Address ジョブを作成および実行するために入力の設定を構成する。これはルール設定であり、さまざまなオプションがあります。これらの設定はジョブごとに異なります。

入力パラメータ

パラメータ	説明
Universal Address エンジン設定	各種ジョブ実行設定を指定します。 1. DPV データベース パス 2. Suite Link DB パス 3. EWS データベース パス 4. RDI データベース パス 5. Lacs データベース パス 6. リファレンス データ パス 7. COBOL ランタイム パス 8. モジュール ディレクトリ

パラメータ	説明
Universal Address 検証入力設定	

パラメータ	説明
	入力設定を指定します。 1. 標準住所を出力 2. 住所要素を出力 3. 郵便データを出力 4. パース済み入力を出力 5. 出力住所ブロック 6. 検索失敗時に出力をフォーマット 7. 出力の大文字と小文字の区別 8. 郵便番号区切り文字を出力 9. 多国籍文字を出力 10. DPV を実行 11. RDI を実行 12. ESM を実行 13. ASM を実行 14. EWS を実行 15. LACS Link を実行 16. LOT を実行 17. CMRA との一致をマッチとみなさない 18. 企業を抽出 19. 都市部を抽出 20. レポート 3553 を出力 21. レポート SERP を出力 22. レポート サマリを出力 23. CASS 詳細を出力 24. フィールドレベルのリターン コードを出力 25. 複数一致を保持 26. 結果の最大数 27. 標準住所フォーマット 28. 標準住所 PMB 行 29. 都市名フォーマット 30. 長い非正式都市フォーマット 31. 国フォーマットを出力 32. 自国 33. 通りマッチングの精度 34. 企業マッチングの精度 35. 道順マッチングの精度 36. 二重住所ロジック 37. DPV 成功ステータス条件 38. レポート リスト ファイル名 39. レポート リスト プロセッサ名 40. レポート リスト番号

パラメータ	説明
	41. レポート差出人の住所 42. レポート差出人の名前 43. レポート差出人住所の都市名行 44. 失敗時の住所行検索 45. 通り名エイリアスを出力 46. VeriMove ブロックを出力 47. DPV で No Stat を調査 48. DPV で空家かどうかを調査 49. 省略形エイリアスを出力 50. よく使用されるエイリアスを出力 51. 優先する都市を出力 52. Suite Link を実行 53. Zplus 疑似キャリア R777 を抑制
Universal Address の全般的な設定	JVM 設定を指定します。 1. DPV ファイル タイプ 2. DPV メモリ モデル 3. Lacs Link メモリ モデル 4. Suite Link メモリ モデル
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。
Compress Output	出力を圧縮するかどうかを示すフラグ。 出力を圧縮する場合は `true` を設定します。

パラメータ	説明
CASS レポート	CASS レポートを生成するための設定。`UAMAddressingFactory` インスタンスを使用して、多重定義メソッド `generateCASSReport()` のいずれかを呼び出します。 CASS レポートは PDF 形式で生成されます。 パラメータは次のとおりです。 **Counters** CASS レポートに含めるカウンタのマップ。 **Job Name** ジョブの名前。これは、CASS レポートのファイル名の一部になります。 **Path** 作成された CASS レポートが配置されるディレクトリ。これは、CASS レポートのオプションの入力値です。 `path` は、SDK ジョブがクラスタ環境で実行している場合はクラスタ上、クライアント コンピュータ上で実行している場合はクライアント コンピュータ上の場所である必要があります。 **注：** `path` が指定されていない場合は、現在の作業ディレクトリに新しい CASS レポートが配置されます。 **Report Type** 生成される CASS レポートのタイプ。列挙 **UAMCASSReportType**（414ページ）からの 1 つ以上の値を指定できます。

出力列

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. ステータス
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.Result
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

注： フィールドの説明については、Spectrum™ Technology Platform の『*Addressing ガイド*』のトピック「*Validate Address*」を参照してください。

Validate Address MapReduce ジョブの使用

注： 最初の Validate Address ジョブを作成および実行する前に、Acushare サービスが実行されていることを確認します。手順については、**Acushare サービスの実行**（13ページ）を参照してください。

1. `UAMAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `UAMAddressingDetail` のインスタンスを作成して、Validate Address ジョブの入力と出力の詳細を指定します。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) このジョブの入力設定を行うには、`UniversalAddressValidateInputConfiguration` のインスタンスを作成します。

   列挙体 **列挙 PreferredCity**（413ページ）、**列挙 CasingType**（411ページ）、**列挙 CityNameFormat**（411ページ）、**列挙 OutputCountryFormat**（412ページ）、**列挙 StandardAddressFormat**（412ページ）、**列挙 StandardAddressPMBLine**（413ページ）、**列挙 StreetMatchingStrictness**（412ページ）、**列挙 FirmMatchingStrictness**（412ページ）、**列挙 DirectionalMatchingStrictness**（413ページ）、**列挙 DualAddressLogic**（412ページ） と該当する場合は **列挙 DPVSuccessStatusCondition**（414ページ） を使用して、さまざまな必須フィールドの値を設定します。

   重要： Validate Address を CASS 認定™モードで実行するには、このインスタンスのフィールド `outputReport3553`、`outputCASSDetail`、および `outputReportSummary` を `true` に設定します。CASS レポートにはジョブを CASS 認定™ モードで実行した場合にのみ有効なコンテンツが含められます。それ以外の場合は、空白のレポート PDF が生成されます。

b) `ReferenceDataPath` のインスタンスを作成することによって、*リファレンス データ パス*の詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。

c) 各種ジョブ実行設定を行うために、上で作成した `ReferenceDataPath` インスタンスと、*COBOL ランタイム パス*および*モジュール ディレクトリ パス* (`String` 値) を引数としてコンストラクタに渡して、`UAMUSAddressingEngineConfiguration` のインスタンスを作成します。

`UAMUSAddressingEngineConfiguration` インスタンスを作成した後、その各種必須フィールドの値を設定します。

d) JVM 設定を構成するには、`UniversalAddressGeneralConfiguration` のインスタンスを作成します。

列挙体 **列挙 DPVFileType**（413ページ）、**列挙 DPVMemoryModel**（413ページ）、**列挙 LacsLinkMemoryModel**（414ページ）、および **列挙 SuiteLinkMemoryModel**（414ページ）を使用します。

e) `UAMAddressingDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `UAMUSAddressingEngineConfiguration`、`UniversalAddressGeneralConfiguration`、および `UniversalAddressValidateInputConfiguration` のインスタンスを引数としてコンストラクタに渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ）タイプのインスタンスである必要があります。

1. `inputPath` インスタンスの `UAMAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `UAMAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

注：

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` インスタンスの `UAMAddressingDetail` フィールドを使用して、ジョブの名前を設定します。
4. `UAMAddressingDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. MapReduce ジョブを作成するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `UAMAddressingDetail` のインスタンスを引数として渡します。
   `createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。
   カウンタの `Map` が返されます。
6. ジョブの正常実行後に CASS レポートを生成するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用して `generateCASSReport()` メソッドを呼び出します。多重定義されている `generateCASSReport()` メソッドのどのバージョンを呼び出しても構いません。

   使用される `generateCASSReport()` メソッド シグネチャによって、1 つ前の手順で得られたレポート カウンタの `Map`、`jobName`、生成された CASS レポートを格納する `path`、作成する `reportType` を引数として渡します。

   `path` は、SDK ジョブがクラスタ環境で実行している場合はクラスタ上、クライアント コンピュータ上で実行している場合はクライアント コンピュータ上の場所である必要があります。

   注： `path` が指定されていない場合は、現在の作業ディレクトリに新しい CASS レポートが配置されます。

   `reportType` パラメータの値は、**列挙 UAMCASSReportType**（414ページ）に記載された値でなければなりません。1 つ以上のレポート タイプをこのパラメータに指定できます。

Validate Address Spark ジョブの使用

注： 最初の Validate Address ジョブを作成および実行する前に、Acushare サービスが実行されていることを確認します。手順については、**Acushare サービスの実行**（13ページ）を参照してください。

1. `UAMAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `UAMAddressingDetail` のインスタンスを作成して、Validate Address ジョブの入力と出力の詳細を指定します。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) このジョブの入力設定を行うには、`UniversalAddressValidateInputConfiguration` のインスタンスを作成します。

      列挙体 **列挙 PreferredCity**（413ページ）、**列挙 CasingType**（411ページ）、**列挙 CityNameFormat**（411ページ）、**列挙 OutputCountryFormat**（412ページ）、**列挙 StandardAddressFormat**（412ページ）、**列挙 StandardAddressPMBLine**（413ページ）、**列挙 StreetMatchingStrictness**（412ページ）、**列挙 FirmMatchingStrictness**（412ページ）、**列挙 DirectionalMatchingStrictness**（413ページ）、**列挙 DualAddressLogic**（412ページ） と該当する場合は **列挙 DPVSuccessStatusCondition**（414ページ） を使用して、さまざまな必須フィールドの値を設定します。

      重要： Validate Address を CASS 認定™モードで実行するには、このインスタンスのフィールド `outputReport3553`、`outputCASSDetail`、および `outputReportSummary` を `true` に設定します。CASS レポートにはジョブを CASS 認定™ モードで実行した場合にのみ有効なコンテンツが含められます。それ以外の場合は、空白のレポート PDF が生成されます。

   b) `ReferenceDataPath` のインスタンスを作成することによって、*リファレンス データ パス*の詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ） を参照してください。
   c) 各種ジョブ実行設定を行うために、上で作成した `ReferenceDataPath` インスタンスと、*COBOL ランタイム パス*および*モジュール ディレクトリ パス* (`String` 値) を引数としてコンストラクタに渡して、`UAMUSAddressingEngineConfiguration` のインスタンスを作成します。

      `UAMUSAddressingEngineConfiguration` インスタンスを作成した後、その各種必須フィールドの値を設定します。

   d) JVM 設定を構成するには、`UniversalAddressGeneralConfiguration` のインスタンスを作成します。

列挙体 **列挙 DPVFileType**（413ページ）、**列挙 DPVMemoryModel**（413ページ）、**列挙 LacsLinkMemoryModel**（414ページ）、および **列挙 SuiteLinkMemoryModel**（414ページ） を使用します。

e) `UAMAddressingDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `UAMUSAddressingEngineConfiguration`、`UniversalAddressGeneralConfiguration`、および `UniversalAddressValidateInputConfiguration` のインスタンスを引数としてコンストラクタに渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

1. `inputPath` インスタンスの `UAMAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `UAMAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` インスタンスの `UAMAddressingDetail` フィールドを使用して、ジョブの名前を設定します。
4. `UAMAddressingDetail` インスタンスの `compressOutput` フラグを `true` に設定して、ジョブの出力を圧縮します。

3. Spark ジョブを作成して実行するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `UAMAddressingDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。
   カウンタの `Map` が返されます。

5. ジョブの正常実行後に CASS レポートを生成するには、先ほど作成した `UAMAddressingFactory` のインスタンスを使用して `generateCASSReport()` メソッドを呼び出します。多重定義されている `generateCASSReport()` メソッドのどのバージョンを呼び出しても構いません。

   使用される `generateCASSReport()` メソッド シグネチャによって、1 つ前の手順で得られたレポート カウンタの `Map`、`jobName`、生成された CASS レポートを格納する `path`、作成する `reportType` を引数として渡します。

   `path` は、SDK ジョブがクラスタ環境で実行している場合はクラスタ上、クライアント コンピュータ上で実行している場合はクライアント コンピュータ上の場所である必要があります。

   > **注:** `path` が指定されていない場合は、現在の作業ディレクトリに新しい CASS レポートが配置されます。

   `reportType` パラメータの値は、**列挙 UAMCASSReportType**（414ページ）に記載された値でなければなりません。1 つ以上のレポート タイプをこのパラメータに指定できます。

Validate Address Global

Validate Address Global

Validate Address Global は、米国およびカナダ以外の住所に対する高度な住所の正規化および検証機能を提供します。Validate Address Global は、米国およびカナダの住所の妥当性も確認できますが、その他の国の住所の妥当性を確認する能力に優れています。米国およびカナダ以外の住所を大量に処理する場合は、Validate Address Global の使用を検討してください。

Validate Address Global は Universal Addressing モジュールの一部です。

Validate Address Global は、パーシング、検証、書式設定など、いくつもの手順を実行して、住所の品質を高めています。

住所のパーシング、書式設定、および正規化

住所データのフィールド入力の誤りを再構成することは、特に他国の住所で行う場合、複雑で難しい作業です。住所データをコンピュータのシステムに入力する際、曖昧になってしまう部分が多いからです。特に問題なのが、(企業や個人名を通りの住所フィールドに入力するなど) 要素を誤ったフィールドに入力したり、省略形を使用する場合に、言語固有だけでなく、国固有の省略形に変えてしまうケースです。Validate Address Global は住所行の住所要素を識別し、正しいフィールドに割り当てます。これは実際の検証前に行う重要な作業です。再構成を行わなければ、"一致が見つからない" という結果になる可能性があります。

住所要素の正しい識別は、特定のフィールド長要件に合わせて住所を切り捨てたり、短縮しなければならない場合にも重要です。正しい情報が正しいフィールドに割り当てられていれば、特定の切り捨てルールを適用することができます。

- 住所行をパースおよび解析し、個々の住所要素を識別
- 30 を越える文字セットを処理
- 宛先国の郵便ルールに従って住所の書式を整える
- 住所要素を正規化 (AVENUE を AVE に変更するなど)

Global Address Validation

住所の検証は、正しくパースされた住所データを郵便組織または他のデータ プロバイダが提供する参照データベースと比較する訂正処理です。Validate Address Global は、洗練されたファジー マッチング テクノロジーを使用して個々の住所要素を検証し、正しいことを確認するとともに、郵便規格とユーザの優先設定に基づいて出力を正規化および書式設定します。FastCompletion 検証タイプは、簡易住所入力アプリケーションに使用できます。いくつかの住所フィールドには切り捨てられたデータを入力することができ、この入力に基づいて提案を生成します。

住所を完全に検証できない場合もあります。Validate Address Global には、配達可能性によって住所を分類する、ユニークな配達可能性評価機能があります。

API エンティティ

GlobalAddressingDetail<T extends ProcessType>

目的
Validate Address Global のジョブの詳細を指定します。

GlobalAddressingEngineConfiguration

目的
Validate Address Global ジョブを作成および実行するために必要なデータベース構成を設定します。

GlobalAddressingFactory

目的

Validate Address Global ジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

GlobalAddressingGeneralConfiguration

目的

Validate Address Global ジョブを作成および実行するために必要な JVM 構成を設定します。

GlobalAddressingEngineConfiguration

目的

Validate Address Global ジョブを作成および実行するために必要なデータベース構成を設定します。

入力パラメータ

パラメータ	説明
Validate Address Global Engine Configuration	次のデータベース構成を設定します。 1. データベース タイプ 2. プリロード タイプ 3. リファレンス データ パス 4. すべての国をサポートするかどうか。サポートしない場合は、サポートする国のリスト。

パラメータ	説明
Validate Address Global Input Configuration	次の入力設定を指定します。 1. 結果の州/省タイプ 2. 処理のマッチング範囲 3. 入力の強制国 ISO3 4. 入力のデフォルト国 ISO3 5. 入力の書式区切り記号 6. 結果の書式区切り記号 7. 結果に入力を含める 8. 結果の国タイプ 9. 処理の最適化レベル 10. 結果の言語設定 11. 処理のモード 12. 結果のスクリプトの設定 13. 結果の最大数 14. 結果の大文字と小文字の区別
Validate Address Global General Configuration	JVM 設定を指定します。 1. キャッシュ サイズ 2. 最大スレッド数 3. 最大住所オブジェクト数 4. 拡張する範囲 5. 柔軟な範囲拡張 6. トランザクションのログ記録の有効化 7. 最大メモリ使用量 (単位: MB)
Unlock Code	データベース内のデータをロック解除します。
リファレンス データ パス	リファレンス データ パスの詳細を指定します。 UAM ジョブの場合は、リファレンス データをクラスタ内のローカル データ ノード、または HDFS に配置できます。 **注:** ローカル データ ノードの場合は、リファレンス データをアーカイブされていないフォルダとして配置する必要があるのに対して、HDFS の場合は、`.zip` 形式でアーカイブされたファイルにする必要があります。

パラメータ	説明
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注 : `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

住所データ

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. County
14. FirmName
15. Floor
16. HouseNumber
17. LastLine
18. LeadingDirectional
19. Locality
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

元の入力データ

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

> **注：** 入力フィールド `AddressLine2.Input`、`AddressLine3.Input`、`AddressLine4.Input`、`AddressLine5.Input`、および `AddressLine6.Input` は、クラス `resultIncludeInputs` の `GlobalAddressingInputConfiguration` フィールドが `true` に設定された場合にのみ出力に含まれます。そうでない場合、これらの `AddressLineX.input` フィールドで入力に含まれるもののみが、出力に含まれます。

結果コード

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

> **注：** フィールドの説明については、Spectrum™ Technology Platform の『*Addressing ガイド*』のトピック「*Validate Address Global*」を参照してください。

Validate Address Global MapReduce ジョブの使用

1. `GlobalAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `GlobalAddressingDetail` のインスタンスを作成して、Validate Address Global ジョブの入力と出力の詳細を指定します。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `GlobalAddressingGeneralConfiguration` のインスタンスを作成することによって、JVM の初期化を設定します。

列挙体 **列挙 CacheSize**（411ページ）、**列挙 RangesToExpand**（411ページ）、および **列挙 FlexibleRangeExpansion**（411ページ） を使用します。

b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ） を参照してください。

c) 必要なデータベース設定を指定します。これには、前述の `ReferenceDataPath` インスタンスを引数として渡して、`GlobalAddressingEngineConfiguration` のインスタンスを作成します。

1. 列挙体 **列挙 PreloadingType**（407ページ） を使用してこのインスタンスの *プリロード タイプ*を設定します。
2. **列挙 DatabaseType**（407ページ） 使用してデータベース *タイプ*を設定します。
3. **列挙 CountryCodes**（407ページ） を使用してサポートされる国を設定します。
4. すべての国をサポートする場合は、`isAllCountries` 属性を true に設定します。そうでない場合は、**列挙 CountryCodes**（407ページ） の値をコンマで区切ったリストで `supportedCountries` 文字列値に指定します。

d) `GlobalAddressingInputConfiguration` のインスタンスを作成することによって、入力を設定します。

このインスタンスの各種フィールドの値を設定するには、列挙体 **列挙 CountryCodes**（407ページ）、**列挙 StateProvinceType**（407ページ）、**列挙 CountryType**（407ページ）、**列挙 PreferredScript**（408ページ）、**列挙 PreferredLanguage**（408ページ）、**列挙 Casing**（409ページ）、**列挙 OptimizationLevel**（409ページ）、**列挙 Mode**（409ページ）、および **列挙 MatchingScope**（409ページ） の該当するものを使用します。

e) データにアンロック キーを `List` の `String` 値として設定します。

f) `GlobalAddressingDetail` のインスタンスを作成します。`Config` タイプのインスタンスと、先ほど作成したアンロック コード値の `List`、`GlobalAddressingEngineConfiguration` インスタンス、および `GlobalAddressingInputConfiguration` インスタンスを引数としてコンストラクタに渡します。

`Config` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

このパラメータの `GROUPBY_REGION` の値は、デフォルトで `true` に設定されます。このジョブは、追加したリファレンス データに従って、このリージョンの住所を処理します。例えば、ドイツのリファレンス データが HDFS に配置されている場合は、ドイツの入力住所が処理されます。

1. JVM 初期化構成を設定します。`GlobalAddressingDetail` インスタンスの `generalConfiguration` フィールドを上で作成した `GlobalAddressingGeneralConfiguration` インスタンスに設定します。
2. `inputPath` インスタンスの `GlobalAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `GlobalAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

4. `jobName` インスタンスの `GlobalAddressingDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `GlobalAddressingFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `GlobalAddressingDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `GlobalAddressingFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Validate Address Global Spark ジョブの使用

1. `GlobalAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `ProcessType` を指定する `GlobalAddressingDetail` のインスタンスを作成して、Validate Address Global ジョブの入力と出力の詳細を指定します。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `GlobalAddressingGeneralConfiguration` のインスタンスを作成することによって、JVM の初期化を設定します。
      列挙体 **列挙 CacheSize**（411ページ）、**列挙 RangesToExpand**（411ページ）、および **列挙 FlexibleRangeExpansion**（411ページ） を使用します。
   b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ） を参照してください。
   c) 必要なデータベース設定を指定します。これには、前述の `ReferenceDataPath` インスタンスを引数として渡して、`GlobalAddressingEngineConfiguration` のインスタンスを作成します。
      1. 列挙体 **列挙 PreloadingType**（407ページ） を使用してこのインスタンスの *プリロード タイプ*を設定します。
      2. **列挙 DatabaseType**（407ページ） 使用してデータベース *タイプ*を設定します。
      3. **列挙 CountryCodes**（407ページ） を使用してサポートされる国を設定します。
      4. すべての国をサポートする場合は、`isAllCountries` 属性を true に設定します。そうでない場合は、**列挙 CountryCodes**（407ページ） の値をコンマで区切ったリストで `supportedCountries` 文字列値に指定します。
   d) `GlobalAddressingInputConfiguration` のインスタンスを作成することによって、入力を設定します。
      このインスタンスの各種フィールドの値を設定するには、列挙体 **列挙 CountryCodes**（407ページ）、**列挙 StateProvinceType**（407ページ）、**列挙 CountryType**（407ページ）、**列挙 PreferredScript**（408ページ）、**列挙 PreferredLanguage**（408ページ）、**列挙 Casing**（409ページ）、**列挙 OptimizationLevel**（409ページ）、**列挙 Mode**（409ページ）、および **列挙 MatchingScope**（409ページ） の該当するものを使用します。
   e) データにアンロック キーを `List` の `String` 値として設定します。
   f) `GlobalAddressingDetail` のインスタンスを作成します。`Config` タイプのインスタンスと、先ほど作成したアンロック コード値の `List`、`GlobalAddressingEngineConfiguration` インスタンス、および

`GlobalAddressingInputConfiguration` インスタンスを引数としてコンストラクタに渡します。

`Config` パラメータは、**SparkJobConfig**（33ページ）タイプのインスタンスである必要があります。

このパラメータの `GROUPBY_REGION` の値は、デフォルトで `true` に設定されます。このジョブは、追加したリファレンスデータに従って、このリージョンの住所を処理します。例えば、ドイツのリファレンス データが HDFS に配置されている場合は、ドイツの入力住所が処理されます。

1. JVM 初期化構成を設定します。`GlobalAddressingDetail` インスタンスの `generalConfiguration` フィールドを上で作成した `GlobalAddressingGeneralConfiguration` インスタンスに設定します。
2. `inputPath` インスタンスの `GlobalAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `GlobalAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

4. `jobName` インスタンスの `GlobalAddressingDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `GlobalAddressingFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `GlobalAddressingDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Validate Address Loqate

Validate Address Loqate

Validate Address Loqate は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。Validate Address Loqate は、情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州/省名など、欠落している郵便情報を追加します。

Validate Address Loqate は、Validate Address Loqate が住所の妥当性を確認したかどうか、返された住所の確信レベル、住所の妥当性が確認できなかった場合はその理由など、検証処理に関する結果インジケータも返します。

Validate Address Loqate は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを Universal Addressing モジュールの各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて正規化します。データベースにマッチしなかった場合、Validate Address Loqate は、オプションで入力住所の書式を整えます。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。
ValidateAddressLoqate は、Universal Addressing モジュールに含まれています。

API エンティティ

LoqateAddressingDetail<T extends ProcessType>

目的
Validate Address Loqate ジョブの詳細を指定します。

LoqateAddressingEngineConfiguration

目的
Validate Address Loqate ジョブを作成および実行するために必要なデータベース構成を設定します。

LoqateAddressingFactory

目的

Validate Address Loqate ジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

LoqateAddressingGeneralConfiguration

目的

Validate Address Loqate ジョブを作成および実行するために必要な JVM 構成を設定します。

LoqateAddressingValidateConfiguration

目的

Validate Address Loqate ジョブを作成および実行するために入力の設定を構成する。

入力パラメータ

パラメータ	説明
Validate Address Loqate のエンジン設定	検証を実行するための設定を指定します。 1. 詳細表示 2. ツール情報 3. 住所フォーマットを出力 4. 入力をログに記録 5. 出力をログに記録 6. ログ ファイル名 7. マッチ スコアの絶対しきい値 8. マッチ スコアのしきい値係数 9. 郵便番号の最大結果数 10. 厳密な参照一致

パラメータ	説明
Validate Address Loqate の検証設定	次の入力設定を指定します。 1. 標準住所を含める 2. 一致した住所要素を含める 3. 正規化された入力住所要素 4. 住所データ ブロックを返す 5. 出力の大文字と小文字の区別 6. 個々のフィールドの結果コードを含める 7. 複数の住所を返す 8. 複数の一致が見つかると失敗 9. 複数住所カウント 10. 国フォーマット 11. デフォルトの国 12. スクリプト アルファベット 13. ジオコード アドレス フィールドを返す 14. 許容レベル 15. 最小マッチ スコア 16. AMAS 表記を使用してデータをフォーマット 17. 重複処理である 18. 単一フィールドの重複処理 19. 複数フィールドの重複処理 20. 非標準フィールドの重複処理 21. 出力フィールドの重複処理
Validate Address Loqate の全般的な設定	JVM 設定を指定します。 1. 最大アイドル オブジェクト数 2. 最小アイドル オブジェクト数 3. 最大アクティブ オブジェクト数 4. 最長待ち時間 5. 枯渇時のアクション 6. 借用時のテスト 7. 復帰時のテスト 8. アイドル中のテスト 9. 退出の実行間隔 (ミリ秒) 10. 退出実行 1 回あたりのテスト数 11. 最小退出可能アイドル時間 (ミリ秒)

パラメータ	説明
リファレンス データ パス	リファレンス データ パスの詳細を指定します。 UAM ジョブの場合は、リファレンス データをクラスタ内のローカル データ ノード、または HDFS に配置できます。 **注 :** ローカル データ ノードの場合は、リファレンス データをアーカイブされていないフォルダとして配置する必要があるのに対して、HDFS の場合は、`.zip` 形式でアーカイブされたファイルにする必要があります。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 注 : `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

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. Status
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

*これはサブフィールドであり、データを含まない場合があります。

表 2 : 都市/通り/郵便番号セントロイド マッチ コード

要素	マッチ コード
住所ポイント	P4
住所ポイント補間済み	I4
通りセントロイド	A4/P3
郵便番号/都市セントロイド	A3/P2/A2

注 : フィールドの説明については、Spectrum™ Technology Platform の『*Addressing ガイド*』のトピック「*Validate Address Loqate*」を参照してください。

Validate Address Loqate MapReduce ジョブの使用

1. `LoqateAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `LoqateAddressingDetail` を指定する `ProcessType` のインスタンスを作成して、Validate Address Loqate ジョブの入力と出力の詳細を指定します。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `LoqateAddressingGeneralConfiguration` のインスタンスを作成することによって、JVM の初期化を設定します。

      列挙体 **列挙 ExhaustedAction**（410ページ） を使用します。
   b) `LoqateAddressingEngineConfiguration` のインスタンスを作成して必要なデータベース設定を行い、各種フィールドを設定します。
   c) `LoqateAddressingValidateConfiguration` のインスタンスを作成して、住所検証の設定を行います。

      このインスタンスの各種フィールドの値を設定するには、列挙体 **列挙 AcceptanceLevel**（410ページ）、**列挙 CountryCodes**（407ページ）、**列挙 OutputCasing**（410ページ）、**列挙 CountryFormat**（410ページ）、および **列挙 ScriptAlphabet**（411ページ） を使用します。
   d) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ） を参照してください。
   e) `LoqateAddressingDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンス、`LocalReferenceDataPath` インスタンス、および先ほど作成した

`LoqateAddressingValidateConfiguration` インスタンスを引数としてコンストラクタに渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

1. `inputPath` インスタンスの `LoqateAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

   注：

   - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `LoqateAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` インスタンスの `LoqateAddressingDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `LoqateAddressingFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `LoqateAddressingDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `LoqateAddressingFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Validate Address Loqate Spark ジョブの使用

1. `LoqateAddressingFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. `LoqateAddressingDetail` を指定する `ProcessType` のインスタンスを作成して、Validate Address Loqate ジョブの入力と出力の詳細を指定します。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。これを行うには、次の手順に従います。
   a) `LoqateAddressingGeneralConfiguration` のインスタンスを作成することによって、JVM の初期化を設定します。

      列挙体 **列挙 ExhaustedAction**（410ページ） を使用します。

   b) `LoqateAddressingEngineConfiguration` のインスタンスを作成して必要なデータベース設定を行い、各種フィールドを設定します。
   c) `LoqateAddressingValidateConfiguration` のインスタンスを作成して、住所検証の設定を行います。

      このインスタンスの各種フィールドの値を設定するには、列挙体 **列挙 AcceptanceLevel**（410ページ）、**列挙 CountryCodes**（407ページ）、**列挙 OutputCasing**（410ページ）、**列挙 CountryFormat**（410ページ）、および **列挙 ScriptAlphabet**（411ページ） を使用します。

   d) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ） を参照してください。
   e) `LoqateAddressingDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンス、`LocalReferenceDataPath` インスタンス、および先ほど作成した `LoqateAddressingValidateConfiguration` インスタンスを引数としてコンストラクタに渡します。

      `JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

      1. `inputPath` インスタンスの `LoqateAddressingDetail` フィールドを使用して、入力ファイルの詳細を設定します。

         注：

         - テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
         - ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。

- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

2. `LoqateAddressingDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

   注：

   - テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
   - ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
   - PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

3. `jobName` インスタンスの `LoqateAddressingDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `LoqateAddressingFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `LoqateAddressingDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

Universal Name モジュールのジョブ

コモンモジュール API

UniversalNameDetail<T extends ProcessType>

目的
Universal Name モジュールのジョブの詳細を指定します。

UniversalNameFactory

目的
Universal Name モジュールのジョブのインスタンスを作成するためのシングルトン ファクトリ クラス。

Open Name Parser

OpenNameParser は、名前データ フィールドにある個人名、企業名、またはその他の名称を構成要素に分解します。パースされたこれらの名前要素は、名前のマッチング、名前の正規化、複数レコード名の統合など、他の自動化処理に使用できます。

API エンティティ

OpenNameParserDetail

目的
Open Name Parser ジョブの詳細を指定します。

OpenNameParserConfiguration

目的
`name` データ フィールドにある個人名、企業名、またはその他の名称を構成要素に分解します。

入力パラメータ

パラメータ	説明
Open Name Parser Configuration	`name` データ フィールドにある個人名、企業名、またはその他の名称を構成要素に分解します。
リファレンス データ パス	リファレンス データ パスの詳細を指定します。
Job Configurations	ジョブ用の Hadoop 設定 MapReduce ジョブの場合、インスタンスのタイプは **MRJobConfig**（32ページ） である必要があります。Spark ジョブの場合、インスタンスのタイプは **SparkJobConfig**（33ページ） である必要があります。

パラメータ	説明
Input File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の入力テキスト ファイルのパス。 **レコード区切り文字** 入力ファイル内で使用されるレコード区切り文字。 **Field Separator** 入力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **Text Qualifier** 区切り記号付きファイル内のテキスト値を囲むのに使用する文字。 **Header Row Fields** 入力ファイルのヘッダー フィールドの配列。 **Skip First Row** 入力ファイル レコードの読み取り時に、先頭行をスキップするかどうかを示すフラグ。 先頭行がヘッダー行である場合は、これを `true` にする必要があります。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の入力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の入力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Field Mappings** キー値ペアのマップ。既存の列名をキーとし、対応する出力列名を値としてマッピングします。

パラメータ	説明
Output File	*テキスト ファイルの場合:* **File Path** Hadoop プラットフォーム上の出力テキスト ファイルのパス。 **Field Separator** 出力ファイルで、レコード内の連続する 2 つのフィールドの間に使用される区切り文字。 **注:** `FilePath` の適切なコンストラクタを呼び出します。 *ORC 形式ファイル:* **ORC File Path** Hadoop プラットフォーム上の出力 ORC 形式ファイルのパス。 *PARQUET 形式ファイル:* **Parquet File Path** Hadoop プラットフォーム上の出力 PARQUET 形式ファイルのパス。 *共通パラメータ:* **Overwrite** 出力ファイルと同じ名前のファイルが既に存在する場合に、上書きするかどうかを示すフラグ。 **Create Output Header** ヘッダー ファイルを Hadoop サーバー上に作成するかどうかを示すフラグ。
Job Name	ジョブの名前。

出力列

入力列に加えて、Open Name Parser ジョブの出力生成時に以下の列が追加されます。

	書式	説明
AccountDescription	文字列	名前の一部であるアカウント説明。例えば、"Mary Jones Account # 12345" で、アカウント説明は "Account#12345"。

書式	説明
会社名関係のフィールド	
FirmConjunction	文字列	"d/b/a" (doing business as)、"o/a" (operating as)、"t/a" (trading as) などの略語を含む企業の名前を示します。
FirmName	文字列	会社名。例えば、"Pitney Bowes"。
FirmSuffix	文字列	会社名の接尾語。例えば、"Co."、"Inc."
IsFirm	文字列	名前が、個人名ではなく、企業名であることを示します。値は True または False です。
個人名に関係するフィールド	
Conjunction	文字列	名前に、"and"、"or"、"&" などの接続詞が含まれることを示します。
CultureCode	文字列	入力データに含まれるカルチャー コード。
CultureCodeUsedToParse	文字列	データのパースに使用されたカルチャー固有のグラマーを特定します。 **Null (empty)** グローバル カルチャー (デフォルト)。 **de** ドイツ語。 **es** スペイン語。 **ja** 日本語。
FirstName	文字列	個人のファースト ネーム。
GeneralSuffix	文字列	個人名の一般/職業接尾語。例えば、 MD PhD 。

	書式	説明
IsParsed	文字列	出力レコードがパースされたかどうかを示します。値は True または False です。
IsPersonal	文字列	名前が企業名ではなく、個人名であるかどうかを示します。値は True または False です。
IsReverseOrder	文字列	入力名が逆順序であるかどうかを示します。値は True または False です。
LastName	文字列	個人名のラスト ネーム。父方の姓が含まれます。
LeadingData	文字列	名前の前に付けられる、名前以外の情報。
MaturitySuffix	文字列	個人の世代/家族接尾語。例えば、Jr.または Sr.。
MiddleName	文字列	個人のミドル ネーム。
Name.	文字列	入力に指定された個人名または企業名。
NameScore	文字列	各名前の既知および不明トークンの平均スコアを示します。NameScore の値は、パーシング グラマーでの定義に従って、0 ～ 100 の間になります。マッチが返されない場合は、0 が返されます。
SecondaryLastName	文字列	スペイン語のパーシング グラマーでは、その人の母の姓。
TitleOfRespect	文字列	"Mr."、"Mrs."、"Dr." など、名前の前に付けられる情報。
TrailingData	文字列	名前の後に付けられる、名前以外の情報。

	書式	説明
結合名関係のフィールド		
Conjunction2	文字列	結合されている 2 番目の名前に、"and"、"or"、"&" などの接続詞が含まれることを示します。
Conjunction3	文字列	結合されている 3 番目の名前に、"and"、"or"、"&" などの接続詞が含まれることを示します。
FirmName2	文字列	結合されている 2 番目の企業名。例えば、Baltimore Gas & Electric dba Constellation Energy。
FirmSuffix2	文字列	結合されている 2 番目の企業の接尾語。
FirstName2	文字列	結合されている FirstName の 2 番目の名
FirstName3	文字列	結合されている FirstName の 3 番目の名。
GeneralSuffix2	文字列	結合されている 2 番目の名前の一般/職業接尾語。例えば、 MD PhD 。
GeneralSuffix3	文字列	結合されている 3 番目の名前の一般/職業接尾語。例えば、 MD PhD 。
IsConjoined	文字列	入力名が結合名であることを示します。結合名は、例えば、"John and Jane Smith"。値は True または False です。
LastName2	文字列	結合されている LastName の 2 番目の姓。
LastName3	文字列	結合されている LastName の 3 番目の姓。

	書式	説明
MaturitySuffix2	文字列	結合されている 2 番目の名前の世代/家族接尾語。例えば、Jr.または Sr.。
MaturitySuffix3	文字列	結合されている 3 番目の名前の世代/家族接尾語。例えば、Jr.または Sr.。
MiddleName2	文字列	結合されている MiddleName の 2 番目の名。
MiddleName3	文字列	結合されている MiddleName の 3 番目の名。
TitleOfRespect2	文字列	"Mr."、"Mrs."、"Dr." など、結合されている 2 番目の名前の前に付けられる情報。
TitleOfRespect3	文字列	"Mr."、"Mrs."、"Dr." など、結合されている 3 番目の名前の前に付けられる情報。

Open Name Parser MapReduce ジョブの使用

1. `UniversalNameFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Open Name Parser ジョブの入力と出力の詳細を指定します。以下の手順に従って、`OpenNameParserDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**MRProcessType**（34ページ） タイプを使用する必要があります。
   a) `OpenNameParserConfiguration` のインスタンスを作成することによって、Open Name Parser ルールを設定します。
   b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。
   c) `OpenNameParserDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `OpenNameParserConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**MRJobConfig**（32ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `OpenNameParserDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `OpenNameParserDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `OpenNameParserDetail` フィールドを使用して、ジョブの名前を設定します。

3. MapReduce ジョブを作成するには、先ほど作成した `UniversalNameFactory` のインスタンスを使用してそのメソッド `createJob()` を呼び出します。ここで、上の `OpenNameParserDetail` のインスタンスを引数として渡します。`createJob()` メソッドは、`List` のインスタンスの `ControlledJob` を返します。
4. `JobControl` のインスタンスを使用して、作成したジョブを実行します。
5. MapReduce ジョブの正常実行後にレポート カウンタを表示するには、先ほど作成した `UniversalNameFactory` のインスタンスを使用して、そのメソッド `getCounters()` を呼び出します。作成したジョブを引数として渡します。

Open Name Parser Spark ジョブの使用

1. `UniversalNameFactory` のインスタンスを、その静的メソッド `getInstance()` を使用して作成します。
2. Open Name Parser ジョブの入力と出力の詳細を指定します。以下の手順に従って、`OpenNameParserDetail` を指定する `ProcessType` のインスタンスを作成することによって、これを行います。このインスタンスは、**SparkProcessType**（34ページ） タイプを使用する必要があります。

a) `OpenNameParserConfiguration` のインスタンスを作成することによって、Open Name Parser ルールを設定します。

b) `ReferenceDataPath` のインスタンスを作成することによって、リファレンス データ パスと場所のタイプの詳細を設定します。**列挙 ReferenceDataPathLocation**（404ページ）を参照してください。

c) `OpenNameParserDetail` のインスタンスを作成します。`JobConfig` タイプのインスタンスと、上で作成した `OpenNameParserConfiguration` と `ReferenceDataPath` のインスタンスを、コンストラクタの引数として渡します。

`JobConfig` パラメータは、**SparkJobConfig**（33ページ） タイプのインスタンスである必要があります。

d) `inputPath` インスタンスの `OpenNameParserDetail` フィールドを使用して、入力ファイルの詳細を設定します。

- テキスト入力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な入力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 入力ファイルの場合、ORC 入力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 入力ファイルの場合、PARQUET 入力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

e) `OpenNameParserDetail` インスタンスの `outputPath` フィールドを使用して、出力ファイルの詳細を設定します。

- テキスト出力ファイルの場合は、適切なコンストラクタを呼び出して、関連する詳細な出力ファイル情報を指定して`FilePath` のインスタンスを作成します。
- ORC 出力ファイルの場合、ORC 出力ファイルのパスを引数に指定して `OrcFilePath` のインスタンスを作成します。
- PARQUET 出力ファイルの場合、PARQUET 出力ファイルのパスを引数に指定して `ParquetFilePath` のインスタンスを作成します。

f) `jobName` インスタンスの `OpenNameParserDetail` フィールドを使用して、ジョブの名前を設定します。

3. Spark ジョブを作成して実行するには、先ほど作成した `UniversalNameFactory` のインスタンスを使用してそのメソッド `runSparkJob()` を呼び出します。ここで、上の `OpenNameParserDetail` のインスタンスを引数として渡します。

   `runSparkJob()` メソッドはジョブを実行し、ジョブのレポート カウンタの `Map` を返します。

4. カウンタを表示することにより、ジョブに対する統計レポートを表示します。

5 - XML 設定ファイル

このセクションの構成

サンプル設定ファイル 196
Advanced Matching モジュール 198
Data Integration モジュール 227
Data Normalization モジュール 235
Global Addressing モジュール 247
Universal Addressing モジュール 255
Universal Name モジュール 275

サンプル設定ファイル

サンプルの設定XMLファイルを使用すると、さまざまな住所およびデータ品質アクティビティに対して MapReduce ジョブや spark ジョブを簡単に実行できます。これらのファイルは、Java コードを理解せずにジョブを実行したいユーザ向けです。ファイルには、キー/値ペアの形式でプロパティが含まれており、必要に応じて変更できます。

コマンド プロンプト (Linux システムの場合) や、Putty などの SSH クライアント (Windows および Unix システムの場合) を使用して、必要なジョブを実行できます。

サンプル設定ファイルは Spectrum™ Technology Platform SDK に付属しており、SDK をインストールした後、次の場所でアクセスできます。

- `<Big Data Quality bundle>\samples\configuration\mr`: MR ジョブの場合
- `<Big Data Quality bundle>\samples\configuration\spark`: Spark ジョブの場合

ファイルの種類

これらの場所にある各フォルダには、ジョブを処理するために必要なプロパティをパラメータと値の形式で持つ以下の種類の設定XMLファイルがあります。実行しているジョブの要件に応じて値をカスタマイズできます。

- *inputFileConfig.xml*: 入力ファイルの種類、保存場所のパス、レコード区切り文字、フィールド区切り文字、テキスト修飾子、ファイル ヘッダーの詳細など、入力ファイルのプロパティを指定します。
- *<job>Config.xml* (例えば、*addressValidationConfig*): ジョブの種類、ジョブの名前、入力オプション、ルール設定、エンジン設定など、ジョブ関連のプロパティを指定します。
- *mapReduceConfig.xml*: MapReduce 設定パラメータを指定します。ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの任意の MapReduce パラメータのカスタマイズにこのファイルを使用します。
- *OutputFileConfig.xml*: 出力ファイルの種類、ファイルの場所、ファイル内で使用されるフィールド区切り文字、ヘッダー ファイルの作成が必要かどうか、レポート カウンタをファイルまたはコンソールに出力するかどうかなどを指定します。

設定プロパティ ファイルの使用

Spectrum™ Data & Address Quality for Big Data SDK がコンピュータ上にインストールされていることを確認します。

Spectrum™ Data & Address Quality for Big Data SDK ジョブは、モジュール固有の JAR ファイルと XML 形式の設定ファイルを使用して実行できます。

モジュール固有の JAR ファイルの一覧については、**SDK Java API のコンポーネント**（28ページ）を参照してください。

1. Linux システムの場合は、コマンド プロンプトを起動します。

   Windows および Unix システムの場合は、*Putty* などの SSH クライアントを起動します。

2. *MapReduce* ジョブの場合は、コマンド `hadoop` を使用します。

   実行するジョブによって、次の操作を行います。

   1. そのモジュールの JAR ファイル名を引き渡します。
   2. ドライバ クラス名 `RunMRSampleJob` を引き渡します。
   3. 各種設定ファイルを引数リストとして渡します。各引数キーに、1 つの設定プロパティ ファイルのパスが指定できます。各ファイルには、複数の設定プロパティが含まれます。

   コマンドの構文は次のとおりです。

   ```
   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>]
   ```

   例えば、MapReduce MatchKeyGenerator ジョブの場合は次のようになります。

   ```
   hadoop jar amm.core.12.2.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. *Spark* ジョブの場合は、コマンド `spark-submit` を使用します。

   実行するジョブによって、次の操作を行います。

   1. そのモジュールの JAR ファイル名を引き渡します。
   2. ドライバ クラス名 `RunSparkSampleJob` を引き渡します。
   3. 各種設定ファイルを引数リストとして渡します。各引数キーに、1 つの設定プロパティ ファイルのパスが指定できます。各ファイルには、複数の設定プロパティが含まれます。

   コマンドの構文は次のとおりです。

   ```
   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>]
   ```

   例えば、Spark MatchKeyGenerator ジョブの場合は次のようになります。

```
spark-submit --class RunSparkSampleJob amm.core.12.2.jar -config
/home/hadoop/spark/matchkey/matchKeyGeneratorConfig.xml -input
/home/hadoop/spark/matchkey/inputFileConfig.xml -output
/home/hadoop/spark/matchkey/outputFileConfig.xml
```

注: `hadoop` または `spark-submit` コマンドでサポートされる引数キーの一覧を表示するには、次のコマンドを実行します。

```
hadoop --help
```

または

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

Advanced Matching モジュール

Best of Breed

Best of Breed は、重複レコードのコレクションから選択する最良のデータを使用して新しい統合レコードを作成することで、重複レコードを統合します。この "スーパー" レコードは、最良の組み合わせレコードと呼ばれます。処理対象レコードの選択で使用するルールを定義します。処理が完了すると、最良の組み合わせレコードがシステムに保持されます。

設定ファイル

以下の表には、Best of Breed ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 3 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/bestofbreed/input/BestOfBreed_Input.csv`

パラメータ	説明
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 4 : bestOfBreedConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`BestOfBreed`です。
pb.bdq.job.name	ジョブの名前。デフォルトは `BestOfBreedSample` です。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.consolidation.json	重複レコードを統合するためのルールを定義する Json 文字列。

表 5 : mapReduceConfig

MapReduce 設定パラメータを指定します。
ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。 **注 :** このファイルは MapReduce ジョブでのみ使用します。

表 6 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例えば、/user/hduser/sampledata/bestofbreed/output。
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注 :** ページ サイズが小さすぎると、圧縮に支障が生じます。

パラメータ	説明
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Candidate Finder

Candidate Finder は、一連の潜在的なマッチを形成する候補レコードを取得します。検索インデックス検索は Transactional Match とは別に機能します。また、Candidate Finder では、データの書式によっては、サスペクト レコード、候補レコード、またはその両方のレコードの名前や住所のパーシングが必要となる場合もあります。

また、Candidate Finder ではフルテキスト インデックス検索も可能で、さまざまな検索タイプ (数値、範囲、すべて含む、いずれも含まない) と条件 (すべて真、いずれかが真) を使用して、文字やテキストの高度な検索条件を容易に定義できます。

> **注：** 検索インデックスを保存するには、クラスタで *HBase NoSQL* データベースが利用でき、アクセス可能であることが必要です。

設定ファイル

これらの表には、*Candidate Finder* ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 7 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/candidatefinder/input/CandidateFinder_Input.csv`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。例: `IN_MonthNumber,IN_WeekNumber,IN_MonthName,IN_WeekdayName`
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 8 : candidateFinderConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブでの値は、`CandidateFinder` です。
pb.bdq.job.name	ジョブの名前。 デフォルト値は `CandidateFinderSample` です。
pb.bdq.amm.search.cf.query.json	Candidate Finder クエリの JSON 文字列を定義します。

パラメータ	説明
pb.bdq.amm.search.cf.index.output.fields	インデックス内のどの保存済みフィールドが出力に含まれるかを指定します。
pb.bdq.amm.search.cf.index.name	インデックスまたはテーブルの名前を定義します。
pb.bdq.amm.search.cf.max.results	ステージによって返されるレスポンスの最大数を指定します。デフォルト値は 10 です。
pb.bdq.amm.search.cf.fetch.batchsize	結果の最大数が任意に大きい場合に備えて、結果を処理するバッチのサイズを指定します。これにより、多数のレコードの処理が最適化されます。 デフォルト値は 10000 です。
pb.bdq.amm.search.cf.start.record	検索の起点となるレコード番号。 デフォルトは 1 です。

表 9 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

注 : このジョブでは、これら 2 つの追加の MapReduce および Spark 設定パラメータの値を指定する必要があります。

- *hbase.zookeeper.quorum*
- *hbase.zookeeper.property.clientPort*

表 10 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: /user/hduser/sampledata/candidatefinder/output。
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。

Duplicate Synchronization

Duplicate Synchronization は、レコードのコレクションから、そのコレクション内のすべてのレコードの対応するフィールドにコピーするフィールドを指定します。フィールド データをコレクション内の別のレコードにコピーするときに従う必要があるルールを指定できます。処理が完了すると、コレクション内のレコードがすべて保持されます。

設定ファイル

これらの表には、Duplicate Synchronization ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 11 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は TEXT、ORC、PARQUET のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: /user/hduser/sampledata/duplicatesync/input/DuplicateSync_Input.csv

パラメータ	説明
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 12 : duplicateSyncConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`DuplicateSynchronization` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `DuplicateSynchronizationSample` です。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.consolidation.json	Duplicate Synchronization 統合ルールの Json 文字列。

表 13 : mapReduceConfig

MapReduce 設定パラメータを指定します。
ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。 **注 :** このファイルは MapReduce ジョブでのみ使用します。

表 14 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例えば、user/hduser/sampledata/duplicatesync/output。
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注 :** ページ サイズが小さすぎると、圧縮に支障が生じます。

パラメータ	説明
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

フィルタ

Filter ステージでは、指定したルールに基づいて、レコードをレコードのグループに保持または削除します。

設定ファイル

これらの表には、Filter ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 15 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。

パラメータ	説明
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/filter/input/Filter_Input.csv`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 16 : filterConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`Filter` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `FilterSample` です。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.consolidation.json	Duplicate Synchronization 統合ルールの Json 文字列。

表 17 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 18 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例えば、`user/hduser/sampledata/filter/output`。
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。

パラメータ	説明
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Interflow Match

Interflow Match は、2 つの入力レコード ストリーム内の類似するデータ レコード間でマッチを検出します。最初のレコード ストリームはサスペクト レコードのソースで、2 番目のストリームは候補レコードのソースです。

Interflow Match では、マッチ グループ条件 (マッチ キー等) を使用して、特定のサスペクト レコードと重複する可能性があるレコードのグループを識別します。

各候補はサスペクトと個々に照合され、マッチ ルールに基づいてスコアが付けられます。候補が重複の場合は、コレクション番号が割り当てられ、そのマッチ レコード タイプに重複が設定され、その候補が書き出されます。マッチしないユニークな候補を書き出すかどうかは、ユーザの選択に基づきます。Interflow Match によって、現在のマッチ グループ内の候補レコードがすべて使用されると、マッチするサスペクト レコードには、重複レコードに対応するコレクション番号が割り当てられます。また、マッチが見つからない場合は、サスペクトにコレクション番号 0 が割り当てられ、そのラベルにユニーク レコードが設定されます。

> **注：** Interflow Match は、サスペクト レコードと候補レコードの照合のみを行います。Intraflow Match のように、サスペクト レコードと他のサスペクト レコードとの照合は行いません。

重複の数を制限している場合に、現在のサスペクトでその制限を越えると、特定のサスペクトのマッチング処理が終了することがあります。

Express キーのマッチ結果を候補マッチ スコアに変換する方法は、マッチングのタイプ (Intraflow または Interflow) によって異なります。Interflow マッチングの場合、正常な Express キー マッチは常に 100 マッチスコアを候補に与えます。一方、Intraflow マッチングの場合、Express キー マッチの結果として候補に与えられるスコアは、その候補がマッチするレコードが他のサスペクトのマッチであるかどうかによって異なります。つまり、Express キーがサスペクトの重複であれば、マッチスコアは常に 100 になり、Express キーが他の候補の重複であれば、その候補のマッチスコア (必ずしも 100 ではない) が継承されます。

設定ファイル

これらの表には、Interflow Match ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 19 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は TEXT、ORC、PARQUET のいずれかです。
Suspect File	
pb.bdq.match.suspect.inputfile.path	サスペクト入力ファイルを HDFS 上に置いたパス。例: /user/hduser/sampledata/intermatch/input/Interflow_Suspect.txt
pb.bdq.match.suspect.recordseparator	サスペクト ファイル内で使用されるレコード区切り文字。例えば、LINUX、MACINTOSH、またはWINDOWS

パラメータ	説明
pb.bdq.match.suspect.fieldseparator	カンマ (,) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.match.suspect.textqualifier	入力ファイルの列またはフィールドのテキスト修飾子(存在する場合)。
pb.bdq.match.suspect.header	サスペクト ファイルで使用されるヘッダー。例: `name`、`firstname`、`lastname`、`matchkey`、`middlename`、`recordid`
pb.bdq.match.suspect.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。
Candidate File	
pb.bdq.match.candidate.inputfile.path	候補入力ファイルを HDFS 上に配置したパス。例: `/user/hduser/sampledata/intermatch/input/Interflow_candidate.txt`
pb.bdq.match.candidate.recordseparator	候補ファイル内で使用されるレコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.match.candidate.fieldseparator	カンマ (,) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.match.candidate.textqualifier	入力ファイルの列またはフィールドのテキスト修飾子(存在する場合)。
pb.bdq.match.candidate.header	候補ファイルで使用されるヘッダー。例: `name`、`firstname`、`lastname`、`matchkey`、`middlename`、`recordid`
pb.bdq.match.candidate.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 20 : interMatchConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`InterMatch` です。

パラメータ	説明
pb.bdq.job.name	ジョブの名前。デフォルトは `InterMatchSample` です。
pb.bdq.match.rule	マッチ ルールを定義する Json 文字列。マッチ ルール階層、マッチング方法、フィールド内のブランク データをスコアリングする方法、スコアリング方法、フィールド名の値が一致するかどうかを判断するアルゴリズムなどの詳細を指定します。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.match.express.column	Express マッチ列の名前。この列の内容がサスペクトと候補の間で合致する場合、サスペクトと候補が重複しているかどうかを判断するための処理は必要ありません。
pb.bdq.match.keygenerator.json	expressMatchKey を使用するかどうか、matchKeyField の名前、使用するアルゴリズムなど、Match Key Generator のルールを定義する Json 文字列。 **注:** これはオプションの詳細です。
pb.bdq.match.unique.collectnumber.zero	値が `true` のときユニーク レコードにコレクション番号 `0` を割り当てます。
pb.bdq.match.inter.comparison	Inter match の比較オプション。 • returnUniqueCandidates: 値を `true` に設定すると、マッチ グループ内のユニーク レコードを返します。 • maxNumOfDuplicates: マッチ グループで比較を停止する前に検出される重複の最大数を指定します。 **注:** これはオプションの詳細です。

表 21 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 22 : 出力ファイル設定

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力を生成するパス。例えば、/user/hduser/sampledata/intermatch/output。
pb.bdq.outputformat.field.delimiter	カンマ (,) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。True は、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が false の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。

パラメータ	説明
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注:** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Intraflow Match

Intraflow Match は、単一の入力ストリーム内の類似するデータ レコード間でマッチを検出します。データフローの別のコンポーネントで定義または作成したフィールドに基づいて、階層的なルールを作成できます。

設定ファイル

これらの表には、Intraflow Match ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 23 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/intramatch/input/Intraflow_Input.txt`。
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダー。例: `name`、`firstname`、`lastname`、`matchkey`、`middlename`、`recordid`
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 24 : intraMatchConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`IntraMatch` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `IntraMatchSample` です。
pb.bdq.match.rule	マッチ ルールを定義する Json 文字列。マッチ ルール階層、マッチング方法、フィールド内のブランク データをスコアリングする方法、スコアリング方法、フィールド名の値が一致するかどうかを判断するアルゴリズムなどの詳細を指定します。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.match.express.column	Express マッチ列の名前。この列の内容がサスペクトと候補の間で合致する場合、サスペクトと候補が重複しているかどうかを判断するための処理は必要ありません。
pb.bdq.match.keygenerator.json	expressMatchKey を使用するかどうか、matchKeyField の名前、使用するアルゴリズムなど、Match Key Generator のルールを定義する Json 文字列。 **注 :** これはオプションの詳細です。
pb.bdq.match.unique.collectnumber.zero	値が `true` のときユニーク レコードにコレクション番号 0 を割り当てます。

表 25 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 26 : 出力ファイル設定

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
outputfile.path	HDFS で出力ファイルを生成するパス。例えば、/user/hduser/sampledata/intramatch/output。
pb.bdq.outputformat.field.delimiter	カンマ (,) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値がtrueのとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。Trueは、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が false の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注 :** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。

パラメータ	説明
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Match Key Generator

Match Key Generator は、レコードごとに非ユニーク キーを作成します。この非ユニーク キーは、潜在的な重複レコードのグループを特定するためにマッチング ステージで使用できます。マッチ キーを使用すると、レコードをマッチ キー別にグループ化し、各グループ内でのみレコードを比較できるので、マッチング プロセスが促進されます。

マッチ キーは、定義したルールを使用して作成され、入力フィールドから構成されます。指定する入力フィールドごとに、そのフィールドで実行されるアルゴリズムが選択されます。その後、各アルゴリズムの結果を連結して、単一のマッチ キー フィールドが作成されます。

マッチ キーの作成に加え、後のデータフローの Intraflow Match ステージまたは Interflow Match ステージで使用する Express マッチ キーも作成できます。

複数のマッチ キーおよび Express マッチ キーを作成できます。

例えば、次のような入力レコードがあり、

名 - Fred
姓 - Mertz
郵便番号 - 21114-1687
性別コード - M

次のようなレコードのデータを組み合わせてマッチ キーを生成するマッチ キー ルールを定義したとします。

入力フィールド	開始位置	長さ
郵便番号	1	5
郵便番号	7	4
姓	1	5
名	1	5
性別コード	1	1

次のようなキーになります。

211141687MertzFredM

設定ファイル

これらの表には、Match Key Generator ジョブを実行する前に指定する必要のあるパラメータと値が記載されています。

表 27 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は TEXT、ORC、PARQUET のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: /user/hduser/sampledata/matchkeygenerator/input/MatchKey_Input.csv。
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、LINUX、MACINTOSH、またはWINDOWS
pb.bdq.inputformat.field.delimiter	カンマ (,) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。

パラメータ	説明
pb.bdq.inputformat.file.header	カンマで区切られた値としての列ヘッダー。例えば、businessname、id、domain。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 28 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 29 : matchKeyGeneratorConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は `MatchKeyGen` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `MatchKeySample` です。
pb.bdq.match.keygenerator.json	Match Key Generator のルールを表す JSON 文字列。マッチ キーの生成に使用するアルゴリズム、選択したアルゴリズムを適用するフィールド、指定したフィールド内の開始位置、開始位置から含める文字列の長さ、数字およびアルファベット以外の文字を除去するかどうか、入力フィールドをソートするかどうかなど。

表 30 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。

パラメータ	説明
pb.bdq.outputformat.field.delimiter	カンマ (`,`) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。`UNCOMPRESSED`、`SNAPPY`、`GZIP`、`LZO` のいずれかを指定します。 デフォルトは `UNCOMPRESSED` です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注 :** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。

パラメータ	説明
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Transactional Match

Transactional Match では、サスペクト レコードを、Candidate Finder ステージから返される候補レコードと照合します。Transactional Match では、マッチング ルールを使用して、サスペクト レコードを、同じ候補グループ番号 (Candidate Finder で割り当てられる番号) を持つすべての候補レコードと比較して、重複を識別します。候補レコードが重複の場合は、コレクション番号が割り当てられ、そのマッチ レコード タイプに重複が設定され、その候補レコードが書き出されます。グループ内のマッチしない候補にはコレクション番号 0 が割り当てられ、そのラベルにユニークが設定され、その候補が書き出されます。

> **注 :** Transactional Match は、サスペクト レコードと候補レコードの照合のみを行います。Intraflow Match のように、サスペクト レコードと他のサスペクト レコードとの照合は行いません。

設定ファイル

これらの表には、Transactional Match ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 31 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/transactionalmatch/input/TransMatch_Input.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`

パラメータ	説明
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダー。例: `name`、`firstname`、`lastname`、`CandidateGroup`、`middlename`、`recordid`。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 32 : transactionalMatchConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`Transactional` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `TransactionalMatchSample` です。
pb.bdq.match.rule	マッチ ルールを定義する Json 文字列。マッチ ルール階層、マッチング方法、フィールド内のブランク データをスコアリングする方法、スコアリング方法、フィールド名の値が一致するかどうかを判断するアルゴリズムなどの詳細を指定します。
pb.bdq.match.groupby	マッチ キューのレコードをグループ化するために使用される列の名前。
pb.bdq.reduce.count	実行されるリデューサーの数。デフォルトは `1` です。
pb.bdq.match.keygenerator.json	expressMatchKey を作成するかどうか、matchKeyField、およびマッチ キーの生成ルールを指定する Json 文字列。 **注 :** オプションの詳細
pb.bdq.match.unique.candidate.return	一意の候補レコードを出力に含める場合は、`true` に設定します。

表 33 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 34 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
outputfile.path	HDFS で出力ファイルを生成するパス。例えば、/user/hduser/sampledata/transactionalmatch/output。
pb.bdq.outputformat.field.delimiter	カンマ (,) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。Trueは、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が false の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。

パラメータ	説明
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注 :** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Data Integration モジュール

カスタム Groovy スクリプト

設定ファイル

これらの表には、カスタム Groovy スクリプト ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 35 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/groovy/input/groovy_Input.csv`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 36 : scriptExecuterConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`CustomScript` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `CustomScriptSample` です。
pb.bdq.dim.date.pattern	*M/d/yy* の形式で、ジョブで使用される日付パターンを指定します。 **注:** これはオプションのプロパティです。
pb.bdq.dim.datetime.pattern	*M/d/yy h:mm a* の形式で、ジョブで使用される日時パターンを指定します。 **注:** これはオプションのプロパティです。
pb.bdq.dim.time.pattern	*h:mm a* の形式で、ジョブで使用される時刻パターンを指定します。 **注:** これはオプションのプロパティです。
pb.bdq.dim.groovy.input.fields.0	`{"name":<"name of the field">,"type":<"field type">}` の形式で、入力フィールドとそのデータ タイプを指定します。 例: `{"name":"AddressLine2","type":"string"}`
pb.bdq.dim.groovy.output.fields.0	`{"name":<"name of the field">,"type":<"field type">}` の形式で、入力フィールドとそのデータ タイプを指定します。 例: `{"name":"AddressLine2","type":"string"}`
pb.bdq.dim.groovy.script.0	実行する Groovy スクリプトのパス。例: `/home/hduser/script/groovy.txt`

表 37 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 38 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/groovy/output`
pb.bdq.outputformat.field.delimiter	カンマ (`,`) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。

パラメータ	説明
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Joiner

設定ファイル

これらの表には、Joiner ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

注： ここでは、Joinier ジョブに 3 つの入力ファイルがあると想定して説明を行っています。ただし、ジョブには任意の数の入力ファイルを使用できます。

表 39 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は TEXT、ORC、PARQUET のいずれかです。
これらの行では、1 つ目の入力ファイルの詳細について説明しています。	
pb.bdq.inputfile.path.0	入力ファイルを HDFS 上に置いたパス。例: `/home/hduser/input/input0.txt`
textinputformat.record.delimiter.0	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、LINUX、MACINTOSH、またはWINDOWS
pb.bdq.inputformat.field.delimiter.0	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier.0	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header.0	カンマで区切られた値としての列ヘッダー。例えば、business name、id、domain。
pb.bdq.inputformat.skip.firstrow.0	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。
これらの行では、2 つ目の入力ファイルの詳細について説明しています。	
pb.bdq.inputfile.path.1	入力ファイルを HDFS 上に置いたパス。例: `/home/hduser/input/input1.txt`
textinputformat.record.delimiter.1	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、LINUX、MACINTOSH、またはWINDOWS
pb.bdq.inputformat.field.delimiter.1	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier.1	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header.1	カンマで区切られた値としての列ヘッダー。例えば、business name、id、domain。

パラメータ	説明
pb.bdq.inputformat.skip.firstrow.1	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。
これらの行では、3 つ目の入力ファイルの詳細について説明しています。	
pb.bdq.inputfile.path.2	入力ファイルを HDFS 上に置いたパス。例: `/home/hduser/input/input2.txt`
textinputformat.record.delimiter.2	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter.2	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier.2	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header.2	カンマで区切られた値としての列ヘッダー。例えば、business name、id、domain。
pb.bdq.inputformat.skip.firstrow.2	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 40 : joinerConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`Joiner` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `JoinerSample` です。
com.pb.bdq.dim.join.left.port	左ポートの入力ファイル インデックスを定義する JSON 文字列。
com.pb.bdq.dim.join.type	実行する結合操作のタイプを指定します。次のオプションがあります。 • `LeftOuter` • 全体 • 内部

パラメータ	説明
com.pb.bdq.dim.join.col.0	結合する列をカンマ (,) で区切って指定します。
com.pb.bdq.dim.join.col.1	結合する列をカンマ (,) で区切って指定します。
com.pb.bdq.dim.join.col.2	結合する列をカンマ (,) で区切って指定します。

表 41 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 42 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が `TEXT`、`ORC`、または `PARQUET` 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/joiner/output`
pb.bdq.outputformat.field.delimiter	カンマ (`,`) またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	

パラメータ	説明
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (True または False)。デフォルトは True です。
parquet.validation	デフォルトの boolean 値は False です。
parquet.writer.version	Writer のバージョンを指定します。PARQUET_1_0 または PARQUET_2_0 を指定する必要があります。デフォルトは PARQUET_1_0 です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は True です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Data Normalization モジュール

Advanced Transformer

Advanced Transformer ジョブは、テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。このコンポーネントは、特定の語、あるいは語の右側または左側から指定した数だけ単語を抽出します。抽出データと非抽出データを既存のフィールドまたは新しいフィールドに配置できます。

例えば、次の住所フィールドから一組の情報を抽出し、その情報を別のフィールドに配置するとします。

2300 BIRCH RD STE 100

これを行うには、語 STE と語 STE の右側にあるすべての単語を抽出する Advanced Transformer を作成することができます、分離されるフィールド：

2300 BIRCH RD

設定ファイル

これらの表には、Advanced Transformer ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 43 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/advancedtransformer/input/AdvancedTransformer_Input.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。

パラメータ	説明
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。
pb.bdq.inputfile.field.mapping	入力ファイルで使用されているヘッダーの値を、更新された値でマップします。 **注:** これはオプションのパラメータです。

表 44 : advancedTransformerConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AdvTransformer` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AdvanceTransformerSample` です。
pb.bdq.dnm.advtransformer.configuration	Advanced Transformer 設定を定義する Json 文字列。これは、スキャンや分割のために評価するソース入力フィールド、抽出されたデータを置く出力フィールド、トークン化する特殊文字、実行する抽出のタイプなどの詳細を指定します。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: {"referenceDataPathLocation":"LocalToDataNodes","dataDir":"/home/data/referenceData"}

表 45 : advancedTransformerConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AdvTransformer` です。

パラメータ	説明
pb.bdq.job.name	ジョブの名前。デフォルトは `AdvanceTransformerSample` です。
pb.bdq.dnm.advtransformer.configuration	Advanced Transformer 設定を定義する Json 文字列。これは、スキャンや分割のために評価するソース入力フィールド、抽出されたデータを置く出力フィールド、トークン化する特殊文字、実行する抽出のタイプなどの詳細を指定します。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例: [illegible]

表 46 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 47 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が `TEXT`、`ORC`、または `PARQUET` 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/advancedtransformer/output`
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。

パラメータ	説明
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。`UNCOMPRESSED`、`SNAPPY`、`GZIP`、`LZO` のいずれかを指定します。 デフォルトは `UNCOMPRESSED` です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。

パラメータ	説明
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Table Lookup

Table Lookup ステージは、語とその語の妥当性確認済みの形式とを照合して正規化し、標準バージョンを適用します。この評価は、正規化する語をテーブルから検索して実行されます。

設定ファイル

これらの表には、Table Lookup ジョブを実行する前に指定する必要があるパラメータと値が記述されています。

表 48 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/tablelookup/input/Tablelookup_Input.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。

パラメータ	説明
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 49 : tableLookupConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`TableLookup` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `TableLookupSample` です。
pb.bdq.dnm.tablelookup.configuration	Table Lookup 設定を定義する Json 文字列。ソース フィールドに対して実行するアクションのタイプ、検索する語を含むフィールド、データフロー内のデータにマッチする語の検索に使用するテーブルなどの詳細を指定します。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: `{"referenceDataPathLocation":"LocaltoDataNodes","dataDir":"/home/data/referenceData"}`

表 50 : tableLookupConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`TableLookup` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `TableLookupSample` です。
pb.bdq.dnm.tablelookup.configuration	Table Lookup 設定を定義する Json 文字列。ソース フィールドに対して実行するアクションのタイプ、検索する語を含むフィールド、データフロー内のデータにマッチする語の検索に使用するテーブルなどの詳細を指定します。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例: [illegible]

表 51 : mapReduceConfig

MapReduce 設定パラメータを指定します。
ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。 **注 :** このファイルは MapReduce ジョブでのみ使用します。

表 52 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/tablelookup/output`
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。

パラメータ	説明
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Open Parser

Open Parser は、世界のさまざまなカルチャーからの入力データを、シンプルかつ強力なパーシング グラマーでパースします。このグラマーを使用すると、ドメイン パターンを表す一連の式を入力データのパース用に定義できます。また、Open Parser は、統計データを収集してパーシング マッチにスコアを付けるため、パーシング グラマーの効果を調べるのも容易です。

Open Parser を使用して、次のことができます。

- ドメインエディタで定義したドメイン固有およびカルチャー固有のパーシング グラマーを使用して、入力データをパースします。
- ドメインエディタで使用できるものと同じ、シンプルかつ強力なパーシング グラマーを使用して Open Parser で定義した、ドメインに依存しないパーシング グラマーを使用して、入力データをパースします。
- データフロー オプションで定義したドメインに依存しないパーシング グラマーを実行時に使用して、入力データをパースします。
- ターゲット入力データファイルを使用してジョブを実行する前に、パーシング グラマーをプレビューして、サンプル入力データがどのようにパースされるかをテストします。
- パーシング グラマー結果をトレースして、定義した式にトークンがどれくらいマッチしたか、またはマッチしなかったかを確認し、マッチング処理に関する理解を深めます。

設定ファイル

これらの表には、Open Parser ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 53 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/opennameparser/input/OpenParser_Input.csv`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`

パラメータ	説明
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 54 : openParserConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`OpenParser` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `OpenParserSample` です。
pb.bdq.dnm.openparser.configuration	オープン パーサーの設定を定義する Json 文字列。使用する構文解析文法、解析するデータの言語やカルチャーなどの詳細を指定します。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: {"referenceDataPathLocation":"LocalToDataNodes","dataDir":"/home/data/referenceData"}

表 55 : openParserConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`OpenParser` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `OpenParserSample` です。
pb.bdq.dnm.openparser.configuration	オープン パーサーの設定を定義する Json 文字列。使用する構文解析文法、解析するデータの言語やカルチャーなどの詳細を指定します。

パラメータ	説明
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例:

表 56 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 57 : OutputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/opennameparser/output`
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
Parquet ファイルのプロパティ	

パラメータ	説明
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (True または False)。デフォルトは True です。
parquet.validation	デフォルトの boolean 値は False です。
parquet.writer.version	Writer のバージョンを指定します。PARQUET_1_0 または PARQUET_2_0 を指定する必要があります。デフォルトは PARQUET_1_0 です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は True です。
parquet.page.size.row.check.min	デフォルト値は 100 です。

パラメータ	説明
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Global Addressing モジュール

Global Address Validation

Global Address Validation は、高度な住所の正規化および検証の機能を提供します。Global Address Validation は Global Addressing モジュールの一部です。

サポートされている国

Global Address Validation は、以下の優先国に対して高度な住所の正規化および検証の機能を提供します。各国に対して 3 桁の ISO 国コードが記載されています。ISO 国コードの全一覧については、**ISO 国コードとコーダー サポート**（417ページ）を参照してください。

- アルゼンチン (ARG)
- オーストラリア (AUS)
- オーストリア (AUT)
- ベルギー (BEL)
- ブラジル (BRA)
- カナダ (CAN)
- 中国 (CHN)
- チェコ共和国 (CHZ)
- デンマーク (DNK)
- フィンランド (FIN)
- フランス (FRA)
- ドイツ (DEU)
- ギリシャ (GRC)
- インド (IND)
- アイルランド (IRL)
- イタリア (ITA)
- 日本 (JPN)

- マレーシア (MYS)
- メキシコ (MEX)
- オランダ (NLD)
- ニュージーランド (NZL)
- ノルウェー (NOR)
- ポーランド (POL)
- ロシア (RUS)
- スペイン (ESP)
- スウェーデン (SWE)
- スイス (CHE)
- 英国 (GBR) (POI 情報を含みます)
- 米国 (USA)

Global Address Validation は上記以外にも、130 カ国を超える世界中の国々をサポートします。

設定ファイル - Address Validation

これらの表には、Global Address Validation ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 58 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/addressing/input/global/Global_Address.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。

パラメータ	説明
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 59 : addressValidationConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/GAM","referenceDataPathLocation": "LocaltoDataNodes"}`
pb.bdq.uam.addressvalidation.input.option	マッチ モード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。
pb.bdq.uam.addressvalidation.engine.configurations	データベース パス、国コード、プロセス タイプなど、エンジン設定を定義する JSON 文字列。

表 60 : usaAddressValidationConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。

パラメータ	説明
pb.bdq.gam.addressvalidation.input.option	マッチモード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。 **注 :** この場合、デフォルトの国は米国 (USA) です。
pb.bdq.gam.addressvalidation.engine.configurations	データベース パス、国コード、プロセス タイプなど、エンジン設定を定義する JSON 文字列。 例: `{"productDatabaseInfoList": [{"referenceDataPath":{"referenceDataPathLocation": "LocaltoDataNodes","dataDir": "/user/hadoop/ReferenceData/GAV_US_DOM"}, "countryCode":["USA"],"processType":"VALIDATE"}]}`

表 61 : addressValidationConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータダウンローダのパス。例: [illegible]
pb.bdq.uam.addressvalidation.input.option	マッチモード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。
pb.bdq.uam.addressvalidation.engine.configurations	データベースパス、国コード、プロセスタイプなど、エンジン設定を定義する JSON 文字列。

表 62 : usaAddressValidationConfigHDFSRefData_DataDownloader

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。
pb.bdq.gam.addressvalidation.input.option	マッチモード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。 **注 :** この場合、デフォルトの国は米国 (USA) です。
pb.bdq.gam.addressvalidation.engine.configurations	データベースパス、国コード、プロセスタイプなど、エンジン設定を定義する JSON 文字列。例: `{"productDatabaseInfoList": [{"referenceDataPath":{"referenceDataPathLocation":"HDFS","dataDir": "/user/hadoop/RefrenceData/AddressValidation", "dataDownloader":{"dataDownloader":"HDFS", "localFSRepository": "/opt/PitneyBowes/ReferenceData/AddressValidation"}}, "countryCode":["USA"],"processType":"VALIDATE"}]}`

表 63 : addressValidationConfigDistributedCache

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータダウンローダのタイプ。例: {"dataDownloader":{"dataDownloader":"HDFS"}}

パラメータ	説明
pb.bdq.uam.addressvalidation.input.option	マッチ モード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。
pb.bdq.uam.addressvalidation.engine.configurations	データベース パス、国コード、プロセス タイプなど、エンジン設定を定義する JSON 文字列。

表 64 : usaAddressValidationConfigDistributedCache

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`AddressValidation` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `AddressValidationSample` です。
pb.bdq.gam.addressvalidation.input.option	マッチ モード、結果の大文字と小文字の区別、デフォルトの国など、入力設定を定義する JSON 文字列。 **注 :** この場合、デフォルトの国は米国 (USA) です。
pb.bdq.gam.addressvalidation.engine.configurations	データベース パス、国コード、プロセス タイプなど、エンジン設定を定義する JSON 文字列。例: `{"productDatabaseInfoList": [{"referenceDataPath":{"referenceDataPathLocation" :"HDFS","dataDir":"/RefrenceData/AddressValidation", "dataDownloader":{"dataDownloader":"DC"}}, "countryCode":["USA"],"processType":"VALIDATE"}]}` **注 :** 国コードは "USA"
pb.bdq.uam.input.groupby.region	リファレンス データが HDFS に配置されている場合、入力データが地域 (APAC、EMEA、アメリカ) 別にグループ化されているかどうかが指定されます。 **注 :** この場合、値は "false" です。

表 65 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 66 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: /user/hduser/sampledata/addressing/output/global
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。Trueは、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が false の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。

パラメータ	説明
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Universal Addressing モジュール

Validate Address

Validate Address は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州または省名など、欠落している郵便情報を追加します。

> **注：** 現在、Validate Address では米国住所のみがサポートされています。

Validate Address は、住所の妥当性確認の有無、返された住所の確信レベル、住所の妥当性が確認できなかった場合の理由など、検証処理に関する結果インジケータも返します。

Validate Address は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを **Universal Addressing モジュール**の各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて*正規化*します。データベースにマッチしなかった場合、Validate Address は、オプションで入力住所の*書式を整えます*。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。

> **注：** Validate Address は CASS 認定™処理をサポートしており、これにより USPS® の郵便料金値引きのための条件を揃えることができます。

設定ファイル

これらの表には、Validate Address ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 67 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/home/hadoop/uamus.txt`

パラメータ	説明
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、LINUX、MACINTOSH、またはWINDOWS
pb.bdq.inputformat.field.delimiter	カンマ (,) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は True または False です。True はスキップを示します。

表 68 : uamusConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、UniversalAddressingValidate です。
pb.bdq.job.name	ジョブの名前。デフォルトは UAMUniversalAddressingSample です。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: {"dat[illegible]}
pb.bdq.uam.universaladdress.input.configuration	プロセス タイプ、出力住所の要素、レポート リストの数など、入力設定を定義する JSON 文字列。
pb.bdq.uam.universaladdress.general.configuration	ファイル タイプ、メモリ モデル、SuiteLink メモリ モデルなど、全般的な設定を定義する JSON 文字列。
pb.bdq.uam.universaladdress.cobol.runtime	COBOL 実行時ディレクトリ パス。例: /home/hduser/PBSpectrum_BigDataSDK/SDK/runtime
pb.bdq.uam.universaladdress.modules.dir	モジュール ディレクトリのあるパス。例: /home/hduser/PBSpectrum_BigDataSDK/SDK/modules

パラメータ	説明
pb.bdq.uam.universaladdress.dpv.db.path	Delivery Point Validation (DPV) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.ews.db.path	早期警告システム (EWS) データベースのパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.lacs.db.path	Locatable Address Conversion System (LACS) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.rdi.db.path	Residential Delivery Indicator (RDI) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.suitelink.db.path	SuiteLink データベース パス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.job.report.create	正常に完了したときにレポートを生成する場合は `true` を指定します。

表 69 : uamusConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`UniversalAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `UAMUniversalAddressingSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例:
pb.bdq.uam.universaladdress.input.configuration	プロセス タイプ、出力住所の要素、レポート リストの数など、入力設定を定義する JSON 文字列。
pb.bdq.uam.universaladdress.general.configuration	ファイル タイプ、メモリ モデル、SuiteLink メモリ モデルなど、全般的な設定を定義する JSON 文字列。
pb.bdq.uam.universaladdress.cobol.runtime	COBOL 実行時ディレクトリ パス。例: `/home/hduser/PBSpectrum_BigDataSDK/SDK/runtime`
pb.bdq.uam.universaladdress.modules.dir	モジュール ディレクトリのあるパス。例: `/home/hduser/PBSpectrum_BigDataSDK/SDK/modules`
pb.bdq.uam.universaladdress.dpv.db.path	Delivery Point Validation (DPV) データベースのあるパス。例: `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.ews.db.path	早期警告システム (EWS) データベースのパス。例: `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip` **注:** このパラメータはオプションです。

パラメータ	説明
pb.bdq.uam.universaladdress.lacs.db.path	Locatable Address Conversion System (LACS) データベースのあるパス。例: `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.rdi.db.path	Residential Delivery Indicator (RDI) データベースのあるパス。例: `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/RDI.zip` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.suitelink.db.path	SuiteLink データベース パス。例: `hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip` **注:** このパラメータはオプションです。
pb.bdq.job.report.create	正常に完了したときにレポートを生成する場合は`true`を指定します。

表 70 : uamusConfigDistributedCache

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`UniversalAddressingValidate`です。
pb.bdq.job.name	ジョブの名前。デフォルトは`UAMUniversalAddressingSample`です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのタイプ。例: {"dat[illegible]":"0"}
pb.bdq.uam.universaladdress.input.configuration	プロセス タイプ、出力住所の要素、レポート リストの数など、入力設定を定義する JSON 文字列。

パラメータ	説明
pb.bdq.uam.universaladdress.general.configuration	ファイル タイプ、メモリ モデル、SuiteLink メモリ モデルなど、全般的な設定を定義する JSON 文字列。
pb.bdq.uam.universaladdress.acushare.license	Acushare ライセンス ファイルを置いたパス。例: `/home/hduser/runcbl.alc`
pb.bdq.uam.universaladdress.acushare.service	値が `true` の場合、Acushare サービスが実行中であることを示します。
pb.bdq.uam.universaladdress.unix.version	クラスタ ノードの Unix バージョンを指定します。例: `REDHAT7`。
pb.bdq.uam.universaladdress.cobol.runtime	COBOL 実行時ディレクトリ パス。例: `/home/hduser/PBSpectrum_BigDataSDK/SDK/runtime`
pb.bdq.uam.universaladdress.modules.dir	モジュール ディレクトリのあるパス。例: `/home/hduser/PBSpectrum_BigDataSDK/SDK/modules`
pb.bdq.uam.universaladdress.dpv.db.path	Delivery Point Validation (DPV) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.ews.db.path	早期警告システム (EWS) データベースのパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.lacs.db.path	Locatable Address Conversion System (LACS) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。

パラメータ	説明
pb.bdq.uam.universaladdress.rdi.db.path	Residential Delivery Indicator (RDI) データベースのあるパス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.uam.universaladdress.suitelink.db.path	SuiteLink データベース パス。例: `/home/hduser/ReferenceData/AddressQuality/UAM/Data` **注:** このパラメータはオプションです。
pb.bdq.job.report.create	正常に完了したときにレポートを生成する場合は `true` を指定します。

表 71 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注: このファイルは MapReduce ジョブでのみ使用します。

表 72 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が `TEXT`、`ORC`、または `PARQUET` 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/home/hadoop/output`。
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。

パラメータ	説明
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。`True`は、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が `false` の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。`UNCOMPRESSED`、`SNAPPY`、`GZIP`、`LZO` のいずれかを指定します。 デフォルトは `UNCOMPRESSED` です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。

パラメータ	説明
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Validate Address Global

Validate Address Global は、米国およびカナダ以外の住所に対する高度な住所の正規化および検証機能を提供します。Validate Address Global は、米国およびカナダの住所の妥当性も確認できますが、その他の国の住所の妥当性を確認する能力に優れています。米国およびカナダ以外の住所を大量に処理する場合は、Validate Address Global の使用を検討してください。

Validate Address Global は Universal Addressing モジュールの一部です。

Validate Address Global は、パーシング、検証、書式設定など、いくつもの手順を実行して、住所の品質を高めています。

住所のパーシング、書式設定、および正規化

住所データのフィールド入力の誤りを再構成することは、特に他国の住所で行う場合、複雑で難しい作業です。住所データをコンピュータのシステムに入力する際、曖昧になってしまう部分が多いからです。特に問題なのが、(企業や個人名を通りの住所フィールドに入力するなど) 要素を誤ったフィールドに入力したり、省略形を使用する場合に、言語固有だけでなく、国固有の省略形に変えてしまうケースです。Validate Address Global は住所行の住所要素を識別し、正しいフィールドに割り当てます。これは実際の検証前に行う重要な作業です。再構成を行わなければ、"一致が見つからない" という結果になる可能性があります。

住所要素の正しい識別は、特定のフィールド長要件に合わせて住所を切り捨てたり、短縮しなければならない場合にも重要です。正しい情報が正しいフィールドに割り当てられていれば、特定の切り捨てルールを適用することができます。

- 住所行をパースおよび解析し、個々の住所要素を識別
- 30 を越える文字セットを処理
- 宛先国の郵便ルールに従って住所の書式を整える
- 住所要素を正規化 (AVENUE を AVE に変更するなど)

Global Address Validation

住所の検証は、正しくパースされた住所データを郵便組織または他のデータ プロバイダが提供する参照データベースと比較する訂正処理です。Validate Address Global は、洗練されたファジー マッチング テクノロジーを使用して個々の住所要素を検証し、正しいことを確認するとともに、郵便規格とユーザの優先設定に基づいて出力を正規化および書式設定します。FastCompletion 検証タイプは、簡易住所入力アプリケーションに使用できます。いくつかの住所フィールドには切り捨てられたデータを入力することができ、この入力に基づいて提案を生成します。

住所を完全に検証できない場合もあります。Validate Address Global には、配達可能性によって住所を分類する、ユニークな配達可能性評価機能があります。

設定ファイル

これらの表には、Validate Address Global ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 73 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/addressing/input/global/Global_Address.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。

パラメータ	説明
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 74 : globalAddressingConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`GlobalAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `GlobalAddressingValidateSample` です。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: `{"dataDir":"/home/hduser/ReferenceData/AddressQuality/Global","referenceDataPathLocation":"LocaltoDataNodes"}`
pb.bdq.uam.global.engine.configurations.preload	グローバル エンジン設定のプリロード タイプ。値は、`NONE`、`FULL`、`PARTIAL` のいずれかです。
pb.bdq.uam.global.engine.configurations.database.type	グローバル エンジン設定のデータベース タイプ。値は、`BATCH_INTERACTIVE`、`FASTCOMPLETION`、`CERTIFIED` のいずれかです。
pb.bdq.uam.global.engine.configurations.supported.countries	United States Of America、Great Britain、Canada など、Global Address Validation ジョブでサポートされている国。 **注 :** 複数の国の値をカンマで区切って指定できます。
pb.bdq.uam.global.input.configuration	マッチ モード、デフォルトの国、結果の最大数、結果の大文字と小文字の区別、州/省タイプ、最適化レベルなど、入力設定を定義する JSON 文字列。
pb.bdq.uam.global.general.configuration	キャッシュ サイズ、最大スレッド数、メモリ使用量の上限など、全般的な設定を定義する JSON 文字列。

パラメータ	説明
pb.bdq.uam.global.unlockCode	データベースのデータのロックを解除するコード。

表 75 : globalAddressingConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`GlobalAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `GlobalAddressingValidateSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例: {"referenceDataPathLocation":"HDFS","dataDir":"/user/hduser/ReferenceData/Global/Global.zip","dataDownloader":{"dataDownloader":"HDFS", "localFSRepository":"/opt/PitneyBowes/ReferenceData/GlobalAddress"}}
pb.bdq.uam.input.groupby.region	*APAC*、*EMEA*、*America* など、入力住所データを地域別にグループ化するかどうかを指定します。 値が `true` の場合、グループ化することを示します。 **注:** このパラメータは、HDFS にリファレンス データが配置されている場合にのみ適用されます。
pb.bdq.uam.global.engine.configurations.preload	グローバル エンジン設定のプリロード タイプ。値は、`NONE`、`FULL`、`PARTIAL` のいずれかです。
pb.bdq.uam.global.engine.configurations.database.type	グローバル エンジン設定のデータベース タイプ。値は、`BATCH_INTERACTIVE`、`FASTCOMPLETION`、`CERTIFIED` のいずれかです。
pb.bdq.uam.global.engine.configurations.supported.countries	United States Of America、Great Britain、Canada など、Global Address Validation ジョブでサポートされている国。 **注:** 複数の国の値をカンマで区切って指定できます。

パラメータ	説明
pb.bdq.uam.global.input.configuration	マッチ モード、デフォルトの国、結果の最大数、結果の大文字と小文字の区別、州/省タイプ、最適化レベルなど、入力設定を定義する JSON 文字列。
pb.bdq.uam.global.general.configuration	キャッシュ サイズ、最大スレッド数、メモリ使用量の上限など、全般的な設定を定義する JSON 文字列。
pb.bdq.uam.global.unlockCode	データベースのデータのロックを解除するコード。

表 76 : globalAddressingConfigDistributedCache

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`GlobalAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `GlobalAddressingValidateSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのタイプ。例:
pb.bdq.uam.global.engine.configurations.preload	グローバル エンジン設定のプリロード タイプ。値は、`NONE`、`FULL`、`PARTIAL` のいずれかです。
pb.bdq.uam.global.engine.configurations.database.type	グローバル エンジン設定のデータベース タイプ。値は、`BATCH_INTERACTIVE`、`FASTCOMPLETION`、`CERTIFIED` のいずれかです。
pb.bdq.uam.global.engine.configurations.supported.countries	United States Of America、Great Britain、Canada など、Global Address Validation ジョブでサポートされている国。 **注 :** 複数の国の値をカンマで区切って指定できます。
pb.bdq.uam.global.input.configuration	マッチ モード、デフォルトの国、結果の最大数、結果の大文字と小文字の区別、州/省タイプ、最適化レベルなど、入力設定を定義する JSON 文字列。

パラメータ	説明
pb.bdq.uam.global.general.configuration	キャッシュ サイズ、最大スレッド数、メモリ使用量の上限など、全般的な設定を定義する JSON 文字列。
pb.bdq.uam.global.unlockCode	データベースのデータのロックを解除するコード。

表 77 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 78 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が `TEXT`、`ORC`、または `PARQUET` 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/addressing/output/global`
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。`True`は、コンソールにカウンタを出力することをます。

パラメータ	説明
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が `false` の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。`UNCOMPRESSED`、`SNAPPY`、`GZIP`、`LZO` のいずれかを指定します。 デフォルトは `UNCOMPRESSED` です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。

パラメータ	説明
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Validate Address Loqate

Validate Address Loqate は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。Validate Address Loqate は、情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州/省名など、欠落している郵便情報を追加します。

Validate Address Loqate は、Validate Address Loqate が住所の妥当性を確認したかどうか、返された住所の確信レベル、住所の妥当性が確認できなかった場合はその理由など、検証処理に関する結果インジケータも返します。

Validate Address Loqate は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを Universal Addressing モジュールの各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて正規化します。データベースにマッチしなかった場合、Validate Address Loqate は、オプションで入力住所の書式を整えます。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。ValidateAddressLoqate は、Universal Addressing モジュールに含まれています。

設定ファイル

これらの表には、Validate Address Loqate ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 79 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。

パラメータ	説明
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/addressing/input/loqate/loqate_input.txt`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 80 : loqateAddressingConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`LoqateAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `LoqateAddressingValidateSample` です。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: {illegible}
pb.bdq.uam.loqate.process.configuration	プロセス タイプ、最小マッチ スコア、デフォルトの国、許容レベルなど、検証設定を定義する JSON 文字列。
pb.bdq.uam.loqate.engine.configuration	ツール情報、ログ ファイル名、マッチ スコアのしきい値係数など、エンジン設定を定義する JSON 文字列。

パラメータ	説明
pb.bdq.uam.loqate.general.configuration	最大アイドルオブジェクト数、最小アイドルオブジェクト数、最長待ち時間など、全般的な設定を定義する JSON 文字列。

表 81 : loqateAddressingConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`LoqateAddressingValidate` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `LoqateAddressingValidateSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータダウンローダのパス。例:
pb.bdq.uam.loqate.process.configuration	プロセス タイプ、最小マッチ スコア、デフォルトの国、許容レベルなど、検証設定を定義する JSON 文字列。
pb.bdq.uam.loqate.engine.configuration	ツール情報、ログ ファイル名、マッチ スコアのしきい値係数など、エンジン設定を定義する JSON 文字列。
pb.bdq.uam.loqate.general.configuration	最大アイドルオブジェクト数、最小アイドルオブジェクト数、最長待ち時間など、全般的な設定を定義する JSON 文字列。

表 82 : loqateAddressingConfigDistributedCache

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`LoqateAddressingValidate` です。

パラメータ	説明
pb.bdq.job.name	ジョブの名前。デフォルトは `LoqateAddressingValidateSample` です。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのタイプ。例: {[illegible]}
pb.bdq.uam.loqate.process.configuration	プロセス タイプ、最小マッチ スコア、デフォルトの国、許容レベルなど、検証設定を定義する JSON 文字列。
pb.bdq.uam.loqate.engine.configuration	ツール情報、ログ ファイル名、マッチ スコアのしきい値係数など、エンジン設定を定義する JSON 文字列。
pb.bdq.uam.loqate.general.configuration	最大アイドル オブジェクト数、最小アイドル オブジェクト数、最長待ち時間など、全般的な設定を定義する JSON 文字列。

表 83 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 84 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が `TEXT`、`ORC`、または `PARQUET` 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: `/user/hduser/sampledata/addressing/output/loqate`。

パラメータ	説明
pb.bdq.outputformat.field.delimiter	カンマ(`,`)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が `true` のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、`true` を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。`True`は、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が `false` の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。`UNCOMPRESSED`、`SNAPPY`、`GZIP`、`LZO` のいずれかを指定します。 デフォルトは `UNCOMPRESSED` です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。

パラメータ	説明
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

Universal Name モジュール

OpenNameParser

OpenNameParser は、名前データ フィールドにある個人名、企業名、またはその他の名称を構成要素に分解します。これらのパースされた名前要素は、名前のマッチング、名前の正規化、複数レコード名の統合など、他の自動化処理に使用できます。

OpenNameParser は、次の処理を行います。

- 名前が担う機能を示すために、その名前のタイプを特定します。名前エンティティ タイプは、個人名と企業名の 2 つのグループに分かれます。それぞれのグループには、さらに複数のサブグループがあります。
- パーシングに使う構文を把握するために、名前の形式を特定します。個人名は、通常、自然な (署名) 順序または逆の順序に従います。企業名は、通常、階層型の順序に従います。

- 名前を構成する各要素が名前全体に占める構文上の関連性を識別するために、要素を特定してラベル付けします。個人名の構文は、敬称、名、ミドルネーム、姓、接尾語、アカウントを示す用語、その他の個人名要素で構成されます。企業名の構文は、企業名や接尾語などで構成されます。
- 結合された個人名と企業名をパースし、それらを 1 つのレコードとして残すか、複数のレコードに分割します。Examples of conjoined names include "Mr.and Mrs.John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- 出力をレコードまたはリストとしてパースします。
- パーシングによる訂正の信頼度を示すパーシング スコアを割り当てます。

設定ファイル

これらの表には、Open Name Parser ジョブを実行する前に指定する必要があるパラメータと値が記載されています。

表 85 : inputFileConfig

パラメータ	説明
pb.bdq.input.type	入力ファイルの種類。値は `TEXT`、`ORC`、`PARQUET` のいずれかです。
pb.bdq.inputfile.path	入力ファイルを HDFS 上に置いたパス。例: `/user/hduser/sampledata/opennameparser/input/OpenNameParser_Input.csv`
textinputformat.record.delimiter	テキスト タイプの入力ファイルで使用されるファイル レコード区切り文字。例えば、`LINUX`、`MACINTOSH`、または`WINDOWS`
pb.bdq.inputformat.field.delimiter	カンマ (`,`) またはタブなど、入力ファイルで使用されるフィールドまたは列の区切り文字。
pb.bdq.inputformat.text.qualifier	入力ファイルの列またはフィールドのテキスト修飾子 (存在する場合)。
pb.bdq.inputformat.file.header	入力ファイルで使用されるヘッダーのカンマ区切りの値。
pb.bdq.inputformat.skip.firstrow	最初の行をスキップするかどうか。値は `True` または `False` です。`True` はスキップを示します。

表 86 : openNameParserConfig

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`OpenNameParser` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `OpenNameParserSample` です。
pb.dq.unm.opennameparser.configuration	解析する名前の種類など、Open Name Parser 設定を定義する JSON 文字列。
pb.bdq.reference.data	リファレンス データを配置している場所のパス。例: {"referenceDataPathLocation":"LocalToDataNodes","dataDir":"/home/data/referenceData"}

表 87 : openNameParserConfigHDFSRefData(DataDownloader)

パラメータ	説明
pb.bdq.job.type	これは、ジョブを定義する定数値です。このジョブの値は、`OpenNameParser` です。
pb.bdq.job.name	ジョブの名前。デフォルトは `OpenNameParserSample` です。
pb.dq.unm.opennameparser.configuration	解析する名前の種類など、Open Name Parser 設定を定義する JSON 文字列。
pb.bdq.reference.data	HDFS 上にあるリファレンス データのパスとデータ ダウンローダのパス。例: {"referenceDataPathLocation":"HDFS","dataDir":"/home/data/referenceData","dataDownloader":{"dataDownloader":"HDFS","localProxy":"/local/data"}}

表 88 : mapReduceConfig

MapReduce 設定パラメータを指定します。

ジョブでの必要性に応じて、*mapreduce.map.memory.mb*、*mapreduce.reduce.memory.mb*、*mapreduce.map.speculative* などの MapReduce パラメータをカスタマイズします。

注 : このファイルは MapReduce ジョブでのみ使用します。

表 89 : outputFileConfig

パラメータ	説明
pb.bdq.output.type	出力が TEXT、ORC、または PARQUET 形式の場合に指定します。
pb.bdq.outputfile.path	HDFS で出力ファイルを生成するパス。例: /user/hduser/sampledata/opennameparser/output。
pb.bdq.outputformat.field.delimiter	カンマ(,)またはタブなどの出力ファイルのフィールドまたは列の区切り文字。
pb.bdq.output.overwrite	値が true のとき、出力フォルダはジョブが実行されるたびに上書きされます。
pb.bdq.outputformat.headerfile.create	出力ファイルにヘッダーが必要な場合は、true を指定します。
pb.bdq.job.print.counters.console	カウンタをコンソールまたはファイルに出力するかどうか。Trueは、コンソールにカウンタを出力することをます。
pb.bdq.job.counter.file.path	カウンタが出力されるパスとファイルの名前。*pb.bdq.job.print.counters.console* の値が false の場合は、これを指定する必要があります。
Parquet ファイルのプロパティ	
parquet.compression	ページの圧縮に使用する圧縮アルゴリズム。UNCOMPRESSED、SNAPPY、GZIP、LZO のいずれかを指定します。 デフォルトは UNCOMPRESSED です。
parquet.block.size	メモリにバッファリングされる行グループのサイズ。 値を大きくするほど読み込み時の *I/O* が向上しますが、書き込み時のメモリ消費が大きくなります。 デフォルトのサイズは 134217728 バイト (= 128 * 1024 * 1024) です。

パラメータ	説明
parquet.page.size	ページはブロックを構成し、単一レコードにアクセスするために完全に読み込む必要がある最小単位です。 デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。 **注：** ページ サイズが小さすぎると、圧縮に支障が生じます。
parquet.dictionary.page.size	デフォルトのサイズは 1048576 バイト (= 1 * 1024 * 1024) です。
parquet.enable.dictionary	辞書エンコーディングの有効または無効を指定する boolean 値 (`True` または `False`)。デフォルトは `True` です。
parquet.validation	デフォルトの boolean 値は `False` です。
parquet.writer.version	Writer のバージョンを指定します。`PARQUET_1_0` または `PARQUET_2_0` を指定する必要があります。デフォルトは `PARQUET_1_0` です。
parquet.writer.max-padding	デフォルト設定は、パディングなし、行グループ サイズの 0% です。
parquet.page.size.check.estimate	デフォルトの boolean 値は `True` です。
parquet.page.size.row.check.min	デフォルト値は 100 です。
parquet.page.size.row.check.max	デフォルト値は 10000 です。

6 - Hive ユーザ定義関数

このセクションの構成

はじめに	281
Advanced Matching モジュールの関数	289
Data Integration モジュールの機能	318
Data Normalization モジュールの関数	321
Global Addressing モジュールの関数	335
Universal Addressing モジュールの機能	358
Universal Name モジュールの関数	378

はじめに

Apache Hive は、ユーザ定義関数 (UDF) を提供します。UDF を定義して、必要なアクションを実行し、所望の目的を達成することができます。

Spectrum™ Data & Address Quality for Big Data SDK では、以下のデータ品質ジョブを実行するための一連の Hive ユーザ定義関数とユーザ定義集約関数が提供されています。

ユーザ定義関数 (UDF)

ユーザ定義関数は、一度に 1 つのレコードを処理します。

UDF に基づくジョブには以下のものがあります。

- Advanced Transformer
- カスタム Groovy スクリプト
- Global Address Validation
- Match Key Generator
- Open Name Parser
- Open Parser
- Table Lookup
- Validate Address
- Validate Address Global
- Validate Address Loqate
- Candidate Finder

ユーザ定義集約関数 (UDAF)

ユーザ定義集約関数は、結合フィールドに基づいてレコードをコレクションに集約してから、一度に 1 つのレコード コレクションを処理します。

UDAF に基づくジョブには以下のものがあります。

- Best of Breed
- Duplicate Synchronization
- フィルタ
- Interflow Match
- Intraflow Match
- Transactional Match

ユーザ定義表関数 (UDTF)

ユーザ定義表関数は、入力として 1 つの行を操作し、出力として複数の行を返します。この関数に基づくジョブには以下のものがあります。

- Candidate Finder

Hive 関数のコンポーネント

Spectrum™ Data & Address Quality for Big Data SDK Hive UDF の実行に必要な主要コンポーネントは、以下のとおりです。

JAR ファイル	必要なデータ品質 Hive UDF が属するモジュールの Spectrum™ Data & Address Quality for Big Data SDK Hive JAR ファイル。いずれかの UDF を使用する前に、これが登録されている必要があります。
ジョブ UDF / UDAF	各データ品質ジョブは、ユーザ定義関数 (UDF) またはユーザ定義集約関数 (UDAF) として提供されます。
エイリアス	Hive UDF に割り当てられたエイリアス。この手順は省略可能です。
設定	実行するジョブに基づく、JSON 形式で指定されたルールとその他の環境設定詳細情報。
リファレンス データ	リファレンス データは、Hadoop Distributed File System (HDFS) またはクラスタ マシンに保存できます。 HDFS の場合、リファレンス データの保存には、以下の 2 つの形式を使用できます。 • ファイル • アーカイブ ローカルに保存する場合、リファレンス データは、クラスタの各ノードで、同じパスに保存する必要があります。
ヘッダー	入力テーブルのヘッダー フィールド (カンマ区切り形式)。
入力テーブル	実行する Hive UDF ごとに入力レコードを提供するテーブル。
候補テーブル	Interflow Match UDAF の場合、実行する Hive UDF に候補レコードを提供するテーブル。
サスペクト テーブル	Interflow Match UDAF の場合、実行する Hive UDF にサスペクト レコードを提供するテーブル。
hive.fetch.task.conversion	選択クエリを単一の FETCH タスクに変換し、遅延を最小化します。 値を `none` または `minimal` に設定します。デフォルトは `minimal` です。 **注 :** この設定はすべての UDF で必要です。

hive.map.aggr

Mapper および Reducer 間でのデータの集約をオンまたはオフにするには、この Hive 環境変数を `false` に設定します。デフォルトでは、`true` となっており、データは集約されます。

SDK 内のすべての Hive ジョブで、この値を `false` に設定します。

注： この設定はすべての UDAF で必要です。

全般的な設定

ジョブを実行するために必要なメモリ設定。

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

入力設定

入力データの設定。

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

エンジン設定

データベース設定、*COBOL 実行時パス*、*プリロード タイプ*など、さまざまな設定を行います。

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

LD_LIBRARY_PATH

この環境変数は、Hive ジョブの実行時に必要なさまざまな COBOL ライブラリへのパスに設定します。

注： この設定は Validate Address の Hive UDF でのみ必要です。

プロセス タイプ

SDK の特定の Hive ジョブで使用される適切な検証レベルを指定します。現時点では、住所検証のみがサポートされています。

この値は `VALIDATE` に設定します。

注： この設定は、Validate Address および Validate Address Loqate の Hive UDAF でのみ必要です。

出力

Hive UDF の出力。コンソールに表示されるか、出力ファイルに書き出されます。

クエリ	必要な Hive UDF を実行するクエリ。 各ジョブに対し、適切なクエリ構文を用いてこれらの操作を実行できます。 • ジョブの出力をコンソールに表示する。 • 指定された出力ファイルに出力を書き出す。

Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. 必要なデータ品質 Hive UDF が属する特定の Spectrum™ Data & Address Quality for Big Data SDK モジュールの JAR ファイルを登録します。
3. Validate Address の UDF の場合、COBOL ライブラリのパスを設定するには、環境変数 `LD_LIBRARY_PATH` を次のように設定します。

```
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. Validate Address Global の UDF の場合は、*libAddressDoctor5.so* ファイルも追加します。
5. Validate Address Loqate の UDF の場合は、以下の必須ファイルを分散キャッシュに追加します。
   - `loqate-core.car`
   - `LoqateVerificationLevel.csv`
   - `Loqate.csv`
   - `countryTables.csv`
   - `countryNameTables.csv`
6. 実行するデータ品質ジョブの Hive UDF のエイリアスを作成します。
   例:

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

7. リファレンス データのパスを指定します。

- リファレンス データを HDFS で使用する場合は、以下に示すように、リファレンス データを追加し、リファレンス ディレクトリを設定します。

  リファレンス データが、*ファイル*形式の場合:

```
hdfs://<HOST>:<PORT>/home/hduser/Refdata/;
set hivevar:refdir='./Refdata';
```

  リファレンス データが、*アーカイブ*形式の場合:

```
hdfs://<HOST>:<PORT>/home/hduser/ref.zip;
set hivevar:refdir='./ref.zip';
```

- リファレンス データをローカル パスで使用する場合は、クラスタの各ノードで、同じパスに保存する必要があります。

  リファレンス ディレクトリを以下のように設定します:

```
set hivevar:refdir='/home/hadoop/reference/';
```

8. ジョブの環境設定 (マッチ ルール、ソート フィールド、Express マッチ列、その他の詳細) を対応する変数や設定プロパティで指定します。

   注: ルールは JSON 形式である必要があります。

   例を次に示します。

```
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}]}]}';
```

   注: それぞれのジョブ環境設定の設定プロパティを使用してください。例えば、それぞれのサンプル HQL ファイルの随所に記載されている `pb.bdq.match.rule`、`pb.bdq.match.express.column`、`pb.bdq.consolidation.sort.field` などです。

9. 入力テーブルのヘッダー フィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

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

注： HQL ファイルに記載されている設定プロパティを使用します。例えば、`pb.bdq.match.header`、`pb.bdq.consolidation.header` などです。

10. 次の例に示すように、`Hive.Map.Aggr` 環境変数の設定を `false` にして、Reducer と Mapper の間でのデータの集約をオフにします。

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

注： この設定はすべての UDF で必要です。

11. ジョブを実行するための全般的な設定を次の例に示すように設定します。

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

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

12. ジョブを実行するための入力設定を次の例に示すように設定します。

```
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"}}};
```

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

13. ジョブを実行するためのエンジン設定を次の例のように設定します。

```
set pb.bdq.uam.universaladdress.engine.configurations = {
"referenceData":{
"dataDir":"/home/hduser/resources/uam/universaladdress/UAM_universaladdress4.0_Feb15/",
 "referenceDataPathLocation":"LocaltoDataNodes"},
"cobolRuntimePath":"/home/hduser/addressquality/",
"modulesDir":"/home/hduser/tapan/addressquality/modules",
"dpvDbPath":null, "suiteLinkDBPath":null, "ewsDBPath":null,
"rdiDBPath":null, "lacsDBPath":null};
```

注： この設定は Universal Addressing モジュールの Hive UDAF でのみ必要です。

14. 適切な検証レベルを示すようにプロセス タイプを設定します。現時点では住所検証のみがサポートされています。

    例えば、*Validate Address* ジョブでは、*プロセス タイプ*を次のように設定します。

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

注： この設定は、Validate Address および Validate Address Loqate の Hive UDAF でのみ必要です。

15. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

```
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;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
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;
```

注： 先ほど UDF に対して定義したエイリアスを必ず使用してください。

重要： すべての UDAF ジョブで、それぞれの設定プロパティを変数として使用するとともに、該当するサンプル HQL ファイルに示されている入力パラメータを定義します。

例えば、`pb.bdq.match.rule`、`pb.bdq.match.express.column`、`pb.bdq.consolidation.sort.field` などです。

Advanced Matching モジュールの関数

Advanced Matching モジュールの Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。

2. Spectrum™ Data & Address Quality for Big Data SDK AMM モジュールの JAR ファイルを登録します。

```
ADD JAR <Directory path>/amm.hive.${project.version}.jar;
```

3. 実行するデータ品質ジョブの Hive UDF のエイリアスを作成します。

   注： 引用符で囲まれている文字列は、このジョブの実行に必要なクラス名です。

   例:

```
CREATE TEMPORARY FUNCTION bestofbreed as
'com.pb.bdq.amm.process.hive.consolidation.bestofbreed.BestOfBreedUDAF';
```

4. 次の例に示すように、`Hive.Map.Aggr` 環境変数の設定を `false` にして、Reducer と Mapper の間でのデータの集約をオフにします。

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

   注： この設定はすべての UDF で必要です。

5. ジョブの環境設定と詳細情報を指定して、それぞれの変数または設定プロパティに代入します。

   注： ルールは JSON 形式である必要があります。

   例を次に示します。

```
set hivevar:rule='{"consolidationConditions":
[{"consolidationRule":{"conditionClass":"simpleRule",
"operation":"HIGHEST", "fieldName":"column2", "value":null,
"valueFromField":false, "valueNumeric":true},
"actions":[]}], "removeDuplicates":true}';
```

   注： それぞれのジョブ環境設定の設定プロパティを使用してください。例えば、それぞれのサンプル HQL ファイルに記載されている `pb.bdq.match.rule`、`pb.bdq.match.express.column`、`pb.bdq.consolidation.sort.field` などです。

6. 入力テーブルのヘッダーフィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

```
set hivevar:header ='column1,column2,column3,column4,column5,id';
```

注： 記載されている設定プロパティを使用してください。例えば、それぞれのサンプル HQL ファイルに記載されている `pb.bdq.match.header`、`pb.bdq.consolidation.header` などです。

7. 環境設定プロパティ `'hivevar:sortfield'` を使用して、ソートのパラメータをクエリで使用するエイリアスに設定します。

```
set hivevar:sortfield='id';
```

8. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

```
SELECT tmp2.record["column1"],
 tmp2.record["column2"],
 tmp2.record["column3"],
 tmp2.record["column4"],
 tmp2.record["column5"]
FROM (
 SELECT  filter (${hivevar:rule},
   ${hivevar:sortfield},
   ${hivevar:header},
   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 ;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
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 ;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

Best of Breed

Best of Breed は、重複レコードのコレクションから選択する最良のデータを使用して新しい統合レコードを作成することで、重複レコードを統合します。この "スーパー" レコードは、最良の組み合わせレコードと呼ばれます。処理対象レコードの選択で使用するルールを定義します。処理が完了すると、最良の組み合わせレコードがシステムに保持されます。

サンプル Hive スクリプト

```
-- 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 'hivevar:rule'

set hivevar: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 'hivevar:header'
set hivevar: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 'hivevar:sortField'
set hivevar:sortField='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(${hivevar:rule},
  ${hivevar:sortField},
  ${hivevar:header},
  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      |
```

```
-+--------+--------+--------+--------+--------+--------+--------+--------+--------+--------+----------------+
```

Candidate Finder

Candidate Finder は、一連の潜在的なマッチを形成する候補レコードを取得します。検索インデックス検索は Transactional Match とは別に機能します。また、Candidate Finder では、データの書式によっては、サスペクト レコード、候補レコード、またはその両方のレコードの名前や住所のパーシングが必要となる場合もあります。

また、Candidate Finder ではフルテキスト インデックス検索も可能で、さまざまな検索タイプ (数値、範囲、すべて含む、いずれも含まない) と条件 (すべて真、いずれかが真) を使用して、文字やテキストの高度な検索条件を容易に定義できます。

> **注：** 検索インデックスを保存するには、クラスタで *HBase NoSQL* データベースが利用でき、アクセス可能であることが必要です。

サンプル Hive スクリプト

ユーザ定義関数

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

CREATE TEMPORARY FUNCTION search as
'com.pb.bdq.amm.process.hive.search.CandidateFinderUDF';

--set hive.fetch.task.conversion=none;
set hivevar:pb.bdq.amm.search.cf.index.name='timeGen';
set hivevar:pb.bdq.amm.search.cf.query.json=
'{"type":"complex","queryName":"P1",
"searchQueries":[{"type":"ContainsAllSearchQuery","queryName":null,"boost":1.0,
"indexFieldName":"MonthNumber","firstSearchField":{"name":"IN_MonthNumber",
"type":"STRING"},"ignoreBlanks":true}],"operator":"and"}';
set hivevar:pb.bdq.amm.search.cf.max.results=1;
set hivevar:pb.bdq.amm.search.cf.fetch.batchsize=10000;
set hivevar:pb.bdq.amm.search.cf.start.record=1;
set
hivevar:pb.bdq.amm.search.cf.index.output.fields='MonthNumber,DayOfMonth,MonthName,
WeekdayName';
set hivevar:pb.bdq.amm.search.cf.input.header='IN_MonthNumber';
--set hive.plan.serialization.format=javaXML;
--set hbase.zookeeper.quorum=;
--set hbase.zookeeper.property.clientPort=;
```

```
select recordid,search(
${hivevar:pb.bdq.amm.search.cf.index.name},
${hivevar:pb.bdq.amm.search.cf.query.json},
${hivevar:pb.bdq.amm.search.cf.max.results},
${hivevar:pb.bdq.amm.search.cf.fetch.batchsize},
${hivevar:pb.bdq.amm.search.cf.start.record},
${hivevar:pb.bdq.amm.search.cf.index.output.fields},
${hivevar:pb.bdq.amm.search.cf.input.header},
recordid
)from sample.busniessnames where recordid = 1;

!q
```

ユーザ定義表関数

```
-- Register Advance Matching Module[AMM] Hive UDF jar
ADD JAR <Directory path>/amm.hive.${project.version}.jar;

CREATE TEMPORARY FUNCTION search as
'com.pb.bdq.amm.process.hive.search.CandidateFinderUDTF';

--set hive.fetch.task.conversion=none;
set pb.bdq.amm.search.cf.index.name=timeGen;
set pb.bdq.amm.search.cf.query.json= {"type":"complex","queryName":"P1",
"searchQueries":[{"type":"ContainsAllSearchQuery","queryName":null,"boost":1.0,
"indexFieldName":"MonthNumber","firstSearchField":{"name":"IN_MonthNumber",
"type":"STRING"},"ignoreBlanks":true}],"operator":"and"};
set pb.bdq.amm.search.cf.max.results=3;
set pb.bdq.amm.search.cf.fetch.batchsize=10000;
set pb.bdq.amm.search.cf.start.record=1;
set
pb.bdq.amm.search.cf.index.output.fields=MonthNumber,DayOfMonth,MonthName,WeekdayName;
set pb.bdq.amm.search.cf.input.header=IN_MonthNumber;
--set hive.plan.serialization.format=javaXML;
--set hbase.zookeeper.quorum=;
--set hbase.zookeeper.property.clientPort=;

select search(recordid)from sample.busniessnames where recordid ="1";

!q
```

Duplicate Synchronization

Duplicate Synchronization は、レコードのコレクションから、そのコレクション内のすべてのレコードの対応するフィールドにコピーするフィールドを指定します。フィールド データをコレクション内の別のレコードにコピーするときに従う必要があるルールを指定できます。処理が完了すると、コレクション内のレコードがすべて保持されます。

サンプル Hive スクリプト

```
-- 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';

-- 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.

CREATE TEMPORARY FUNCTION dupsync as
'com.pb.bdq.amm.process.hive.consolidation.duplicatesync.DuplicateSyncUDAF';

-- This rowid is needed by duplicateSync 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.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'

set hivevar: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 'hivevar:header'
set hivevar:header='column1,column2,column3,column4,column5,id';

-- Set sort field name to alias used in query using
configuration property 'hivevar:sortfield'
set hivevar:sortfield='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 (${hivevar:rule},
  ${hivevar:sortfield},
  ${hivevar:header},
  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|
--+----------+----------+----------+----------+----------+
```

フィルタ

Filter ステージでは、指定したルールに基づいて、レコードをレコードのグループに保持または削除します。

サンプル Hive スクリプト

```
-- 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';

-- Filter 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.

CREATE TEMPORARY FUNCTION filter as
'com.pb.bdq.amm.process.hive.consolidation.filter.FilterUDAF';

-- 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.

-- Disable map side aggregation
set hive.map.aggr = false;

-- Set the rule using configuration property 'hivevar:rule'
set hivevar: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 'hivevar:header'
set hivevar:header='column1,column2,column3,column4,column5,id';

-- Set sort field name to alias used in query using
 configuration property 'hivevar:sortfield'
set hivevar:sortfield='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 (${hivevar:rule},
   ${hivevar:sortfield},
   ${hivevar:header},
   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|
--+----------+----------+----------+----------+----------+
```

Interflow Match

Interflow Match は、2 つの入力レコード ストリーム内の類似するデータ レコード間でマッチを検出します。最初のレコード ストリームはサスペクト レコードのソースで、2 番目のストリームは候補レコードのソースです。

Interflow Match では、マッチ グループ条件 (マッチ キー等) を使用して、特定のサスペクト レコードと重複する可能性があるレコードのグループを識別します。

レポート

Interflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクトレコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Expressマッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。
AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。
UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATE_CANDIDATES	検出された重複候補の総数。
TOTAL_DUPLICATE_SCORE	すべての重複の合計マッチ スコア。

サンプル Hive スクリプト

```
-- 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 'hivevar:rule'

set hivevar: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
'hivevar:suspectheader'
set
hivevar:suspectheader='name,firstname,lastname,matchkey,middlename,recordid';

-- Set the header for candidate table using configuration property
'hivevar:Candidateheader'
set
hivevar:Candidateheader='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 hivevar:sortfield='c_id';
```

```
-- Set the express match column(optional)
set hivevar:expressMatchColumn='matchkey';


-- Optionally, one can also set
'hivevar:intercomparison='returnUniqueCandidates,true'';
set hivevar:intercomparison='returnUniqueCandidates,true';

-- Set sort collection number option for unique records using
 configuration property 'hivevar:collectionNumberZero'
set hivevar:collectionNumberZero='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 innerresult.record ["MatchRecordType"],
 innerresult.record ["MatchScore"],
 innerresult.record ["HasDuplicate"],
 innerresult.record ["CollectionNumber"],
 coalesce(innerresult.record ["ExpressMatched"], ''),
 innerresult.record ["SourceType"],
 innerresult.record ["name"],
 innerresult.record ["firstname"],
 innerresult.record ["lastname"],
 innerresult.record ["matchkey"],
 innerresult.record ["middlename"],
 innerresult.record ["recordid"]
FROM (
 SELECT
interMatch(${hivevar:rule},${hivevar:sortfield},${hivevar:expressMatchColumn},


${hivevar:collectionNumberZero},${hivevar:interComparison},${hivevar:Candidateheader}


${hivevar:Suspectheader},unionresults.id,unionresults.name,unionresults.firstname,

        unionresults.lastname, unionresults.matchkey,
unionresults.middlename,
        unionresults.recordid ,unionresults.TYPE)
       AS matchgroup
    FROM (
 SELECT rowid(*) AS id, 'Suspect' AS TYPE,fullname as name,fname
    as firstname,lname as lastname,matchkey as matchkey,mname as
middlename,recordid as recordid
    FROM customer_name_suspect
    UNION ALL
 SELECT rowid(*) AS id , 'Candidate' AS TYPE, name as name,firstname as

 firstname,lastname as lastname,matchkey as matchkey,middlename as
middlename ,customerid as recordid
```

```
    FROM customer_name_candidate) unionresults
    GROUP BY matchkey) AS innerResult LATERAL VIEW
explode(innerResult.matchgroup) innerresult 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 innerresult.record ["MatchRecordType"],
 innerresult.record ["MatchScore"],
 innerresult.record ["HasDuplicate"],
 innerresult.record ["CollectionNumber"],
 coalesce(innerresult.record ["ExpressMatched"], ''),
 innerresult.record ["SourceType"],
 innerresult.record ["name"],
 innerresult.record ["firstname"],
 innerresult.record ["lastname"],
 innerresult.record ["matchkey"],
 innerresult.record ["middlename"],
 innerresult.record ["recordid"]
FROM (
 SELECT
interMatch(${hivevar:rule},${hivevar:sortfield},${hivevar:expressMatchColumn},

       ${hivevar:collectionNumberZero},${hivevar:interComparison},
       ${hivevar:Candidateheader},${hivevar:Suspectheader},
       unionresults.id,unionresults.name,unionresults.firstname,
       unionresults.lastname, unionresults.matchkey,
      unionresults.middlename,unionresults.recordid ,unionresults.TYPE)
 AS matchgroup
    FROM (
 SELECT rowid(*) AS id, 'Suspect' AS TYPE,fullname as name,fname as
       firstname,lname as lastname,matchkey as matchkey,mname as
middlename,recordid as recordid
    FROM customer_name_suspect
    UNION ALL
 SELECT rowid(*) AS id , 'Candidate' AS TYPE, name as name,
    firstname as firstname,lastname as lastname,matchkey as matchkey,
    middlename as middlename ,customerid as recordid
    FROM customer_name_candidate) unionresults
    GROUP BY matchkey) AS innerResult LATERAL VIEW
explode(innerResult.matchgroup) innerresult 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

Intraflow Match は、単一の入力ストリーム内の類似するデータ レコード間でマッチを検出します。データフローの別のコンポーネントで定義または作成したフィールドに基づいて、階層的なルールを作成できます。

レポート

Intraflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

INPUT_RECORDS マッチング ソート実行前のマッチング ステージにおけるレコードの数。

DUPLICATE_RECORDS	マッチ グループ内の重複レコード (サスペクト レコードまたは候補レコード) の数。
UNIQUE_RECORDS	各マッチ グループで他のレコードにマッチしないサスペクトまたは候補レコードの数。 マッチ グループ内に 1 つしか存在していないレコードであれば、サスペクトは自動的にユニーク レコードとなります。
MATCH_GROUPS	(グループ化) マッチ キーでグループ化されたレコード。
DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクトレコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Express マッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。
AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
TOTAL_DUPLICATES	検出された重複の総数。
TOTAL_SCORE	すべての重複の合計マッチ スコア。

サンプル Hive スクリプト

```
-- 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 'hivevar:rule'
set hivevar: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 'hivevar:header'
set hivevar:header='firstname,lastname,matchkey,middlename,id';

-- Set the express match column (optional)
set hivevar:expresscolumn='matchkey';

-- Set sort field name to the alias used in the query,
 using the configuration property 'hivevar:sortfield'
set hivevar:sortfield='id';

-- Set sort collection number option for unique records using
configuration property 'hivevar:UniqueCollectionNumber'
set hivevar:UniqueCollectionNumber='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( ${hivevar:rule},
    ${hivevar:sortfield},
    ${hivevar:expresscolumn},
    ${hivevar:UniqueCollectionNumber},
    ${hivevar:header},
    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   |
--+---------+------+---------+------+----------+----------+----------+--------+
```

Match Key Generator

Match Key Generator は、レコードごとに非ユニーク キーを作成します。この非ユニーク キーは、潜在的な重複レコードのグループを特定するためにマッチング ステージで使用できます。マッチ キーを使用すると、レコードをマッチ キー別にグループ化し、各グループ内でのみレコードを比較できるので、マッチング プロセスが促進されます。

マッチ キーは、定義したルールを使用して作成され、入力フィールドから構成されます。指定する入力フィールドごとに、そのフィールドで実行されるアルゴリズムが選択されます。その後、各アルゴリズムの結果を連結して、単一のマッチ キー フィールドが作成されます。

マッチ キーの作成に加え、後のデータフローの Intraflow Match ステージまたは Interflow Match ステージで使用する Express マッチ キーも作成できます。

複数のマッチ キーおよび Express マッチ キーを作成できます。

例えば、次のような入力レコードがあり、

名 - Fred
姓 - Mertz

郵便番号 - 21114-1687
性別コード - M

次のようなレコードのデータを組み合わせてマッチ キーを生成するマッチ キー ルールを定義したとします。

入力フィールド	開始位置	長さ
郵便番号	1	5
郵便番号	7	4
姓	1	5
名	1	5
性別コード	1	1

次のようなキーになります。

211141687MertzFredM

サンプル Hive スクリプト

```
-- 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
hivevar:rule='{"matchKeys":[{"expressMatchKey":false,"matchKeyField":"MatchKey1",


"rules":[{"algorithm":"Soundex","field":"busniessname","startPosition":1,"length":0,
```

```
"active":true,"sortInput":null,"removeNoiseCharacters":false}]},
                 {"expressMatchKey":false,"matchKeyField":"MatchKey2",

"rules":[{"algorithm":"Koeln","field":"busniessname","startPosition":1,"length":0,


"active":true,"sortInput":null,"removeNoiseCharacters":false}]}]}';


set hivevar:header='busniessname,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 busniessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
bar.ret["MatchKey2"]
AS MatchKey2 FROM (SELECT *, matchkeygenerator (${hivevar:rule},
${hivevar:header},
 busniessname, 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 busniessname, recordid, bar.ret["MatchKey1"] AS MatchKey1,
 bar.ret["MatchKey2"] AS MatchKey2 FROM (
SELECT *, matchkeygenerator (${hiveconf:rule}, ${hiveconf:header},
 busniessname, recordid) AS ret FROM cust ) bar;

--Sample data in input table customer
--+-----------------------------------------+----------------+--+
--|            cust.busniessname            | 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
+--------------------------------------+----------+-----------+-----------------+--+
|               busniessname           | 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        |
+-------------------------------------+----------+-----------+-----------------+--+
```

Transactional Match

Transactional Match は、重複を特定するため、サスペクト レコードと、あるグループのレコードを照合します。これらのレコードはまず、選択した列によりグループ化され、最初のレコードがサスペクト レコードとしてマークされます。グループの残りすべてのレコードは候補レコードと呼ばれ、サスペクト レコードと照合されます。

候補レコードが重複の場合は、コレクション番号が割り当てられ、そのマッチ レコード タイプに重複が設定され、その候補レコードが書き出されます。グループ内のマッチしない候補にはコレクション番号 0 が割り当てられ、そのラベルにユニークが設定され、その候補が書き出されます。

レポート

Transactional Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。
UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATES_SCORE	すべての重複の合計マッチ スコア。
TOTAL_DUPLICATES	検出された重複の総数。

サンプル Hive スクリプト

```
-- 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 'hivevar:rule'
set hivevar: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 'hivevar:header'
set
hivevar:header='name,firstname,lastname,matchkey,middlename,recordid,id';
```

```
-- Set sort field name to the alias used in the query,
 using the configuration property 'hivevar:sort'
set hivevar:sort='id';

-- Set sort collection number option for unique records using
configuration property
 'hivevar:uniquecolumnsreturn'. The default value is false.
set hivevar:uniquecolumnsreturn='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(${hivevar:rule},
   ${hivevar:sort},
   ${hivevar:uniquecolumnsreturn},
   ${hivevar:header},
   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       |
--+----------+-------+-------+-------+--------+--------+----------+--------+----------+
```

Data Integration モジュールの機能

カスタム Groovy スクリプト Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. Spectrum™ Data & Address Quality for Big Data SDK DIM モジュールの JAR ファイルを登録します。

```
ADD JAR /home/hduser/script/dim.hive.${project.version}.jar;
```

3. `CustomGroovyScript` ジョブの Hive UDF のエイリアスを作成します。

   注： 引用符で囲まれている文字列は、このジョブの実行に必要なクラス名です。

   例を次に示します。

```
CREATE TEMPORARY FUNCTION customscript as
'com.pb.bdq.dim.process.hive.script.groovy.CustomGroovyScriptExecutionUDF';
```

4. Hive フェッチ タスクの変換を有効または無効にします。
   以下に例を示します。

```
set hive.fetch.task.conversion=none;
```

5. `hivevar:defaultConfiguration` を使用して、日付、日時、時刻のパターンを指定します。この設定を該当する変数に割り当てます。

```
set hivevar:defaultConfiguration='{"datePattern":"M/d/yy",
"dateTimePattern":"M/d/yy h:mm a","timePattern":"h:mm a"}';
```

注： これはオプションの設定です。

6. 入力テーブルのヘッダー フィールドをカンマ区切り形式で指定し、変数に割り当てます。例を次に示します。

```
set hivevar:header='busniessname,recordid';
```

7. `hivevar:scriptConfigurations` を使用して、Groovy スクリプトの設定を行います。groovyScriptFile、inputFields、outputFields などの詳細が含まれています。以下に例を示します。

```
set hivevar:scriptConfigurations =
'[{"groovyScriptFile":"/home/hduser/script/groovy_hive.txt",
"inputFields":[{"name":"busniessname","type":"string"},
{"name":"recordid","type":"integer"}],
"outputFields":[{"name":"outtan","type":"double"}]},
{"groovyScriptFile":"/home/hduser/script/groovy2.txt",
"inputFields":[],
"outputFields":[{"name":"outtan2","type":"double"}]}]';
```

8. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

注： このクエリは、Groovy スクリプトによって変換された出力フィールドを返します。

```
SELECT customscript(${hivevar:scriptConfigurations},"",
${hivevar:header}, InputKeyValue, AddressLine1) FROM groovy_tc1;
```

ジョブを実行して出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/script/output'
row format delimited FIELDS TERMINATED BY ',' lines TERMINATED BY '\n'

STORED AS TEXTFILE SELECT * FROM (SELECT customscript
(${hivevar:scriptConfigurations},
${hivevar:defaultConfiguration},${hivevar:header},
InputKeyValue, AddressLine1)
```

```
as mygp FROM groovy_tc1 ) record;
!q;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

サンプル Hive スクリプト

```
-- Register Data Integration Module [DIM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dim.hive.${project.version}.jar;

ADD FILE /home/hduser/script/groovy_hive.txt;
ADD FILE /home/hduser/script/groovy2.txt;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for this job to run.
CREATE TEMPORARY FUNCTION customscript as
'com.pb.bdq.dim.process.hive.script.groovy.CustomGroovyScriptExecutionUDF';

set hive.fetch.task.conversion=none;

-- Set default configuration
set hivevar:defaultConfiguration='{"datePattern":"M/d/yy",
"dateTimePattern":"M/d/yy h:mm a","timePattern":"h:mm a"}';

-- Set the header configuration
set hivevar:header='busniessname,recordid';

set hivevar:scriptConfigurations =
'[{"groovyScriptFile":"/home/hduser/script/groovy_hive.txt",
"inputFields":[{"name":"busniessname","type":"string"},
{"name":"recordid","type":"integer"}],
"outputFields":[{"name":"outtan","type":"double"}]},
{"groovyScriptFile":"/home/hduser/script/groovy2.txt",
"inputFields":[],
"outputFields":[{"name":"outtan2","type":"double"}]}]';

SELECT customscript(${hivevar:scriptConfigurations},"",${hivevar:header},
InputKeyValue, AddressLine1) FROM groovy_tc1;

INSERT OVERWRITE LOCAL DIRECTORY '/home/hduser/script/output'
row format delimited FIELDS TERMINATED BY ',' lines TERMINATED BY '\n'

STORED AS TEXTFILE SELECT * FROM (SELECT
customscript(${hivevar:scriptConfigurations},
${hivevar:defaultConfiguration},${hivevar:header}, InputKeyValue,
AddressLine1)
```

```
as mygp FROM groovy_tc1 ) record;
!q;
```

Data Normalization モジュールの関数

Data Normalization モジュールの Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. Spectrum™ Data & Address Quality for Big Data SDK DNM モジュールの JAR ファイルを登録します。

   ```
   ADD JAR <Directory path>/dnm.hive.${project.version}.jar;
   ```

3. 実行するデータ品質ジョブの Hive UDF のエイリアスを作成します。

   注： 引用符で囲まれている文字列は、このジョブの実行に必要なクラス名です。

   例:

   ```
   CREATE TEMPORARY FUNCTION tablelookup as
   'com.pb.bdq.dnm.process.hive.tablelookup.TableLookUpUDF';
   ```

4. リファレンス データのパスを指定します。
   - **リファレンス データが HDFS 上にある場合**
     - リファレンス データがジョブ用の**作業ディレクトリにダウンロードされる**場合
       - リファレンス データがアーカイブされていないファイル形式の場合は、リファレンス ディレクトリを次のように設定します。

         ```
         set
         hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
         "dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';
         ```

- リファレンス データがアーカイブ済みの形式の場合は、リファレンス ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":
{"dataDownloader":"DC"}}';
```

- リファレンス データがジョブ用の**ローカル ノードにダウンロードされる**場合この場合は、リファレンス データ ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/local/download"}}';
```

- **リファレンス データがローカル パスにある場合**: クラスタの各ノードでデータが同じパスに配置されるようにします。

  リファレンス ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';
```

5. ジョブの環境設定と詳細情報を指定して、それぞれの変数または設定プロパティに代入します。

   注： ルールは JSON 形式である必要があります。

   例を次に示します。

```
set hivevar:rule='{"rules":[{"action":"Standardize",
"source":"CityCode", "tableName":"State Name Abbreviations",
"lookupMultipleWordTerms":false,
"lookupIndividualTermsWithinField":false,
"destination":"CityCode"}]}';
```

   注: それぞれのジョブ環境設定の設定プロパティを、必ず使用してください。例えば、それぞれのサンプル HQL ファイルに記載されている `pb.bdq.match.rule`、`pb.bdq.match.express.column`、`pb.bdq.consolidation.sort.field` などです。

6. 入力テーブルのヘッダーフィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

```
set hivevar:header
='AccountDescription,Address,ApartmentNumber,CityCode';
```

7. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

```
SELECT bar.ret["StandardizationTermIdentified"],
 bar.ret["accountdescription"],
 bar.ret["address"],
 bar.ret["apartmentnumber"],
 bar.ret["citycode"]
FROM (
 SELECT tablelookup(${hivevar:rule}, ${hiveconf:refdir},
${hiveconf:header}, accountdescription, address, apartmentnumber,
citycode)
 AS ret
 FROM citizen_data
 ) bar;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
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;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

Advanced Transformer

Advanced Transformer ジョブは、テーブルまたは正規表現を使用して、一連のデータ列をスキャンして複数のフィールドに分割します。このコンポーネントは、特定の語、あるいは語の右側または左側から指定した数だけ単語を抽出します。抽出データと非抽出データを既存のフィールドまたは新しいフィールドに配置できます。

例えば、次の住所フィールドから一組の情報を抽出し、その情報を別のフィールドに配置するとします。

2300 BIRCH RD STE 100

これを行うには、語 STE と語 STE の右側にあるすべての単語を抽出する Advanced Transformer を作成することができます、分離されるフィールド：

2300 BIRCH RD

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- 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.advancedtransformer.AdvancedTransformerUDF';

-- Set rule
set hivevar: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
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
```

```
"dataDir":"/home/data/referenceData"}';

-- set header
set hivevar:header ='AccountDescription,Address';

set hive.fetch.task.conversion=none;

-- 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(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar: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(${hivevar:rule}, ${hivevar:refereceDataDetails},

${hivevar: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          |
+-----------------------------+-----------------+---------------------+-----------------+
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- 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.advancedtransformer.AdvancedTransformerUDF';


-- Set rule
set hivevar: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 data details for Download manager, paas dataDir where
 data resides in HDFS
 and localFS path to download the data and dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData",
"dataDownloader":{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar:header ='AccountDescription,Address';

set hive.fetch.task.conversion=none;
```

```
-- 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(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar: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(${hivevar:rule}, ${hivevar:refereceDataDetails},

 ${hivevar: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          |
+-------------------------------+-------------------+-------------------+-------------------+
```

Open Parser

Open Parser は、世界のさまざまなカルチャーからの入力データを、シンプルかつ強力なパーシング グラマーでパースします。このグラマーを使用すると、ドメイン パターンを表す一連の式を入力データのパース用に定義できます。また、Open Parser は、統計データを収集してパーシング マッチにスコアを付けるため、パーシング グラマーの効果を調べるのも容易です。

Open Name Parser を使用して、次のことができます。

- ドメインエディタで定義したドメイン固有およびカルチャー固有のパーシング グラマーを使用して、入力データをパースします。
- ドメインエディタで使用できるものと同じ、シンプルかつ強力なパーシング グラマーを使用して Open Parser で定義した、ドメインに依存しないパーシング グラマーを使用して、入力データをパースします。
- データフロー オプションで定義したドメインに依存しないパーシング グラマーを実行時に使用して、入力データをパースします。
- ターゲット入力データファイルを使用してジョブを実行する前に、パーシング グラマーをプレビューして、サンプル入力データがどのようにパースされるかをテストします。
- パーシング グラマー結果をトレースして、定義した式にトークンがどれくらいマッチしたか、またはマッチしなかったかを確認し、マッチング処理に関する理解を深めます。

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Register the grammar for open parser
ADD FILE
/home/hadoop/testing/openparser/hive/testdata/Open_parser_grammer_tc1.txt;

-- Provide alias to UDF class (optional). String in quotes
represent class names needed for this job to run.
-- Open 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 openparser as
'com.pb.bdq.dnm.process.hive.openparser.OpenParserUDF';
```

```
set hive.fetch.task.conversion=minimal;

set hivevar:header='InputKeyValue,addressline';

-- 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 hivevar:refereceDataDetails='{"referenceDataPathLocation":
"LocaltoDataNodes","dataDir":"/home/data/referenceData"}';

-- Set rule with grammar file name
set hivevar:rule='{"grammar":null,"domain":null,"defaultCulture":null,
"culture":null,"returnMultipleParsedRecords":false,"debug":false,
"grammerFilePath":"Open_parser_grammer_tc1.txt"}';

--Execute Query on desired table to sump the output to local dir.
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/testing/openparser/hive/testdata/output'
row format delimited FIELDS TERMINATED BY '|'
lines terminated by '\n'
STORED AS TEXTFILE
SELECT coalesce(tmp2.record ["InputKeyValue"], '')
 ,coalesce(tmp2.record ["addressline"], '')
 ,coalesce(tmp2.record ["ApartmentNumber"], '')
 ,coalesce(tmp2.record ["HouseNumber"], '')
 ,coalesce(tmp2.record ["IsParsed"], '')
 ,coalesce(tmp2.record ["LeadingDirectional"], '')
 ,coalesce(tmp2.record ["ParserScore"], '')
 ,coalesce(tmp2.record ["POBox"], '')
 ,coalesce(tmp2.record ["PrivateMailbox"], '')
 ,coalesce(tmp2.record ["RRHC"], '')
 ,coalesce(tmp2.record ["RuleID"], '')
 ,coalesce(tmp2.record ["StreetName"], '')
 ,coalesce(tmp2.record ["StreetSuffix"], '')
 ,coalesce(tmp2.record ["TrailingDirectional"], '')
FROM (
 SELECT openparser(${hivevar:rule},${hivevar:refereceDataDetails},
       ${hivevar:header}, InputKeyValue, addressline) AS mygp
 FROM openparserinputtable1
 ) AS addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 AS
record;
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- Register Data Normalisation Module [DNM] BDQ Hive UDF Jar
ADD JAR <Directory path>/dnm.hive.${project.version}.jar;

-- Register the grammar for open parser
ADD FILE
/home/hadoop/testing/openparser/hive/testdata/Open_parser_grammer_tc1.txt;
```

```
-- Provide alias to UDF class (optional).
String in quotes represent class names needed for this job to run.
-- Open 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 openparser as
'com.pb.bdq.dnm.process.hive.openparser.OpenParserUDF';

set hive.fetch.task.conversion=minimal;

set hivevar:header='InputKeyValue,addressline';

-- Set reference data details for Download manager, paas
dataDir where data resides in HDFS and localFS path to download the data
 and dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"HDFS","dataDir":"/home/data/dm/referenceData","dataDownloader":
{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- Set rule with grammar file name
set hivevar:rule='{"grammar":null,"domain":null,"defaultCulture":null,
"culture":null,"returnMultipleParsedRecords":false,"debug":false,
"grammerFilePath":"Open_parser_grammer_tc1.txt"}';

--Execute Query on desired table to sump the output to local dir.
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hadoop/testing/openparser/hive/testdata/output'
row format delimited FIELDS TERMINATED BY '|'
lines terminated by '\n'
STORED AS TEXTFILE
SELECT coalesce(tmp2.record ["InputKeyValue"], '')
 ,coalesce(tmp2.record ["addressline"], '')
 ,coalesce(tmp2.record ["ApartmentNumber"], '')
 ,coalesce(tmp2.record ["HouseNumber"], '')
 ,coalesce(tmp2.record ["IsParsed"], '')
 ,coalesce(tmp2.record ["LeadingDirectional"], '')
 ,coalesce(tmp2.record ["ParserScore"], '')
 ,coalesce(tmp2.record ["POBox"], '')
 ,coalesce(tmp2.record ["PrivateMailbox"], '')
 ,coalesce(tmp2.record ["RRHC"], '')
 ,coalesce(tmp2.record ["RuleID"], '')
 ,coalesce(tmp2.record ["StreetName"], '')
 ,coalesce(tmp2.record ["StreetSuffix"], '')
 ,coalesce(tmp2.record ["TrailingDirectional"], '')
FROM (
 SELECT openparser(${hivevar:rule},${hivevar:refereceDataDetails},
       ${hivevar:header}, InputKeyValue, addressline) AS mygp
 FROM openparserinputtable1
 ) AS addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 AS
record;
```

Table Lookup

Table Lookup ステージは、語とその語の妥当性確認済みの形式とを照合して正規化し、標準バージョンを適用します。この評価は、正規化する語をテーブルから検索して実行されます。

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- 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 hivevar: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
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar: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(${hivevar:rule}, ${hivevar:refereceDataDetails},
${hivevar: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:refereceDataDetails},

      ${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 |
--+-----------------------+-----------------+----------------+----------------+-------+
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- 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 hivevar:rule='{"rules":[{"action":"Standardize", "source":"CityCode",

 "tableName":"State Name Abbreviations", "lookupMultipleWordTerms":false,

"lookupIndividualTermsWithinField":false, "destination":"CityCode"}]}';

-- Set reference data details for Download manager, paas dataDir where

data resides in HDFS and localFS path to download the data and
dataDownloader as HDFS
set hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":"HDFS",
"localFSRepository":"/local/download"}}';

set hive.fetch.task.conversion=none;

-- set header
set hivevar: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(${hivevar:rule}, ${hivevar:refereceDataDetails},
 ${hivevar: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:refereceDataDetails},
 ${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 |
--+--------------------+--------------------+---------------+----------------+-------+
```

Global Addressing モジュールの関数

Global Addressing モジュールの Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. Spectrum™ Data & Address Quality for Big Data SDK GAM モジュールの JAR ファイルを登録します。

   ```
   ADD JAR
   /home/hduser/gam/gam.globaladdressvalidation.hive.${project.version}.jar
   ```

3. 実行する住所品質ジョブの Hive UDF のエイリアスを作成します。

   注： 引用符で囲まれている文字列は、このジョブの実行に必要なクラス名です。

   例:

   ```
   CREATE TEMPORARY FUNCTION addressvalidation as
   'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';
   ```

4. Hive フェッチ タスクの変換を有効または無効にします。
   例:

   ```
   set hive.fetch.task.conversion=none;
   ```

5. データベース パス、国コード、プロセス タイプなど、エンジン設定を設定するには、`hivevar:engineconf` を使用します。
   例:

```
set hivevar:engineconf='{"productDatabaseInfoList":
[{"dbPath":"/home/hduser/ReferenceData/AddressQuality/GAM/GGB062017",
"countryCode":"GBR","processType":"VALIDATE"}]}';
```

6. `hivevar:inputoption` パラメータを使用して、入力データの設定を指定します。
   例:

```
set hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed",
"defaultCountry":"GBR","maximumResults":2,"returnInputAddress":false,
"returnParsedAddress":false,"returnPrecisionCode":false,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,"mustMatchPostCode":false,
"keepMultiMatch":true,"preferPostalOverCity":false,"cityFallback":true,
"postalFallback":true,"validationLevel":"ADDRESS"}';
```

7. 入力テーブルのヘッダー フィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

```
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,Country';
```

8. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

   注： このクエリは、各行の入力フィールドを含むキー/値ペアのマップを返します。

```
SELECT
tmp2.record["HouseNumber"],tmp2.record["Confidence"],tmp2.record["AddressLine1"],
tmp2.record["StreetName"],tmp2.record["PostalCode"] FROM (SELECT
```

```
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,stateprovince,firmname,
country)as mygp from address_validation) as addressgroup LATERAL VIEW
 explode(addressgroup.mygp)tmp2 as record;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
INSERT OVERWRITE LOCAL DIRECTORY
'/home/hduser/GlobalAddressValidation/'row format delimited
FIELDS TERMINATED BY ',' lines terminated by '\n' STORED AS TEXTFILE
 SELECT * FROM (SELECT addressvalidation(${hivevar:engineconf},
${hivevar:inputoption},${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,
stateprovince,firmname,country)as mygp from address_validation) as
addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

Global Address Validation

Global Addressing Validation は、国際住所に対する高度な住所正規化と検証の機能を提供します。Global Address Validation は Global Addressing モジュールの一部です。

Global Address Validation は、以下の優先国に対して高度な住所の正規化および検証の機能を提供します。各国に対して 3 桁の ISO 国コードが記載されています。ISO 国コードの全一覧については、**ISO 国コードとコーダー サポート**（417ページ）を参照してください。

- アルゼンチン (ARG)
- オーストラリア (AUS)
- オーストリア (AUT)
- ベルギー (BEL)
- ブラジル (BRA)
- カナダ (CAN)
- 中国 (CHN)
- チェコ共和国 (CHZ)
- デンマーク (DNK)
- フィンランド (FIN)
- フランス (FRA)
- ドイツ (DEU)
- ギリシャ (GRC)
- インド (IND)

- アイルランド (IRL)
- イタリア (ITA)
- 日本 (JPN)
- マレーシア (MYS)
- メキシコ (MEX)
- オランダ (NLD)
- ニュージーランド (NZL)
- ノルウェー (NOR)
- ポーランド (POL)
- ロシア (RUS)
- スペイン (ESP)
- スウェーデン (SWE)
- スイス (CHE)
- 英国 (GBR) (POI 情報を含みます)
- 米国 (USA)

サンプル Hive スクリプト - Addressing Validation

ローカル ノードに配置されたリファレンス データ

```
-- Register Global Addressing Module [Global Address Validation] BDQ
Hive UDF Jar
ADD JAR <directory
path>/gam-globaladdressvalidation-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 addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":
[{"dbPath":"<path to extracted spd>",
"countryCode":["FRA"],"processType":"VALIDATE"}],
"referenceDataPathLocation":"LocaltoDataNodes"}';

-- set input configuration
set hivevar:inputoption='{"casing":"Upper","matchMode":"Relaxed",
"defaultCountry":"FRA","maximumResults":1,"returnInputAddress":true,
"returnParsedAddress":true,"returnPrecisionCode":true,
"returnCountrySpecificFields":true,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"keepMultiMatch":false,"preferPostalOverCity":false,
"cityFallback":false,"postalFallback":false,"validationLevel":"ADDRESS"}';
```

```
-- set header
set hivevar:header='RecordID,AddressLine1,City,PostalCode,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '<local directory path>' row format
delimited FIELDS TERMINATED BY '|' lines terminated by '\n' STORED AS
TEXTFILE
SELECT
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
```

```
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["RecordID"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["VendorCode"] ,'')
 FROM (SELECT  addressvalidation
(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},RecordID,AddressLine1,City,PostalCode,Country)
as mygp from gavtable )as addressgroup
 LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- Register Global Addressing Module [Global Address Validation] BDQ
Hive UDF Jar
ADD JAR <directory
path>/gam-globaladdressvalidation-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 addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":[{"referenceDataPath":
{"referenceDataPathLocation":"HDFS",
"dataDir":"/user/hadoop/RefrenceData/AddressValidation",
"dataDownloader":{"dataDownloader":"HDFS","localFSRepository":
"/opt/PitneyBowes/ReferenceData/AddressValidation"}},
"countryCode":["GBR"],"processType":"VALIDATE"}]}';

-- set input configuration
set hivevar:inputoption='{"casing":"Upper","matchMode":
"Relaxed","defaultCountry":"GBR","maximumResults":1,"returnInputAddress"
:true,"returnParsedAddress":true,"returnPrecisionCode":true,"returnCountrySpecificFields":true,
"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,
"mustMatchLocality":false,"mustMatchState":false,
"mustMatchStateProvince":false,"mustMatchPostCode":false,"keepMultiMatch":false,
"preferPostalOverCity":false,"cityFallback":false,
"postalFallback":false,"validationLevel":"ADDRESS"}';

-- set header
set hivevar:header='RecordID,AddressLine1,City,PostalCode,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY '<local directory path>' row format
delimited FIELDS TERMINATED BY '|' lines terminated by '\n' STORED AS
TEXTFILE
SELECT
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
```

```
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["POBox"] ,''),
```

```
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["RecordID"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["VendorCode"] ,'') FROM
(SELECT  addressvalidation(${hivevar:engineconf},
${hivevar:inputoption},${hivevar:header},
RecordID,AddressLine1,City,PostalCode,Country) as mygp from gavtable )
 as
 addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

HDFS に配置され、ジョブ用の作業ディレクトリにダウンロードされたリファレンス データ

```
-- Register Global Addressing Module [GAV] BDQ Hive UDF Jar
ADD JAR <directory
path>/gam-globaladdressvalidation-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 addressvalidation as
 'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set
hivevar:engineconf='{"dataDirPath":{"dataDir":"/home/hduser/gav/GGB062017.zip",
"referenceDataPathLocation":"HDFS"}, "productDatabaseInfoList":
[{"dbPath":"/home/hduser/gav/GGB062017.zip","countryCode":
["GBR"],"processType":"VALIDATE"}]}';

-- set input configuration
set hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed",
"defaultCountry":"GBR","maximumResults":2,"returnInputAddress":false,
"returnParsedAddress":false,"returnPrecisionCode":false,"mustMatchAddressNumber":false,
```

```
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,
"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"keepMultiMatch":true,"preferPostalOverCity":false,
"cityFallback":true,"postalFallback":true,"validationLevel":"ADDRESS"}';


-- set header
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,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 * FROM (SELECT
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},inputkeyvalue,addressline1,AddressLine2,city,postalcode,
stateprovince,firmname,country)as mygp from address_validation) as
addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

サンプル Hive スクリプト - 米国 Addressing Validation

ローカル ノードに配置されたリファレンス データ

```
-- Register Universal Addressing Module [UAM-GAV(Global Address
Validation)] BDQ Hive UDAF Jar
ADD JAR <directory path>/gam.globaladdressvalidation.hive.${version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for
this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":[{"referenceDataPath":
{"referenceDataPathLocation":"LocaltoDataNodes","dataDir":"/user/hadoop/ReferenceData/
GAV_US_DOM"},"countryCode":["USA"],"processType":"VALIDATE"}]}';

-- set input configuration
set
hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed","defaultCountry":"USA",
"maximumResults":2,"returnInputAddress":false,"returnParsedAddress":false,
"returnPrecisionCode":false,"returnCountrySpecificFields":true,"mustMatchAddressNumber":false,
"mustMatchStreet":false,"mustMatchCity":false,"mustMatchLocality":false,"mustMatchState":false,
"mustMatchStateProvince":false,"mustMatchPostCode":false,"preferPostalOverCity":false,
"cityFallback":true,"postalFallback":true,"validationLevel":"ADDRESS",
"preferPOBoxOverStreet":false,"allStreetMatching":true,"assignABBRCity":false,
"assignLot":true,"cassFlag":true,"convertSecToPMB":false,"dpv":true,"dpvCMRA":true,
```

```
"dpvDNA":true,"dpvIndicator":true,"dpvNDD":true,"dpvNSL":true,"dpvNoStat":true,
"dpvPBSA":true,"dpvThrowback":true,"dpvTieBreak":true,"dpvVacant":true,
"dualAddress":"POBoxPreference","earlyWarningSystem":true,"lacsLinkIndicator":true,
"logLevel":"Error","mailerAddress":"","mailerAddress2":"","mailerAddress3":"",
"mailerAddress4":"","mailerCityLine":"","mailerName":"","processingMode":"",
"r777Deliverable":false,"removeNoiseChars":true,"residentialDeliveryIndicator":true,
"returnAliasStreet":"AbbrPreferredAliasBase","returnInputFirm":false,
"returnSLKSecondary":"BothSuitelinkAndInput","standalonePMB":false,"standaloneUnit":false,
"suiteLinkIndicator":false,"vmDataBlock":false,"dataBlocks":false,"return3553Data":false,
"returnAdsInfo":false,"returnAlternate":false,"returnRunStatistics":false,
"returnSetupInfo":false}';

-- set header
set hivevar:header='AddressLine1,City,PostalCode,StateProvince,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY
'/opt/PitneyBowes/addressvalidation_us/out' row format
delimited FIELDS TERMINATED BY '|' lines terminated by '\n' STORED AS
TEXTFILE
SELECT
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["VendorCode"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
```

```
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
```

```
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["DualAddressParsed.Input"] ,''),
coalesce(tmp2.record["IND.ExtendedResultCode"] ,''),
coalesce(tmp2.record["IND.IsRural"] ,''),
coalesce(tmp2.record["IND.SubLocality"] ,''),
coalesce(tmp2.record["IND.POICategory"] ,''),
coalesce(tmp2.record["JPN.BANCHI"] ,''),
coalesce(tmp2.record["JPN.CHOMOKO"] ,''),
coalesce(tmp2.record["JPN.CHOOAZA"] ,''),
coalesce(tmp2.record["JPN.GO"] ,''),
coalesce(tmp2.record["JPN.JUSHOCODE"] ,''),
coalesce(tmp2.record["USA.DPV"] ,''),
coalesce(tmp2.record["USA.DPV.Footnote"] ,''),
coalesce(tmp2.record["USA.DPV.CMRA"] ,''),
coalesce(tmp2.record["USA.DPV.FalsePositive"] ,''),
coalesce(tmp2.record["USA.DPV.Flags"] ,''),
coalesce(tmp2.record["USA.DPV.NoStat"] ,''),
coalesce(tmp2.record["USA.POBoxOnly"] ,''),
coalesce(tmp2.record["USA.DPV.Vacant"] ,''),
coalesce(tmp2.record["USA.DPV.PBSAFound"] ,''),
coalesce(tmp2.record["USA.LACS"] ,''),
coalesce(tmp2.record["USA.LACS.SeedHit"] ,''),
coalesce(tmp2.record["USA.LACS.ReturnCode"] ,''),
coalesce(tmp2.record["USA.LACS.Indicator"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSAddress"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSRange"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSLeadingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSStreetName"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSSuffix"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSTrailingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitD"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitN"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP4"] ,''),
coalesce(tmp2.record["USA.LOTCode"] ,''),
coalesce(tmp2.record["USA.LOTSequence"] ,''),
coalesce(tmp2.record["USA.SuiteLink.ReturnCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.MatchCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.Fidelity"] ,''),
coalesce(tmp2.record["VeriMoveDataBlock"] ,''),
coalesce(tmp2.record["USA.CarrierRouteCode"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Input"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type.Input"] ,''),
coalesce(tmp2.record["USA.RDI"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine1"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine2"] ,''),
```

```
coalesce(tmp2.record["USA.CASSCityName"] ,''),
coalesce(tmp2.record["USA.FullCityName"] ,''),
coalesce(tmp2.record["USA.AbbreviatedCityName"] ,''),
coalesce(tmp2.record["USA.NonMailingCityName"] ,''),
coalesce(tmp2.record["USA.PreferredCityName"] ,''),
coalesce(tmp2.record["USA.PreferredState"] ,''),
coalesce(tmp2.record["USA.EWSFailure"] ,''),
coalesce(tmp2.record["USA.MatchLevel"] ,''),
coalesce(tmp2.record["USA.DefaultMatch"] ,''),
coalesce(tmp2.record["USA.Status"] ,''),
coalesce(tmp2.record["USA.Status.Code"] ,''),
coalesce(tmp2.record["USA.Status.Description"] ,''),
coalesce(tmp2.record["USA.PostalBarcode"] ,''),
coalesce(tmp2.record["USA.BCCheckDigit"] ,''),
coalesce(tmp2.record["USA.AdvancedBarcode"] ,''),
coalesce(tmp2.record["USA.FiveDigitBarcode"] ,''),
coalesce(tmp2.record["USA.ZIPValid"] ,''),
coalesce(tmp2.record["USA.ZIP4Valid"] ,''),
coalesce(tmp2.record["USA.AddressLocation"] ,''),
coalesce(tmp2.record["USA.CongressionalDistrict"] ,''),
coalesce(tmp2.record["USA.FIPSCountyNumber"] ,''),
coalesce(tmp2.record["USA.AltStreetType"] ,''),
coalesce(tmp2.record["USA.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.Range"] ,''),
coalesce(tmp2.record["USA.Parsed.PreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.PostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetSuffix"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitNumber"] ,''),
coalesce(tmp2.record["USA.Parsed.Unit2Designator"] ,''),
coalesce(tmp2.record["USA.Parsed.Unit2Number"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitNumber"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.AltRange"] ,''),
coalesce(tmp2.record["USA.Parsed.AltPreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltPostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltStreetSuffix"], '') FROM
(SELECT  addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},AddressLine1,City,PostalCode,StateProvince,Country)
as mygp from usaddress )
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- Register Universal Addressing Module [UAM-GAV(Global Address
Validation)] BDQ Hive UDAF Jar
ADD JAR <directory path>/gam.globaladdressvalidation.hive.${version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
```

```
class names needed for
this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":[{"referenceDataPath":
{"referenceDataPathLocation":"HDFS","dataDir":"/user/hadoop/RefrenceData/AddressValidation",
"dataDownloader":{"dataDownloader":"HDFS","localFSRepository":
"/opt/PitneyBowes/ReferenceData/AddressValidation"}},"countryCode":["USA"],
"processType":"VALIDATE"}]}';

-- set input configuration
set
hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed","defaultCountry":"USA",
"maximumResults":2,"returnInputAddress":false,"returnParsedAddress":false,
"returnPrecisionCode":false,"returnCountrySpecificFields":true,
"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,
"mustMatchLocality":false,"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"preferPostalOverCity":false,"cityFallback":true,
"postalFallback":true,"validationLevel":"ADDRESS","preferPOBoxOverStreet":false,
"allStreetMatching":true,"assignABBRCity":false,"assignLot":true,"cassFlag":true,
"convertSecToPMB":false,"dpv":true,"dpvCMRA":true,"dpvDNA":true,"dpvIndicator":true,
"dpvNDD":true,"dpvNSL":true,"dpvNoStat":true,"dpvPBSA":true,"dpvThrowback":true,
"dpvTieBreak":true,"dpvVacant":true,"dualAddress":"POBoxPreference",
"earlyWarningSystem":true,"lacsLinkIndicator":true,"logLevel":"Error",
"mailerAddress":"","mailerAddress2":"","mailerAddress3":"","mailerAddress4":"",
"mailerCityLine":"","mailerName":"","processingMode":"","r777Deliverable":false,
"removeNoiseChars":true,"residentialDeliveryIndicator":true,
"returnAliasStreet":"AbbrPreferredAliasBase","returnInputFirm":false,
"returnSLKSecondary":"BothSuitelinkAndInput","standalonePMB":false,"standaloneUnit":false,
"suiteLinkIndicator":false,"vmDataBlock":false,"dataBlocks":false,"return3553Data":false,
"returnAdsInfo":false,"returnAlternate":false,"returnRunStatistics":false,
"returnSetupInfo":false}';

-- set header
set hivevar:header='AddressLine1,City,PostalCode,StateProvince,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY
'/opt/PitneyBowes/addressvalidation_us/out'
row format delimited FIELDS TERMINATED BY '|' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["Country"] ,''),
```

```
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["VendorCode"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
```

```
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["DualAddressParsed.Input"] ,''),
coalesce(tmp2.record["IND.ExtendedResultCode"] ,''),
coalesce(tmp2.record["IND.IsRural"] ,''),
coalesce(tmp2.record["IND.SubLocality"] ,''),
coalesce(tmp2.record["IND.POICategory"] ,''),
coalesce(tmp2.record["JPN.BANCHI"] ,''),
coalesce(tmp2.record["JPN.CHOMOKO"] ,''),
coalesce(tmp2.record["JPN.CHOOAZA"] ,''),
coalesce(tmp2.record["JPN.GO"] ,''),
coalesce(tmp2.record["JPN.JUSHOCODE"] ,''),
coalesce(tmp2.record["USA.DPV"] ,''),
coalesce(tmp2.record["USA.DPV.Footnote"] ,''),
coalesce(tmp2.record["USA.DPV.CMRA"] ,''),
coalesce(tmp2.record["USA.DPV.FalsePositive"] ,''),
coalesce(tmp2.record["USA.DPV.Flags"] ,''),
coalesce(tmp2.record["USA.DPV.NoStat"] ,''),
coalesce(tmp2.record["USA.POBoxOnly"] ,''),
coalesce(tmp2.record["USA.DPV.Vacant"] ,''),
coalesce(tmp2.record["USA.DPV.PBSAFound"] ,''),
coalesce(tmp2.record["USA.LACS"] ,''),
coalesce(tmp2.record["USA.LACS.SeedHit"] ,''),
coalesce(tmp2.record["USA.LACS.ReturnCode"] ,''),
coalesce(tmp2.record["USA.LACS.Indicator"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSAddress"] ,''),
```

```
coalesce(tmp2.record["USA.LACS.PreLACSRange"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSLeadingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSStreetName"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSSuffix"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSTrailingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitD"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitN"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP4"] ,''),
coalesce(tmp2.record["USA.LOTCode"] ,''),
coalesce(tmp2.record["USA.LOTSequence"] ,''),
coalesce(tmp2.record["USA.SuiteLink.ReturnCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.MatchCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.Fidelity"] ,''),
coalesce(tmp2.record["VeriMoveDataBlock"] ,''),
coalesce(tmp2.record["USA.CarrierRouteCode"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Input"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type.Input"] ,''),
coalesce(tmp2.record["USA.RDI"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine1"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine2"] ,''),
coalesce(tmp2.record["USA.CASSCityName"] ,''),
coalesce(tmp2.record["USA.FullCityName"] ,''),
coalesce(tmp2.record["USA.AbbreviatedCityName"] ,''),
coalesce(tmp2.record["USA.NonMailingCityName"] ,''),
coalesce(tmp2.record["USA.PreferredCityName"] ,''),
coalesce(tmp2.record["USA.PreferredState"] ,''),
coalesce(tmp2.record["USA.EWSFailure"] ,''),
coalesce(tmp2.record["USA.MatchLevel"] ,''),
coalesce(tmp2.record["USA.DefaultMatch"] ,''),
coalesce(tmp2.record["USA.Status"] ,''),
coalesce(tmp2.record["USA.Status.Code"] ,''),
coalesce(tmp2.record["USA.Status.Description"] ,''),
coalesce(tmp2.record["USA.PostalBarcode"] ,''),
coalesce(tmp2.record["USA.BCCheckDigit"] ,''),
coalesce(tmp2.record["USA.AdvancedBarcode"] ,''),
coalesce(tmp2.record["USA.FiveDigitBarcode"] ,''),
coalesce(tmp2.record["USA.ZIPValid"] ,''),
coalesce(tmp2.record["USA.ZIP4Valid"] ,''),
coalesce(tmp2.record["USA.AddressLocation"] ,''),
coalesce(tmp2.record["USA.CongressionalDistrict"] ,''),
coalesce(tmp2.record["USA.FIPSCountyNumber"] ,''),
coalesce(tmp2.record["USA.AltStreetType"] ,''),
coalesce(tmp2.record["USA.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.Range"] ,''),
coalesce(tmp2.record["USA.Parsed.PreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.PostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetSuffix"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitNumber"] ,''),
```

```
coalesce(tmp2.record["USA.Parsed.Unit2Designator"] ,''),
coalesce(tmp2.record["USA.Parsed.Unit2Number"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitNumber"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.AltRange"] ,''),
coalesce(tmp2.record["USA.Parsed.AltPreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltPostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltStreetSuffix"], '')
FROM (SELECT
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},
${hivevar:header},AddressLine1,City,PostalCode,StateProvince,Country)
as mygp from usaddress )
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

HDFS に配置され、ジョブ用の作業ディレクトリにダウンロードされたリファレンス データ

```
-- Register Universal Addressing Module [UAM-GAV(Global Address
Validation)] BDQ Hive UDAF
Jar
ADD JAR <directory path>/gam.globaladdressvalidation.hive.${version}.jar;

-- Provide alias to UDF class (optional). String in quotes represent
class names needed for
this job to run.
CREATE TEMPORARY FUNCTION addressvalidation as
 'com.pb.bdq.gam.process.hive.addressvalidation.AddressValidationUDF';

ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/GAM/GAV_US_DOM.spd;

-- set Engine configuration
set hivevar:engineconf='{"productDatabaseInfoList":[{"referenceDataPath":
{"referenceDataPathLocation":"HDFS","dataDir":"/user/hadoop/RefrenceData/AddressValidation",
"dataDownloader":{"dataDownloader":"DC"}},"countryCode":["USA"],"processType":"VALIDATE"}]}';

-- set input configuration
set
hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed","defaultCountry":"USA",
"maximumResults":2,"returnInputAddress":false,"returnParsedAddress":false,
"returnPrecisionCode":false,"returnCountrySpecificFields":true,
"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,
"mustMatchLocality":false,"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"preferPostalOverCity":false,"cityFallback":true,
"postalFallback":true,"validationLevel":"ADDRESS","preferPOBoxOverStreet":false,
"allStreetMatching":true,"assignABBRCity":false,"assignLot":true,"cassFlag":true,
"convertSecToPMB":false,"dpv":true,"dpvCMRA":true,"dpvDNA":true,"dpvIndicator":true,
"dpvNDD":true,"dpvNSL":true,"dpvNoStat":true,"dpvPBSA":true,"dpvThrowback":true,
"dpvTieBreak":true,"dpvVacant":true,"dualAddress":"POBoxPreference",
"earlyWarningSystem":true,"lacsLinkIndicator":true,"logLevel":"Error",
```

```
"mailerAddress":"","mailerAddress2":"","mailerAddress3":"","mailerAddress4":"",
"mailerCityLine":"","mailerName":"","processingMode":"","r777Deliverable":false,
"removeNoiseChars":true,"residentialDeliveryIndicator":true,
"returnAliasStreet":"AbbrPreferredAliasBase","returnInputFirm":false,
"returnSLKSecondary":"BothSuitelinkAndInput","standalonePMB":false,
"standaloneUnit":false,"suiteLinkIndicator":false,"vmDataBlock":false,
"dataBlocks":false,"return3553Data":false,"returnAdsInfo":false,
"returnAlternate":false,"returnRunStatistics":false,"returnSetupInfo":false}';

-- set header
set hivevar:header='AddressLine1,City,PostalCode,StateProvince,Country';

-- Query to dump output data to a file
INSERT OVERWRITE LOCAL DIRECTORY
'/opt/PitneyBowes/addressvalidation_us/out'
row format delimited FIELDS TERMINATED BY '|' lines terminated by '\n'
 STORED AS TEXTFILE
SELECT
coalesce(tmp2.record["AddressLine1"] ,''),
coalesce(tmp2.record["AddressBlock1"] ,''),
coalesce(tmp2.record["AddressBlock2"] ,''),
coalesce(tmp2.record["City"] ,''),
coalesce(tmp2.record["CitySubdivision"] ,''),
coalesce(tmp2.record["StateProvince"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision"] ,''),
coalesce(tmp2.record["PostalCode"] ,''),
coalesce(tmp2.record["Country"] ,''),
coalesce(tmp2.record["PrecisionCode"] ,''),
coalesce(tmp2.record["VendorCode"] ,''),
coalesce(tmp2.record["FirmName"] ,''),
coalesce(tmp2.record["PostalCode.AddOn"] ,''),
coalesce(tmp2.record["ProcessedBy"] ,''),
coalesce(tmp2.record["MultimatchCount"] ,''),
coalesce(tmp2.record["HouseNumber"] ,''),
coalesce(tmp2.record["StreetName"] ,''),
coalesce(tmp2.record["ApartmentLabel"] ,''),
coalesce(tmp2.record["ApartmentNumber"] ,''),
coalesce(tmp2.record["StreetType"] ,''),
coalesce(tmp2.record["Confidence"] ,''),
coalesce(tmp2.record["Building"] ,''),
coalesce(tmp2.record["POBox"] ,''),
coalesce(tmp2.record["Principality"] ,''),
coalesce(tmp2.record["LeadingDirectional"] ,''),
coalesce(tmp2.record["TrailingDirectional"] ,''),
coalesce(tmp2.record["MatchOnAllStreetFields"] ,''),
coalesce(tmp2.record["MatchOnStreetDirectional"] ,''),
coalesce(tmp2.record["AdditionalInputData"] ,''),
coalesce(tmp2.record["PlaceName"] ,''),
coalesce(tmp2.record["AddressLine1.Input"] ,''),
coalesce(tmp2.record["City.Input"] ,''),
coalesce(tmp2.record["CitySubdivision.Input"] ,''),
coalesce(tmp2.record["StateProvince.Input"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Input"] ,''),
```

```
coalesce(tmp2.record["PostalCode.Input"] ,''),
coalesce(tmp2.record["Country.Input"] ,''),
coalesce(tmp2.record["FirmName.Input"] ,''),
coalesce(tmp2.record["City.Matched"] ,''),
coalesce(tmp2.record["CitySubdivision.Matched"] ,''),
coalesce(tmp2.record["StateProvince.Matched"] ,''),
coalesce(tmp2.record["StateProvinceSubdivision.Matched"] ,''),
coalesce(tmp2.record["StreetName.Matched"] ,''),
coalesce(tmp2.record["StreetType.Matched"] ,''),
coalesce(tmp2.record["Firmname.Matched"] ,''),
coalesce(tmp2.record["Housenumber.Matched"] ,''),
coalesce(tmp2.record["Postalcode.Matched"] ,''),
coalesce(tmp2.record["County.Matched"] ,''),
coalesce(tmp2.record["County"] ,''),
coalesce(tmp2.record["Status"] ,''),
coalesce(tmp2.record["Status.Code"] ,''),
coalesce(tmp2.record["Status.Description"] ,''),
coalesce(tmp2.record["AUS.Address.Class"] ,''),
coalesce(tmp2.record["AUS.Level.Number"] ,''),
coalesce(tmp2.record["AUS.Parcel.ID"] ,''),
coalesce(tmp2.record["AUS.PID"] ,''),
coalesce(tmp2.record["AUS.Principal.Pid"] ,''),
coalesce(tmp2.record["AUS.SA1"] ,''),
coalesce(tmp2.record["CAN.Census.CD"] ,''),
coalesce(tmp2.record["CAN.Census.CMA"] ,''),
coalesce(tmp2.record["CAN.Census.CSD"] ,''),
coalesce(tmp2.record["CAN.Census.CT"] ,''),
coalesce(tmp2.record["CAN.Census.DA"] ,''),
coalesce(tmp2.record["CAN.FormattedStreet.Range"] ,''),
coalesce(tmp2.record["CouldNotValidate"] ,''),
coalesce(tmp2.record["GBR.Aliased.Locality"] ,''),
coalesce(tmp2.record["GBR.Dependent.Locality"] ,''),
coalesce(tmp2.record["GBR.DependentStreet.Name"] ,''),
coalesce(tmp2.record["GBR.DoubleDependent.Locality"] ,''),
coalesce(tmp2.record["GBR.Historic.Postcode"] ,''),
coalesce(tmp2.record["GBR.OSAPR"] ,''),
coalesce(tmp2.record["GBR.RPC"] ,''),
coalesce(tmp2.record["GBR.UPRN"] ,''),
coalesce(tmp2.record["IRL.Eircode"] ,''),
coalesce(tmp2.record["ITA.Historical.Postcode"] ,''),
coalesce(tmp2.record["NZL.Aliased.SUBURB"] ,''),
coalesce(tmp2.record["ParsedAddressLine1.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentLable.Input"] ,''),
coalesce(tmp2.record["ParsedApartmentNumber.Input"] ,''),
coalesce(tmp2.record["ParsedCity.Input"] ,''),
coalesce(tmp2.record["ParsedCitySubDivision.Input"] ,''),
coalesce(tmp2.record["ParsedCountry.Input"] ,''),
coalesce(tmp2.record["ParsedHouseNumber.Input"] ,''),
coalesce(tmp2.record["ParsedPlaceName.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeAddOn.Input"] ,''),
coalesce(tmp2.record["ParsedPostCodeBase.Input"] ,''),
coalesce(tmp2.record["ParsedPostStreetType.Input"] ,''),
coalesce(tmp2.record["ParsedPreStreetType.Input"] ,''),
```

```
coalesce(tmp2.record["ParsedStateProvince.Input"] ,''),
coalesce(tmp2.record["ParsedStateProvinceSubdivision.Input"] ,''),
coalesce(tmp2.record["Subcity"] ,''),
coalesce(tmp2.record["DualAddressParsed.Input"] ,''),
coalesce(tmp2.record["IND.ExtendedResultCode"] ,''),
coalesce(tmp2.record["IND.IsRural"] ,''),
coalesce(tmp2.record["IND.SubLocality"] ,''),
coalesce(tmp2.record["IND.POICategory"] ,''),
coalesce(tmp2.record["JPN.BANCHI"] ,''),
coalesce(tmp2.record["JPN.CHOMOKO"] ,''),
coalesce(tmp2.record["JPN.CHOOAZA"] ,''),
coalesce(tmp2.record["JPN.GO"] ,''),
coalesce(tmp2.record["JPN.JUSHOCODE"] ,''),
coalesce(tmp2.record["USA.DPV"] ,''),
coalesce(tmp2.record["USA.DPV.Footnote"] ,''),
coalesce(tmp2.record["USA.DPV.CMRA"] ,''),
coalesce(tmp2.record["USA.DPV.FalsePositive"] ,''),
coalesce(tmp2.record["USA.DPV.Flags"] ,''),
coalesce(tmp2.record["USA.DPV.NoStat"] ,''),
coalesce(tmp2.record["USA.POBoxOnly"] ,''),
coalesce(tmp2.record["USA.DPV.Vacant"] ,''),
coalesce(tmp2.record["USA.DPV.PBSAFound"] ,''),
coalesce(tmp2.record["USA.LACS"] ,''),
coalesce(tmp2.record["USA.LACS.SeedHit"] ,''),
coalesce(tmp2.record["USA.LACS.ReturnCode"] ,''),
coalesce(tmp2.record["USA.LACS.Indicator"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSAddress"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSRange"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSLeadingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSStreetName"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSSuffix"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSTrailingDirectional"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitD"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSUnitN"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP"] ,''),
coalesce(tmp2.record["USA.LACS.PreLACSMatchedZIP4"] ,''),
coalesce(tmp2.record["USA.LOTCode"] ,''),
coalesce(tmp2.record["USA.LOTSequence"] ,''),
coalesce(tmp2.record["USA.SuiteLink.ReturnCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.MatchCode"] ,''),
coalesce(tmp2.record["USA.SuiteLink.Fidelity"] ,''),
coalesce(tmp2.record["VeriMoveDataBlock"] ,''),
coalesce(tmp2.record["USA.CarrierRouteCode"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Input"] ,''),
coalesce(tmp2.record["USA.PrivateMailbox.Type.Input"] ,''),
coalesce(tmp2.record["USA.RDI"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine1"] ,''),
coalesce(tmp2.record["USA.CASSAddressLine2"] ,''),
coalesce(tmp2.record["USA.CASSCityName"] ,''),
coalesce(tmp2.record["USA.FullCityName"] ,''),
coalesce(tmp2.record["USA.AbbreviatedCityName"] ,''),
```

```
coalesce(tmp2.record["USA.NonMailingCityName"] ,''),
coalesce(tmp2.record["USA.PreferredCityName"] ,''),
coalesce(tmp2.record["USA.PreferredState"] ,''),
coalesce(tmp2.record["USA.EWSFailure"] ,''),
coalesce(tmp2.record["USA.MatchLevel"] ,''),
coalesce(tmp2.record["USA.DefaultMatch"] ,''),
coalesce(tmp2.record["USA.Status"] ,''),
coalesce(tmp2.record["USA.Status.Code"] ,''),
coalesce(tmp2.record["USA.Status.Description"] ,''),
coalesce(tmp2.record["USA.PostalBarcode"] ,''),
coalesce(tmp2.record["USA.BCCheckDigit"] ,''),
coalesce(tmp2.record["USA.AdvancedBarcode"] ,''),
coalesce(tmp2.record["USA.FiveDigitBarcode"] ,''),
coalesce(tmp2.record["USA.ZIPValid"] ,''),
coalesce(tmp2.record["USA.ZIP4Valid"] ,''),
coalesce(tmp2.record["USA.AddressLocation"] ,''),
coalesce(tmp2.record["USA.CongressionalDistrict"] ,''),
coalesce(tmp2.record["USA.FIPSCountyNumber"] ,''),
coalesce(tmp2.record["USA.AltStreetType"] ,''),
coalesce(tmp2.record["USA.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.Range"] ,''),
coalesce(tmp2.record["USA.Parsed.PreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.PostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.StreetSuffix"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.UnitNumber"] ,''),
coalesce(tmp2.record["USA.Parsed.Unit2Designator"] ,''),
coalesce(tmp2.record["USA.Parsed.Unit2Number"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitDesignator"] ,''),
coalesce(tmp2.record["USA.Parsed.PMUnitNumber"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreet"] ,''),
coalesce(tmp2.record["USA.Parsed.AltRange"] ,''),
coalesce(tmp2.record["USA.Parsed.AltPreDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.AltStreetName"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltPostDirectional"] ,''),
coalesce(tmp2.record["USA.Parsed.cAltStreetSuffix"], '') FROM
(SELECT
addressvalidation(${hivevar:engineconf},${hivevar:inputoption},${hivevar:header},
AddressLine1,City,PostalCode,StateProvince,Country) as mygp from
usaddress )
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record;
```

Universal Addressing モジュールの機能

Universal Addressing モジュールの Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. Spectrum™ Data & Address Quality for Big Data SDK UAM モジュールの JAR ファイルを登録します。

```
ADD JAR
/home/hduser/uam/uam.universaladdress.hive.${project.version}.jar;
```

3. 実行する住所品質ジョブの Hive UDF のエイリアスを作成します。

   注： 引用符で囲まれている文字列は、このジョブの実行に必要なクラス名です。

   例:

```
CREATE TEMPORARY FUNCTION addressvalidation as
'com.pb.bdq.uam.process.hive.addressvalidation.AddressValidationUDF';
```

4. Hive フェッチ タスクの変換を有効または無効にします。
   例:

```
set hive.fetch.task.conversion=none;
```

5. エンジン設定を設定するには、`hivevar:engineconf` を使用します。これには、データベース設定、COBOL 実行時パス、プロセス タイプ、DPV DB パス、suiteLinkDBPath、ewsDBPath、rdiDBPath、lacsDBPath、プリロード タイプなどの詳細が含まれます。
   例:

```
set
hivevar:engineconf='{"referenceData":{"dataDir":"/user/hduser/ReferenceData/


AddressQuality/UAM/Data.zip","referenceDataPathLocation":"HDFS"},"cobolRuntimePath":"",
```

```
"modulesDir":"","dpvDbPath":"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip",
"suiteLinkDBPath":"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","ewsDBPath":
"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","rdiDBPath":null,"lacsDBPath":
"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip"}';
```

6. `hivevar:inputoption` パラメータを使用して、入力データの設定を指定します。
例:

```
set
hivevar:inputoption='{"casing":"Mixed","matchMode":"Relaxed","defaultCountry":"GBR",
                        "maximumResults":2,"returnInputAddress":false,
"returnParsedAddress":false,"returnPrecisionCode":false,"returnMatchScore":true,
"mustMatchAddressNumber":false,"mustMatchStreet":false,"mustMatchCity":false,
"mustMatchLocality":false,"mustMatchState":false,"mustMatchStateProvince":false,
"mustMatchPostCode":false,"keepMultiMatch":true,"preferPostalOverCity":false,
                          "cityFallback":true,"postalFallback":true,
                          "validationLevel":"ADDRESS"}';
```

7. `hivevar:generalconf` を使用して、cacheSize、maxAddressObjectCount、maxMemoryUsageMB などの全般的な環境設定を設定します。
例:

```
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';
```

8. Hiveジョブで使用される適切な検証レベルを指定します。現時点では、住所検証のみがサポートされています。この値は VALIDATE に設定します。

例を次に示します。

```
set hivevar:processtype='VALIDATE';
```

9. 入力テーブルのヘッダーフィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

```
set
hivevar:header='inputkeyvalue,AddressLine1,AddressLine2,City,postalcode,
StateProvince,firmname,Country';
```

10. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

    注： このクエリは、各行の入力フィールドを含むキー/値ペアのマップを返します。

```
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(${hivevar:engineconf},
${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},recordid,
addressline1,city,stateprovince,postalcode,country) as mygp from
address)
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record
 ;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
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(${hivevar:engineconf},
${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},
recordid,addressline1,city,stateprovince,postalcode,country) as mygp
 from address)
as addressgroup LATERAL VIEW explode(addressgroup.mygp) tmp2 as record
 ;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

Validate Address

Validate Address は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州または省名など、欠落している郵便情報を追加します。

> **注：** 現在、Validate Address では米国住所のみがサポートされています。

Validate Address は、住所の妥当性確認の有無、返された住所の確信レベル、住所の妥当性が確認できなかった場合の理由など、検証処理に関する結果インジケータも返します。

Validate Address は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを **Universal Addressing** モジュールの各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて*正規化*します。データベースにマッチしなかった場合、Validate Address は、オプションで入力住所の*書式*を整えます。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。

> **注：** Validate Address は CASS 認定™処理をサポートしており、これにより USPS® の郵便料金値引きのための条件を揃えることができます。

> **注：** 最初の Validate Address ジョブを作成および実行する前に、Acushare サービスが実行されていることを確認します。手順については、**Acushare サービスの実行**（13ページ）を参照してください。

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- Register Universal Addressing Module - US [UAM-US] 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.UAMUSAddressingUDF';
-- 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 engine configuration
set
hivevar:engineconf='{"referenceData":{"dataDir":"/home/hduser/ReferenceData/AddressQuality/
UAM/Data","referenceDataPathLocation":"LocaltoDataNodes"},"cobolRuntimePath":"","modulesDir":"",
"dpvDbPath":"/home/hduser/ReferenceData/AddressQuality/UAM/Data","suiteLinkDBPath":"/home/hduser/
ReferenceData/AddressQuality/UAM/Data","ewsDBPath":"/home/hduser/ReferenceData/AddressQuality/UAM/
Data","rdiDBPath":"/home/hduser/ReferenceData/AddressQuality/UAM/Data","lacsDBPath":"/home/hduser/
ReferenceData/AddressQuality/UAM/Data"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData":false,
"outputParsedInput":false,"outputAddressBlocks":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,
"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":"DPV_CONDITON_ALWAYS",
"reportListFileName":"","reportlistProcessorName":"","reportlistNumber":1,"reportMailerAddress":
"","reportMailerName":"","reportMailerCityLine":"","addressLineSearchOnFail":true,
"outputStreetAlias":true,"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,
"dpvDetermineVacancy":false,"outputAbbreviatedAlias":false,"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,
"suppressZplusPhantomCarrierR777":false,"dpvSeedList":null,"lacsSeedList":null,
"zipInputSet":null,"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N",
"DualAddressLogic":"N","PerformASM":"N","ExtractUrb":"N","OutputCasing":"M",
"AddressLineSearchOnFail":"Y","ReportListFileName":"","ReportMailerCityLine":"",
"OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N","OutputStreetNameAlias":"Y",
"ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":"N",
"DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":"N",
"StandardAddressFormat":"C","SuppressZplusPhantomCarrierR777":"N","OutputCountryFormat":"E",
"OutputRecordType":"A    ","HomeCountry":"United
States","ReportMailerAddress":"",
"OutputReport3553":"N","OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"",
"OutputAbbreviatedAlias":"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y",
"OutputReportSummary":"Y","OutputPreferredAlias":"N","StreetMatchingStrictness":
"M","DPVDetermineNoStat":"N","MaximumResults":"10"}}';
```

```
-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';


-- set reference path
set
hivevar:location='/home/hduser/ReferenceData/AddressQuality/UAM/Data';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,DefectNumber,FirmName,PostalCode,
StateProvince';

-- 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
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},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
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:inputconf},${hivevar:header},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       |
+----------+-------------+----------+-------------------------+-------------+
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- Register Universal Addressing Module - US [UAM-US] 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.
UAMUSAddressingUDF';

set hive.fetch.task.conversion=none;

-- 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/acushareInstall/modules/c1p/lib:/home/
hduser/acushareInstall/runtime/bin:/home/hduser/acushareInstall/runtime/lib,
ACU_RUNCBL_JNI_ONLOAD_DISABLE=1,G1RTS=/home/hduser/acushareInstall/runtime;


-- set engine configuration
set
hivevar:engineconf='{"referenceData":{"referenceDataPathLocation":"HDFS","dataDir":
"/user/hadoop/ReferenceData/UAM_US","dataDownloader":{"dataDownloader":"HDFS","localFSRepository":
"/opt/ReferenceData/UAM-US"}},"cobolRuntimePath":"","modulesDir":"","acushareServiceRunning":false,
"unixVersion":"REDHAT7","acushareLicensePath":"","dpvDbPath":"/user/hadoop/ReferenceData/UAM_US",
"suiteLinkDBPath":"/user/hadoop/ReferenceData/UAM_US","ewsDBPath":"/user/hadoop/ReferenceData/
UAM_US","rdiDBPath":"/user/hadoop/ReferenceData/UAM_US","lacsDBPath":"/user/hadoop/ReferenceData/
UAM_US"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData":false,
"outputParsedInput":false,"outputAddressBlocks":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,"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":
"DPV_CONDITON_ALWAYS","reportListFileName":"","reportlistProcessorName":"",
"reportlistNumber":1,"reportMailerAddress":"","reportMailerName":"",
"reportMailerCityLine":"","addressLineSearchOnFail":true,"outputStreetAlias":true,
"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,"dpvDetermineVacancy":false,
"outputAbbreviatedAlias":false,"outputPreferredAlias":false,"outputPreferredCity":
"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,"suppressZplusPhantomCarrierR777":
false,"dpvSeedList":null,"lacsSeedList":null,"zipInputSet":null,
"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N",
```

```
"DualAddressLogic":"N","PerformASM":"N","ExtractUrb":"N","OutputCasing":"M",
"AddressLineSearchOnFail":"Y","ReportListFileName":"",
"ReportMailerCityLine":"","OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N",
"OutputStreetNameAlias":"Y","ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":
"N","DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":
"N","StandardAddressFormat":"C","SuppressZplusPhantomCarrierR777":"N",
"OutputCountryFormat":"E","OutputRecordType":"A ",
"HomeCountry":"United
States","ReportMailerAddress":"","OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"","OutputAbbreviatedAlias":
"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y","OutputReportSummary":"Y",
"OutputPreferredAlias":"N","StreetMatchingStrictness":"M","DPVDetermineNoStat":"N","MaximumResults":
"10"}}';

-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,DefectNumber,FirmName,PostalCode,
StateProvince';

-- 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["Status"],tmp2.record["Status.Description"],tmp2.record["Confidence"],
tmp2.record["AddressLine1"],tmp2.record["InputKeyValue"] FROM ( select
 uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},inputkeyvalue,addressline1,addressline2,city,defectnumber,firmname,
postalcode,stateprovince) as mygp from address_uam) as addressgroup
LATERAL VIEW explode
(addressgroup.mygp) tmp2 as record ;
```

HDFS に配置され、ジョブ用の作業ディレクトリにダウンロードされたリファレンス データ

```
-- Register Universal Addressing Module - US [UAM-US] 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.
UAMUSAddressingUDF';

--Provide reference data zip file to be added to cache
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/UAM/Data.zip;

set hive.fetch.task.conversion=none;

-- 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/acushareInstall/modules/c1p/lib:/home/
hduser/acushareInstall/runtime/bin:/home/hduser/acushareInstall/runtime/lib,
ACU_RUNCBL_JNI_ONLOAD_DISABLE=1,G1RTS=/home/hduser/acushareInstall/runtime;

-- set engine configuration
set
hivevar:engineconf='{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip","referenceDataPathLocation":"HDFS"},"cobolRuntimePath":"","modulesDir":"","dpvDbPath":
"/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip","suiteLinkDBPath":"/user/hduser/
ReferenceData/AddressQuality/UAM/Data.zip","ewsDBPath":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip","rdiDBPath":null,"lacsDBPath":"/user/hduser/ReferenceData/AddressQuality/
UAM/Data.zip"}';

-- set input configuration
set
hivevar:inputconf='{"processType":"VALIDATE","performUSProcessing":true,
"outputStandardAddress":true,"outputAddressElements":false,"outputPostalData"
:false,"outputParsedInput":false,"outputAddressBlocks":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,"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":"DPV_CONDITON_ALWAYS",
"reportListFileName":"","reportlistProcessorName":"","reportlistNumber":1,"reportMailerAddress":"",
"reportMailerName":"","reportMailerCityLine":"","addressLineSearchOnFail"
:true,"outputStreetAlias":true,"outputVeriMoveBlock":false,"dpvDetermineNoStat":false,
"dpvDetermineVacancy":false,"outputAbbreviatedAlias":false,"outputPreferredAlias":false,
"outputPreferredCity":"CITY_OVERRIDE_NAME_ZIP4","performSuiteLink":false,
"suppressZplusPhantomCarrierR777":false,"dpvSeedList":null,"lacsSeedList":null,"zipInputSet":null,
"reportName":null,"jobRequest":false,"properties":{"DPVDetermineVacancy":"N","DualAddressLogic":"N",
```

```
"PerformASM":"N","ExtractUrb":"N","OutputCasing":"M","AddressLineSearchOnFail":"Y",
"ReportListFileName":"","ReportMailerCityLine":
"","OutputFormattedOnFail":"N","OutputFieldLevelReturnCodes":"N","OutputStreetNameAlias":"Y",
"ReportListProcessorName":"","OutputAddressBlocks":"N","ExtractFirm":"N",
"DirectionalMatchingStrictness":"M","OutputPreferredCity":"Z","ReportListNumber":"1",
"FirmMatchingStrictness":"M","KeepMultimatch":"N","StandardAddressPMBLine":"N",
"OutputMultinationalCharacters":"N","PerformSuiteLink":"N","OutputShortCityName":"S",
"DPVSuccessfulStatusCondition":"A","PerformLACSLink":"N","PerformEWS":"N",
"OutputPostalCodeSeparator":"Y","FailOnCMRAMatch":"N","PerformLOT":"N","StandardAddressFormat":
"C","SuppressZplusPhantomCarrierR777":"N","OutputCountryFormat":"E","OutputRecordType":"A

","HomeCountry":"United
States","ReportMailerAddress":"","OutputReport3553":"N",
"OutputVeriMoveDataBlock":"N","PerformRDI":"N","ReportMailerName":"","OutputAbbreviatedAlias":
"N","PerformESM":"N","PerformDPV":"N","OutputVanityCityFormatLong":"Y","OutputReportSummary":
"Y","OutputPreferredAlias":"N","StreetMatchingStrictness":"M","DPVDetermineNoStat":"N",
"MaximumResults":"10"}}';

-- set general configuration
set
hivevar:generalconf='{"dFileType":"SPLIT","dMemoryModel":"MEDIUM","lacsLinkMemoryModel":
"MEDIUM","suiteLinkMemoryModel":"MEDIUM"}';

-- set reference path
set
hivevar:location='/user/hduser/ReferenceData/AddressQuality/UAM/Data.zip';

-- set process type
set hivevar:processtype='VALIDATE';

-- set header
set hivevar:header='InputKeyValue,AddressLine1,AddressLine2,
City,DefectNumber,FirmName,PostalCode,StateProvince';

-- 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["Status"],tmp2.record["Status.Description"],tmp2.record["Confidence"],
tmp2.record["AddressLine1"],tmp2.record["InputKeyValue"] FROM ( select
 uamvalidation
(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},${hivevar:inputconf},
${hivevar:header},inputkeyvalue,addressline1,addressline2,city,defectnumber,firmname,postalcode,
stateprovince) as mygp from address_uam) as addressgroup LATERAL VIEW
explode(addressgroup.mygp)
 tmp2 as record ;
```

Validate Address Global

Validate Address Global は、米国およびカナダ以外の住所に対する高度な住所の正規化および検証機能を提供します。Validate Address Global は、米国およびカナダの住所の妥当性も確認できますが、その他の国の住所の妥当性を確認する能力に優れています。米国およびカナダ以外の住所を大量に処理する場合は、Validate Address Global の使用を検討してください。

Validate Address Global は Universal Addressing モジュールの一部です。

Validate Address Global は、パーシング、検証、書式設定など、いくつもの手順を実行して、住所の品質を高めています。

住所のパーシング、書式設定、および正規化

住所データのフィールド入力の誤りを再構成することは、特に他国の住所で行う場合、複雑で難しい作業です。住所データをコンピュータのシステムに入力する際、曖昧になってしまう部分が多いからです。特に問題なのが、(企業や個人名を通りの住所フィールドに入力するなど) 要素を誤ったフィールドに入力したり、省略形を使用する場合に、言語固有だけでなく、国固有の省略形に変えてしまうケースです。Validate Address Global は住所行の住所要素を識別し、正しいフィールドに割り当てます。これは実際の検証前に行う重要な作業です。再構成を行わなければ、"一致が見つからない" という結果になる可能性があります。

住所要素の正しい識別は、特定のフィールド長要件に合わせて住所を切り捨てたり、短縮しなければならない場合にも重要です。正しい情報が正しいフィールドに割り当てられていれば、特定の切り捨てルールを適用することができます。

- 住所行をパースおよび解析し、個々の住所要素を識別
- 30 を越える文字セットを処理
- 宛先国の郵便ルールに従って住所の書式を整える
- 住所要素を正規化 (AVENUE を AVE に変更するなど)

Global Address Validation

住所の検証は、正しくパースされた住所データを郵便組織または他のデータ プロバイダが提供する参照データベースと比較する訂正処理です。Validate Address Global は、洗練されたファジー マッチング テクノロジーを使用して個々の住所要素を検証し、正しいことを確認するとともに、郵便規格とユーザの優先設定に基づいて出力を正規化および書式設定します。FastCompletion 検証タイプは、簡易住所入力アプリケーションに使用できます。いくつかの住所フィールドには切り捨てられたデータを入力することができ、この入力に基づいて提案を生成します。

住所を完全に検証できない場合もあります。Validate Address Global には、配達可能性によって住所を分類する、ユニークな配達可能性評価機能があります。

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- 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.GlobalAddressingUDF';

-- set Engine configuration
set
hivevar:engineconf='[{"referenceData":{"dataDir":"/home/hduser/ReferenceData/AddressQuality/
Global","referenceDataPathLocation":"LocaltoDataNodes"},"databaseType":"BATCH_INTERACTIVE","preloadingType":"NONE",
"allCountries":true,"supportedCountries":"CAN,USA,AUS"}]';

-- set input configuration
set
hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD","processMatchingScope":
"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":"USA","inputFormatDelimiter":"CRLF",
"resultFormatDelimiter":"CRLF","resultIncludeInputs":false,"resultCountryType":"NAME_EN",
"processOptimizationLevel":"STANDARD","resultPreferredLanguage":"DATABASE","processMode":"BATCH",
"resultPreferredScript":"DATABASE","resultMaximumResults":1,"resultCasing":"NATIVE","properties":
{"Result.MaximumResults":"1","Database.AddressGlobal":"Database","Input.FormatDelimiter":"CRLF",
"Process.Mode":"BATCH","Input.ForceCountryISO3":"","Result.CountryType":"NAME_EN",
"Process.OptimizationLevel":"STANDARD","Result.IncludeInputs":"false","Input.DefaultCountryISO3"
:"USA","Process.EnrichmentAMAS":"false","Result.PreferredScript":"DATABASE","Process.MatchingScope"
:"ALL","Result.Casing":"NATIVE","Result.PreferredLanguage":"DATABASE","Result.StateProvinceType"
:"COUNTRY_STANDARD","Result.FormatDelimiter":"CRLF"}}';

-- set general configuration
set hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,
"maxAddressObjectCount":8,"rangesToExpand":"NONE","flexibleRangeExpansion":"ON",
"enableTransactionLogging":false,"maxMemoryUsageMB":1024,"verbose":false}';

-- set unlock codec
set hivevar:unlockCode='';

-- set header
set
hivevar: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(${hivevar:engineconf},${hivevar:generalconf},${hivevar:inputconf},${hivevar:unlockCode},
${hivevar:header},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(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},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       |
+----------+-------------+-----------+-----------------------------+-------------+
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- 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.GlobalAddressingUDF';

-- set Engine configuration
set hivevar:engineconf='[{"referenceData":{"referenceDataPathLocation"
:"HDFS","dataDir":"/user/hadoop/ReferenceData/Global","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/opt/PitneyBowes/UAM_Global"}},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"PARTIAL","allCountries":true,"supportedCountries":"ALL"}]';



-- set input configuration
set hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD",
"processMatchingScope":"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":"USA",
"inputFormatDelimiter":"CRLF","resultFormatDelimiter":"CRLF","resultIncludeInputs":false,
```

```
"resultCountryType":"NAME_EN","processOptimizationLevel":"STANDARD","resultPreferredLanguage":
"DATABASE","processMode":"BATCH","resultPreferredScript":"DATABASE","resultMaximumResults":1,
"resultCasing":"NATIVE","properties":{"Result.MaximumResults":"1","Database.AddressGlobal":"Database",
"Input.FormatDelimiter":"CRLF","Process.Mode":"BATCH","Input.ForceCountryISO3":"","Result.CountryType":
"NAME_EN","Process.OptimizationLevel":"STANDARD","Result.IncludeInputs":"false",
"Input.DefaultCountryISO3":"USA","Process.EnrichmentAMAS":"false","Result.PreferredScript":
"DATABASE","Process.MatchingScope":"ALL","Result.Casing":"NATIVE","Result.PreferredLanguage"
:"DATABASE","Result.StateProvinceType":"COUNTRY_STANDARD","Result.FormatDelimiter":"CRLF"}}';


-- set general configuration
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';

-- set unlock codec
set hivevar:unlockCode='';

-- set header
set hivevar:header='InputKeyValue,AddressLine1,AddressLine2,
City,PostalCode,StateProvince,FirmName,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["Country"],tmp2.record["Confidence"],tmp2.record["MailabilityScore"],
tmp2.record["HouseNumber"],tmp2.record["AddressLine1"],tmp2.record["StreetName"],
tmp2.record["PostalCode"],tmp2.record["ElementInputStatus"] FROM
(SELECT  globalvalidation(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},inputkeyvalue,
addressline1,addressline2,city,postalcode,stateprovince,firmname,country)

 as mygp from address_global) as addressgroup LATERAL VIEW explode
(addressgroup.mygp) tmp2 as record ;
```

HDFS に配置され、ジョブ用の作業ディレクトリにダウンロードされたリファレンス データ

```
-- 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.GlobalAddressingUDF';

-- Add Reference Data zipped Files
```

```
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/AD/AD2032017_590.zip;
ADD ARCHIVE
hdfs:///user/hduser/ReferenceData/AddressQuality/AD/AD3032017_590.zip;

-- set Engine configuration
set
hivevar:engineconf='[{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/AD/AD2032017_590.zip",
"referenceDataPathLocation":"HDFS"},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"NONE","allCountries":false,"supportedCountries":"ALL"},
{"referenceData":{"dataDir":"/user/hduser/ReferenceData/AddressQuality/AD/AD3032017_590.zip",
"referenceDataPathLocation":"HDFS"},"databaseType":"BATCH_INTERACTIVE",
"preloadingType":"NONE","allCountries":false,"supportedCountries":"ALL"}]';


-- set input configuration
set hivevar:inputconf='{"resultStateProvinceType":"COUNTRY_STANDARD",
"processMatchingScope":"ALL","inputForceCountryISO3":null,"inputDefaultCountryISO3":
"USA","inputFormatDelimiter":"CRLF","resultFormatDelimiter":"CRLF","resultIncludeInputs":false,
"resultCountryType":"NAME_EN","processOptimizationLevel":"STANDARD","resultPreferredLanguage"
:"DATABASE","processMode":"BATCH","resultPreferredScript":"DATABASE","resultMaximumResults":1,
"resultCasing":"NATIVE","properties":{"Result.MaximumResults":"1","Database.AddressGlobal":
"Database","Input.FormatDelimiter":"CRLF","Process.Mode":"BATCH","Input.ForceCountryISO3":
"","Result.CountryType":"NAME_EN","Process.OptimizationLevel":"STANDARD","Result.IncludeInputs"
:"false","Input.DefaultCountryISO3":"USA","Process.EnrichmentAMAS":"false",
"Result.PreferredScript":"DATABASE","Process.MatchingScope":"ALL","Result.Casing":"NATIVE",
"Result.PreferredLanguage":"DATABASE","Result.StateProvinceType":"COUNTRY_STANDARD",
"Result.FormatDelimiter":"CRLF"}}';

-- set general configuration
set
hivevar:generalconf='{"cacheSize":"LARGE","maxThreadCount":8,"maxAddressObjectCount":8,
"rangesToExpand":"NONE","flexibleRangeExpansion":"ON","enableTransactionLogging":false,
"maxMemoryUsageMB":1024,"verbose":false}';
-- set unlock codec
set hivevar:unlockCode='';

-- set header
set
hivevar:header='InputKeyValue,AddressLine1,AddressLine2,City,PostalCode,StateProvince,FirmName,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["Country"],tmp2.record["Confidence"],tmp2.record["MailabilityScore"],
tmp2.record["HouseNumber"],tmp2.record["AddressLine1"],tmp2.record["StreetName"],tmp2.record
["PostalCode"],tmp2.record["ElementInputStatus"]
FROM (SELECT
globalvalidation(${hivevar:engineconf},${hivevar:generalconf},
${hivevar:inputconf},${hivevar:unlockCode},${hivevar:header},inputkeyvalue,
addressline1,addressline2,city,postalcode,stateprovince,firmname,country)
```

```
as mygp from address_global) as addressgroup LATERAL VIEW
explode(addressgroup.mygp) tmp2 as record ;
```

Validate Address Loqate

Validate Address Loqate は、郵便当局の住所データを使用して、住所を正規化し、妥当性を確認します。Validate Address Loqate は、情報を修正し、管轄の郵便当局が推奨する書式で住所の書式を整えることができます。また、郵便番号、都市名、州/省名など、欠落している郵便情報を追加します。

Validate Address Loqate は、Validate Address Loqate が住所の妥当性を確認したかどうか、返された住所の確信レベル、住所の妥当性が確認できなかった場合はその理由など、検証処理に関する結果インジケータも返します。

Validate Address Loqate は、住所のマッチングと正規化において、住所行をコンポーネントに分割し、それらを Universal Addressing モジュールの各種データベースの内容と比較します。マッチを検出した場合、入力住所をデータベース情報に合わせて正規化します。データベースにマッチしなかった場合、Validate Address Loqate は、オプションで入力住所の書式を整えます。書式設定プロセスでは、該当する郵便当局の規則に従って住所行の構成を試みます。
ValidateAddressLoqate は、Universal Addressing モジュールに含まれています。

サンプル Hive スクリプト

```
-- 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.LoqateAddressingUDF';

-- 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 set
hivevar:processconf='{"processType":"VALIDATE","includeMatchedAddressElements":
false,"includeStandardizedInputAddressElements":false,"returnAddressDataBlocks":false,
```

```
"casing":"Mixed","outputReportSummary":false,"includeResultCodesforIndividualFields":false,
"returnMultipleAddresses":false,"failedOnMultiMatchFound":false,"countryFormat":"ENGLISH",
"defaultCountry":"USA","scriptAlphabet":"InputScript","returnGeocodedAddressFields":false,
"acceptanceLevel":"Level0","minimumMatchScore":0,"formatDataUsingAMASConventions":false,
"singleFieldDuplicateHandling":false,"multiFieldDuplicateHandling":false,
"nonStandardFieldDuplicateHandling":false,"outputFieldDuplicateHandling":false,
"returnMultipleAddressCount":10,"duplicateHandling":false,"includeStandardAddress":true}';

-- set general configuration
set
hivevar:generalconf='{"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
hivevar:engineconf='{"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 hivevar:location='/home/hduser/ReferenceData/AddressQuality/Loqate';

-- set process type
set hivevar:processtype='VALIDATE';


-- set input header
set
hivevar:header='InputKeyValue,AddressLine1,City,StateProvince,PostalCode,Country';


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(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:processconf},${hivevar:location},${hivevar:header},inputkeyvalue,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(${hivevar:engineconf},${hivevar:generalconf},${hivevar:processtype},
${hivevar:processconf},${hivevar:location},${hivevar:header},inputkeyvalue,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
```

Universal Name モジュールの関数

Universal Name モジュールの Hive UDF の使用

Hive UDF の各ジョブを実行する場合は、Hive クライアントでの以下のステップの個別実行を 1 つのセッションの範囲内で行うことも、必要なすべてのステップをまとめた HQL ファイルを作成してそれを一度に実行することもできます。

1. Hive クライアントで、必要な Hive データベースにログインします。
2. Spectrum™ Data & Address Quality for Big Data SDK UNM モジュールの JAR ファイルを登録します。

   ```
   ADD JAR <Directory path>/unm.hive.${project.version}.jar;
   ```

3. 実行するデータ品質ジョブの Hive UDF のエイリアスを作成します。
   例:

   ```
   CREATE TEMPORARY FUNCTION opennameparser as
   'com.pb.bdq.unm.process.hive.
   opennameparser.OpenNameParserUDF';
   ```

4. リファレンス データのパスを指定します。
   - **リファレンス データが HDFS 上にある場合**
   - リファレンス データがジョブ用の**作業ディレクトリにダウンロードされる**場合

- リファレンス データがアーカイブされていないファイル形式の場合は、リファレンス ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData","dataDownloader":{"dataDownloader":"DC"}}';
```

- リファレンス データがアーカイブ済みの形式の場合は、リファレンス ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"./referenceData.zip","dataDownloader":
{"dataDownloader":"DC"}}';
```

- リファレンス データがジョブ用の**ローカル ノードにダウンロードされる**場合この場合は、リファレンス データ ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS",
"dataDir":"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":
"HDFS","localFSRepository":"/local/download"}}';
```

- **リファレンス データがローカル パスにある場合**: クラスタの各ノードでデータが同じパスに配置されるようにします。

  リファレンス ディレクトリを次のように設定します。

```
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"LocaltoDataNodes",
"dataDir":"/home/data/referenceData"}';
```

5. ジョブの環境設定と詳細情報を指定して、それぞれの変数または設定プロパティに代入します。

   注： ルールは JSON 形式である必要があります。

   例を次に示します。

```
set hivevar: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}';
```

注： それぞれのジョブ環境設定の設定プロパティを使用してください。例えば、それぞれのサンプル HQL ファイルに記載されている `pb.bdq.match.rule`、`pb.bdq.match.express.column`、`pb.bdq.consolidation.sort.field` などです。

6. 入力テーブルのヘッダーフィールドをカンマ区切り形式で指定し、変数または設定プロパティに割り当てます。

```
set hivevar:header='inputrecordid,Name,nametype';
```

7. ジョブを実行してジョブ出力をコンソールに表示するには、次の例に示すようにクエリを記述します。

```
select adTable.adid["Name"], adTable.adid["NameScore"],
adTable.adid["CultureCode"] from (select
opennameparser(${hivevar:rule}, ${hivevar:refdir}, ${hivevar:header},
inputrecordid, name, nametype) as tmp1 from nameparser) as tmp LATERAL
 VIEW explode(tmp1) adTable AS adid;
```

ジョブを実行してジョブ出力を指定されたファイルに書き出すには、以下の例に示すようにクエリを記述します。

```
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(${hivevar:rule}, ${hivevar:refdir},
${hivevar:header}, inputrecordid, name, nametype) as tmp1 from
nameparser) as tmp LATERAL VIEW explode(tmp1) adTable AS adid;
```

注： 先ほど UDF に対して定義したエイリアスを使用してください。

Open Name Parser

OpenNameParser は、名前データフィールドにある個人名、企業名、またはその他の名称を構成要素に分解します。パースされたこれらの名前要素は、名前のマッチング、名前の正規化、複数レコード名の統合など、他の自動化処理に使用できます。

OpenNameParser は、次の処理を行います。

- 名前が担う機能を示すために、その名前のタイプを特定します。名前エンティティ タイプは、個人名と企業名の 2 つのグループに分かれます。それぞれのグループには、さらに複数のサブグループがあります。
- パーシングに使う構文を把握するために、名前の形式を特定します。個人名は、通常、自然な (署名) 順序または逆の順序に従います。企業名は、通常、階層型の順序に従います。
- 名前を構成する各要素が名前全体に占める構文上の関連性を識別するために、要素を特定してラベル付けします。個人名の構文は、敬称、名、ミドルネーム、姓、接尾語、アカウントを示す用語、その他の個人名要素で構成されます。企業名の構文は、企業名や接尾語などで構成されます。
- 結合された個人名と企業名をパースし、それらを 1 つのレコードとして残すか、複数のレコードに分割します。Examples of conjoined names include "Mr.and Mrs.John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- 出力をレコードまたはリストとしてパースします。
- パーシングによる訂正の信頼度を示すパーシング スコアを割り当てます。

サンプル Hive スクリプト

ローカル ノードに配置されたリファレンス データ

```
-- 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 hivevar: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 on each node of the cluster at the same path.
set hivevar:refereceDataDetails='{"referenceDataPathLocation":
"LocaltoDataNodes","dataDir":"/home/data/referenceData"}';

-- set header
```

```
set hivevar:header='inputrecordid,Name,nametype';

set hive.fetch.task.conversion=none;
-- 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(${hivevar:rule},
${hivevar:refereceDataDetails}, ${hivevar: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(${hivevar:rule},
${hivevar:refereceDataDetails}, ${hivevar: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           |
+----------------------------+-----+-------------------------------+
```

HDFS に配置され、ジョブ用のローカル ノードにダウンロードされたリファレンス データ

```
-- 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 hivevar: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 on each node of the cluster at the same path.
set
hivevar:refereceDataDetails='{"referenceDataPathLocation":"HDFS","dataDir"
:"/home/data/dm/referenceData","dataDownloader":{"dataDownloader":"HDFS","localFSRepository":"/local/download"}}';

-- set header
set hivevar:header='inputrecordid,Name,nametype';

set hive.fetch.task.conversion=none;
-- 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(${hivevar:rule},
${hivevar:refereceDataDetails},
 ${hivevar: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(${hivevar:rule},
${hivevar:refereceDataDetails},
 ${hivevar: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           |
+----------------------------+-----+--------------------------------+
```

7 - レポート カウンタ

このセクションの構成

レポート カウンタ	386
Advanced Matching モジュール	386
Global Addressing モジュール	389
Universal Addressing モジュール	391
Universal Name モジュール	396

レポート カウンタ

レポート カウンタは、MapReduce または Spark ジョブを実行しているときにのみ表示されます。出力は、使用するクエリ構文に基づき、コンソールに表示されるか、ファイルとしてダンプされます。HIVE ジョブの場合、レポート カウンタは Hadoop クラスタのリソース マネージャに表示されます。

Advanced Matching モジュール

Interflow Match レポート

Interflow Match は、2 つの入力レコード ストリーム内の類似するデータ レコード間でマッチを検出します。最初のレコード ストリームはサスペクト レコードのソースで、2 番目のストリームは候補レコードのソースです。

Interflow Match では、マッチ グループ条件 (マッチ キー等) を使用して、特定のサスペクト レコードと重複する可能性があるレコードのグループを識別します。

レポート

Interflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクト レコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Express マッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。

AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。
UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATE_CANDIDATES	検出された重複候補の総数。
TOTAL_DUPLICATE_SCORE	すべての重複の合計マッチ スコア。

Intraflow Match レポート

Intraflow Match は、単一の入力ストリーム内の類似するデータ レコード間でマッチを検出します。データフローの別のコンポーネントで定義または作成したフィールドに基づいて、階層的なルールを作成できます。

レポート

Intraflow Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

INPUT_RECORDS	マッチング ソート実行前のマッチング ステージにおけるレコードの数。
DUPLICATE_RECORDS	マッチ グループ内の重複レコード (サスペクト レコードまたは候補レコード) の数。
UNIQUE_RECORDS	各マッチ グループで他のレコードにマッチしないサスペクトまたは候補レコードの数。 マッチ グループ内に 1 つしか存在していないレコードであれば、サスペクトは自動的にユニーク レコードとなります。
MATCH_GROUPS	(グループ化) マッチ キーでグループ化されたレコード。

DUPLICATE_COLLECTIONS	コレクション番号によってグループ化されたサスペクトレコードとその重複レコードで構成される、重複コレクションの数。
EXPRESS_MATCHES	1 つのコレクションで作成された Express マッチの数。 Express マッチは、サスペクトと候補が指定されたフィールド内の内容に正確にマッチした場合に作成され、通常は ExpressMatchKey が Match Key Generator によって提供されます。Express マッチが作成された場合、サスペクトと候補の重複を判定するためのそれ以上の処理は行われません。
AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
TOTAL_DUPLICATES	検出された重複の総数。
TOTAL_SCORE	すべての重複の合計マッチ スコア。

Transactional Match レポート

Transactional Match は、重複を特定するため、サスペクト レコードと、あるグループのレコードを照合します。これらのレコードはまず、選択した列によりグループ化され、最初のレコードがサスペクト レコードとしてマークされます。グループの残りすべてのレコードは候補レコードと呼ばれ、サスペクト レコードと照合されます。

候補レコードが重複の場合は、コレクション番号が割り当てられ、そのマッチ レコード タイプに重複が設定され、その候補レコードが書き出されます。グループ内のマッチしない候補にはコレクション番号 0 が割り当てられ、そのラベルにユニークが設定され、その候補が書き出されます。

レポート

Transactional Match ジョブを使用して、ジョブの結果を監視することができます。使用可能なカウンタは次の通りです：

AVERAGE_SCORE	すべての重複の平均マッチ スコア。 有効な値は 0 ～ 100 です。0 は精度の低いマッチを意味し、100 は完全一致を意味します。
INPUT_SUSPECTS	マッチャーが他のレコードとの照合を試みた入力ストリーム内のレコードの数。
SUSPECTS_WITH_DUPLICATES	少なくとも 1 つの候補レコードと一致した入力サスペクトの数。

UNIQUE_SUSPECTS	どの候補レコードとも一致しなかった入力サスペクトの数。
SUSPECTS_WITH_CANDIDATES	マッチ グループ内に候補レコードが少なくとも 1 つある、つまり照合の試みが少なくとも 1 回は行われた入力サスペクトの数。
SUSPECTS_WITHOUT_CANDIDATES	マッチ グループ内に候補レコードがない、つまり照合の試みが行われなかった入力サスペクトの数。
TOTAL_DUPLICATES_SCORE	すべての重複の合計マッチ スコア。
TOTAL_DUPLICATES	検出された重複の総数。

Global Addressing モジュール

Global Address Validation レポート

以下のカウンタは、Global Address Validation ジョブが実行されるサポート対象国全体にわたるレポート統計情報を提供します。

サポートされている国のリストについては、**ISO 国コードとコーダー サポート**（417ページ）を参照してください。使用可能なカウンタは次の通りです：

レコードのカウンタ

- *TOTAL_INPUT*- 入力レコードの総数。
- *TOTAL_COUNTRY_RECORDS*- 当該国に関して処理された入力レコードの総数。

一致要素のカウンタ

このセクションには、一致要素に関するサマリ情報が国別に示されます。

- *HOUSE_NUMBER_MATCHED*: 家番号が一致したレコードの数。
- *STREET_NAME_MATCHED*: 通り名が一致したレコードの数。
- *CTY_MATCHED*: 都市名が一致したレコードの数。
- *POST_CODE_MATCHED*: 郵便番号が一致したレコードの数。
- *STATE_PROVINCE_MATCHED*: 州/省が一致したレコードの数。

適合率コードのカウンタ

ジョブで一致したレコードの数に関する統計が適合率コード別に示されます。適合率コードは、個々のレコードの住所マッチングの精度を表します。

適合率コード Z のカテゴリ

Z カテゴリは、マッチングが郵便番号レベルで成立したことを示します。

- *PRECISION_Z1*- ZIP Code (郵便番号 1) に一致するレコードの数。
- *PRECISION_Z2*- ZIP + 2 (郵便番号 2 への部分一致) に適合するレコードの数。
- *PRECISION_Z3*- ZIP + 4 (郵便番号 2) に一致するレコードの数。

適合率コード G のカテゴリ

地理的レベルの候補は、文字 G で始まる適合率コードを返します。適合率コードの G に続く数字は、一致の精度に関するより詳細な情報を示します。

- *PRECISION_G1*- 州/省に一致するレコードの数。
- *PRECISION_G2*- 郡/地域に一致するレコードの数。
- *PRECISION_G3*- 都市/町に一致するレコードの数。
- *PRECISION_G4*- 郊外/村に一致するレコードの数。

適合率コード S のカテゴリ

通りレベルの候補は、文字 S で始まる適合率コードを返します。適合率コードの S に続く文字は、一致の精度に関するより詳細な情報を示します。

- *PRECISION_SX*- 交差点で検証されたレコードの数。
- *PRECISION_SC*- 最も近いセグメントから予測された家レベルで一致したレコードの数。
- *PRECISION_SL*- 通りレベルで地方下位区分 (ブロックまたはセクター) に一致したレコードの数。
- *PRECISION_SG*- 地方 (areaName3) の中心部、または地形特性から導き出された地方レベルのジオコードに一致したレコードの数。
- *PRECISION_S0*- 住所の一部がソース データに一致したレコードの数。
- *PRECISION_S1*- ZIP Code にあるポイントに一致したレコードの数。
- *PRECISION_S2*- ZIP + 2 にあるポイントに一致したレコードの数。
- *PRECISION_S3*- ZIP + 4 にあるポイントに一致したレコードの数。
- *PRECISION_S4*- 通りレベルで一致したレコードの数。
- *PRECISION_S5*- 通りの住所に一致したレコードの数。
- *PRECISION_S6*- ポイント ZIP セントロイドにあるポイントに一致したレコードの数。
- *PRECISION_S7*- 家屋の間を補間して得られた通りの住所に一致したレコードの数。
- *PRECISION_S8*- 通りの住所または家番号に一致したレコードの数。

適合率コード B のカテゴリ

- *PRECISION_B1*- 未検証 PO Box に一致したレコードの数。
- *PRECISION_B2*- 検証済み PO Box に一致したレコードの数。

信頼度のカウンタ

このセクションには、さまざまな信頼度で一致したレコードの数に関する統計が国別に示されます。返される住所の信頼度は 0 ～ 100 の範囲で変化します。*0* は、失敗を表します。*100* は、マッチ結果の正しさに対する信頼度が非常に高いことを表します。

- *CONF_LOW*- 40 未満の信頼度で一致するレコード (*CONF_10*、*CONF_20* など)。
- *CONF_MID*- 40 ～ 85 の間の信頼度で一致するレコード (*CONF_50*、*CONF_80* など)。
- *CONF_HIGH*- 85 を超える信頼度で一致するレコード (*CONF_90*、*CONF_100* など)。
- *信頼性*- 信頼度が確認されるレコードの総数。この値はレコードの総数と同じです。

Universal Addressing モジュール

Validate Address のレポート

Validate Address の場合、次の 3 種類のレポートを生成できます。

> **注：** レポートの生成方法については、『*Dataflow Designer ガイド*』を参照してください。

1. 住所検証サマリ レポート
2. CASS レポート 3553 サマリ レポート
3. CASS 詳細レポート

> **注：** CASS レポート 3553 と CASS 詳細レポートは、CASS 認定™モードで生成することができます。

住所検証サマリ レポート

住所検証サマリには次の詳細が表示されます。

- **ジョブに関する統計** - 入力レコードの数、処理された住所レコードの数、住所検証が試行されたレコードの総数、正常に一致したレコードの総数、不一致レコードの総数、および正常に返された標準住所の数。

- **レコードの検証/修正に関する統計** - 住所マッチングで確認された元の郵便番号の数、住所マッチングで修正された郵便番号の数、保持されている元の郵便番号の数、利用できない郵便番号の数、および住所検証が試みられたレコードの総数。
- **一致レコードに関する統計** - 入力時に有効だったレコードの総数、修正されたレコードの総数、および正常に一致したレコードの総数。
- **不一致レコードに関する統計** - 通りの不一致のレコード数、家の不一致のレコード数、数値範囲の不一致のレコード数、および不一致レコードの総数。
- **処理されたレコードに関する統計** - 米国、カナダおよび国際向けに処理された一致レコードおよび不一致レコード。処理されたレコードの総数も表示されます。

  注： さらに、住所検証サマリには、マッチング サマリ、一致レコード、不一致レコード、および合計レコードに関して、グラフも表示されます。

CASS 3553 サマリ レポート

USPS から値引きを受けるには、USPS CASS 3553 レポートを郵便物と一緒に提出する必要があります。このレポートは、以下の詳細を表形式で表示します。

ソフトウェア

- CASS 認定企業およびソフトウェアの名前、ソフトウェアのバージョンおよび構成。
- Z4Change 認定企業およびソフトウェアの名前、ソフトウェアのバージョンおよび構成。
- DirectDPV 認定企業およびソフトウェアの名前、ソフトウェアのバージョンおよび構成。

リスト

- 住所リストをコーディングした会社の名前、および/または ZIP+4 マッチングを実行した会社の名前。
- 各リストの処理日。
- 使用された各データベース パッケージの日付。
- 住所リストの名前または識別番号。
- メーリングを生成するために使用されるリストの数。
- リストがコード化されたときに提出された住所レコードの総数。

出力 - CASS 処理についてまとめた出力。以下のレベルで検証されたレコードの総数で構成されます。

- ZIP+4
- DPV
- Z4 変更
- ダイレクト DPV
- 5 桁の配達ルート
- 配達ルート (CR-RT)

- Enhanced Line Of Travel (eLOT) シーケンス番号

検証されたレコードのバリデーション期間も表示されます。

メーラー - リストを処理した個人の署名、氏名、住所、および書式が署名された日付。

Qualitative Statistical Summary (QSS) - リスト プロセッサにおいて、内容がメールストリームに入る前に住所リストの品質を評価できるようにする情報。

- High Rise Default - 通り住所レベルでのマッチングには成功したが補助的な有効住所とのマッチングができなかったレコードの数。
- High Rise Exact - 主要な住所と補助的な住所に完全一致したレコードの数。
- Rural RTE Default - ルート番号レベルで検証されたが主要な有効範囲とのマッチングができなかった地方配送路住所レコードの数。
- Rural RTE Exact - ルート番号レベルでは一致したが主要な有効範囲とのマッチングができなかった地方配送路住所レコードの数。
- Locatable Address Conversion System (LACS)/ LACSLink System - LACS/LACSLink システムを介して変換された住所レコードの数。
- EWS (Early Warning System) - 現在の US Postal Service (USPS、米国郵政公社) の ZIP + 4 ファイルには存在しない新しい住所レコード。
- SuiteLink - SuiteLink によって修正された住所の数。

USPS Form 3553 の詳細については、**www.usps.com** を参照してください。

CASS 詳細レポート

CASS 認定™ 処理により、USPS CASS 詳細レポートも生成されます。郵便料金の割引を受ける際に、このレポートの提出は必須ではありません。レポートには、次の詳細情報が含まれます。

ジョブ識別および一般情報 - 処理日およびジョブ名。

CASS 実行情報

- ソフトウェア名
- 処理されたレコードの総数
- 郵便番号付加レコードの総数
- 郵便番号付加に失敗したレコードの総数
- ZIP+4 マッチ統計: PO Box、RR Default、RR Exact、Street など。
- 処理情報: ZIP+4 にマッチまたはマッチしない入力住所の数など。

DPV 実行情報

LACS and LACSLink 実行情報

SuiteLink 実行情報

注： DPV、LACS and LACSLink、および Suitelink についての一致/不一致の結果と妥当性は、CASS 詳細レポートの最後で脚注として説明されます。

SDK を使用しているときの CASS 設定の詳細については、**Validate Address MapReduce ジョブの使用**（156ページ） と **Validate Address Spark ジョブの使用**（159ページ） を参照してください。

Validate Address Global レポート

これらのカウンタは、Validate Address Global ジョブが実行されるすべてのサポート対象国についてのレポート統計情報を提供します。

サポートされている国のリストについては、**ISO 国コードとコーダー サポート**（417ページ）を参照してください。使用可能なカウンタは次の通りです：

サマリ カウンタ

サマリ カウンタは、国々における特定の国タイプの値の和を示します。

例えば、`SUMMARY_FAILED_COUNT` は、特定の Validate Address Global ジョブが実行されているすべてのサポート対象国についての `FAILED_COUNT` カウンタの値の和です。

- ステータスの詳細: 各国の検証および修正の結果をリストします。
  - SUMMARY_STATUS_I4_COUNT— 修正できなかったが、かなりの確率で正しく配達される住所。
  - SUMMARY_STATUS_I2_COUNT— 修正できず、正しく配達される可能性が低い住所。
  - SUMMARY_STATUS_V_COUNT— 入力時に正しかった住所。
  - SUMMARY_STATUS_C_COUNT— Validate Address Global によって修正された住所。
  - SUMMARY_STATUS_I3_COUNT— 修正できなかったが、配達される可能性が十分にある住所。
  - SUMMARY_STATUS_S_COUNT— 正しくパーシングされた住所。
  - SUMMARY_FAILED_COUNT— 検証も修正もパーシングもできなかった住所。
  - COUNTRY— 住所検証が実行される国コードのカンマ区切りリスト。
  - SUMMARY_CASING— 出力の大文字/小文字を区別する方法。詳細については、『*Addressing ガイド*』の「*Validate Address Global*」の「*オプション*」セクションを参照してください。
- ジョブの概要: ジョブの概要をリストします。
  - SUMMARY_END_TIME— ジョブが終了した日付と時刻。
  - SUMMARY_START_TIME— ジョブが開始された日付と時刻。
  - SUMMARY_CHARSET— 処理された文字セット。

- SUMMARY_DEFAULT_COUNTRY— [デフォルトの国 (ISO3 形式)] オプションで指定されたデフォルトの国。

国固有のカウンタ

これらのカウンタは、さまざまなサポート対象国に対するレポート統計を提供します。それぞれの国ラベルは、カウンタ値に対応する国コードで始まります。

例えば、以下のカウンタは、米国のレポート統計を示しています。

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

Validate Address Global ジョブを実行したすべてのサポート対象国についても同様のカウンタが表示されます。

Validate Address Loqate レポート

Validate Address Loqate ジョブは、次のレポート カウンタを表示します。

Input Name/Address

- **Input Record Count** — ジョブの入力住所の総数。
- **Address Records Processed** — ジョブの入力住所の総数。
- **Total Records For Which Address Validation Attempted** — 検証が試みられた入力レコードの数。
- **Total Records Successfully Matched** — 検証または訂正された入力住所の数。結果がステータス "F" にならなかった入力住所の数です。
- **Total Unmatched Records** — 検証または訂正できなかった入力住所の数。結果がステータス "F" になった入力住所の数と等しくなります。
- **Standard Address Returned Successfully** — Validate Global Address が正規化したマッチしない (失敗した) 住所の数。正規化は、オプション **[マッチしなかった場合に正規化データを返す]** が有効になっている場合にのみ実行されます。詳細については、「出力データ オプション」を参照してください。

郵便番号の検証と修正

- **Original Postal Code Confirmed Via Address Match** — 郵便番号の ACR コンポーネント ステータスが 2 である住所の数。
- **Postal Code Corrected Via Address Match** — Validate Global Address によって誤りが訂正された入力郵便番号の数。
- **Original Postal Code Retained** — 郵便番号の ACR コンポーネント ステータスが 1 である住所の数。
- **No Postal Code Available** — 住所に対応する郵便番号が郵便データに含まれていません。

データベース内の既知の住所にマッチする入力住所

1. **Total Records Valid On Input** — 正しいことが確認された住所の数。
2. **Total Corrected** — Validate Global Address が訂正した住所の数。
3. **Total Records Successfully Matched** — 検証または訂正に成功した住所の総数。

確認または修正できなかった入力住所

1. **Street Mismatch** — 通り名を検証または訂正できなかった住所の数。
2. **House Mismatch** — 家番号を検証または訂正できなかった住所の数。
3. **Total Unmatched Records** — 検証または訂正できなかった住所の総数。

処理したレコード

- **RecordsProcessedByLOQATE** — ジョブで処理したレコードの総数。

Universal Name モジュール

Open Name Parser レポート

Open Name Parser では、入力レコードの合計数や、名前データが含まれないレコードの合計数など、ジョブに関するサマリ統計を提供します。

一般的な結果

INPUT_RECORDS	入力内のレコード数。
NO_NAME_DATA_RECORDS	入力内の、パースする名前データを含まないレコードの数。

NAMES_PARSED_OUT	入力内の、パースされた名前の数。
LOWEST_NAME_PARSING_SCORE	入力内の名前に与えられたパーシング スコアの最小値。
HIGHEST_NAME_PARSING_SCORE	入力内の名前に与えられたパーシング スコアの最大値。
AVERAGE_NAME_PARSING_SCORE	入力内のパースされたすべての名前に与えられたパーシング スコアの平均値。

個人名のパーシング結果

PERSONAL_NAME_RECORDS	入力内の個人名の数。
CONJOINED_NAMES_PARSED	結合名を含むレコードからパースされた名前の数。 例えば、入力に、2 つの結合名を持つレコードが 5 つ、3 つの結合名を持つレコードが 7 つある場合、このフィールドのカウンタ値は、(5 x 2) + (7 x 3) = 31 になります。
TWO_CONJOINED_NAMES_RECORDS	結合名を 2 つ含む入力レコードの数。
THREE_CONJOINED_NAMES_RECORDS	結合名を 3 つ含む入力レコードの数。
TITLE_OF_RESPECT_NAMES	パースされた名前のうち、敬称を含む名前の数。
MATURITY_SUFFIX_NAMES	パースされた名前のうち、世代接尾語を含む名前の数。
GENERAL_SUFFIX_NAMES	パースされた名前のうち、一般接尾語を含む名前の数。
ACCOUNT_DESCRIPTION_PERSONAL_NAMES	パースされた名前のうち、アカウント説明を含む名前の数。
TOTAL_REVERSE_ORDER_NAMES	パースされた名前のうち、逆順序の名前の数 (出力フィールド `IsReverseOrder` が "True")。

企業名のパーシング結果

BUSINESS_NAME_RECORDS	企業名を含む入力レコードの数。
FIRM_SUFFIX_NAMES	パースされた名前のうち、企業接尾語を含む名前の数。
ACCOUNT_DESCRIPTION_BUSINESS_NAMES	入力レコードのうち、アカウント説明を含む名前の数。
TOTAL_DBA_RECORDS	略語 Doing Business As (DBA) を含み、出力フィールド `isPersonal` と `isFirm` の両方が "True" となる入力レコードの数。

TOTAL_PARSED	パースされた名前の総数。
TOTAL_NAME_PARSING_SCORE	すべての名前の合計パーシング スコア。

付録

このセクションの構成

例外 400
列挙体 402
ISO 国コードとモジュール サポート 416

A - 例外

このセクションの構成

例外メッセージ 401

例外メッセージ

例外 - Java API

- `<Classname>.<Member>` is null or empty.
- `GroupbyMROption.numReduceTasks = 0` min values should be 1.
- `maxNumOfDuplicates` = 0 min values should be 1.
- No files available in the specified path.
- Unable to identify the input file as either Suspect or Candidate File.
- ExpressMatchKey defined but not available for the record\t
- Unable to get the FileName of the InputSplit.
- Unable to initialize engine.
- Error processing consolidated records.

例外 - Hive ユーザ定義関数

- _FUNC_ must have the minimum arguments.
- Unable to initialize engine.Rule passed: `<Rule used>`
- Expected argument type: String.Received argument type: `<Mismatched Type>`
- Exception: `<Header string>` configuration missing.
- Error processing consolidated records: `<Exception details>`
- Exception: Sort field column `<column name>` missing from job configuration.

B - 列挙体

このセクションの構成

一般的な列挙 403
Universal Addressing 列挙体 407

一般的な列挙

列挙 `MatchingAlgorithm`

パッケージ: `com.pb.bdq.api.matcher`

クラス: `Algorithm`

1. Acronym (頭字語)
2. CharacterFrequency (文字出現頻度)
3. DaitchMokotoffSoundex (Daitch-Mokotoff Soundex)
4. 日付
5. DoubleMetaphone (Double Metaphone)
6. EditDistance (編集距離)
7. EuclideanDistance (ユークリッド距離)
8. ExactMatch (完全一致)
9. Initials (頭文字)
10. JaroWinklerDistance (Jaro-Winker 距離)
11. KeyboardDistance (キーボード距離)
12. Koeln
13. KullbackLeiblerDistance (Kullback-Liebler 距離)
14. Metaphone
15. SpanishMetaphone (スペイン語 Metaphone)
16. Metaphone3
17. NGramDistance (NGram 距離)
18. NGramSimilarity
19. NumericString (数値文字列)
20. Nysiis
21. Phonix
22. Soundex
23. SubString (部分文字列)
24. SyllableAlignment (シラブル配置)

列挙 `Algorithm`

パッケージ: `com.pb.bdq.api.matchkeygenerator`

クラス: `MatchKeyRule`

1. Soundex
2. Metaphone

3. SpanishMetaphone (スペイン語 Metaphone)
4. DoubleMetaphone (Double Metaphone)
5. Nysiis
6. Phonix
7. Metaphone3
8. Koeln
9. Consonant (子音)
10. SubString (部分文字列)

列挙 RecordSeparator

パッケージ: `com.pb.bdq.common.job`

クラス: `FilePath`

1. WINDOWS
2. LINUX
3. MACINTOSH

列挙 ReferenceDataPathLocation

パッケージ: `com.pb.bdq.common.job`

Enum 定数	説明
HDFS	リファレンス データは HDFS に配置されます。
LocaltoDataNodes	リファレンス データはクラスタ内の使用可能なすべてのデータ ノードに配置されます。

列挙 Operation

パッケージ: `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

列挙 MatchingMethod

パッケージ: `com.pb.bdq.api.matcher`

クラス: `ParentMatchRule`

1. AllTrue
2. AnyTrue
3. BasedOnThreshold

列挙 ScoringMethod

パッケージ: `com.pb.bdq.api.matcher`

クラス: `MatchRule`

1. 最小値
2. 最大値
3. 平均
4. WeightedAverage
5. VectorSummation

列挙 MissingDataMethod

パッケージ: `com.pb.bdq.api.matcher`

クラス: `MatchRule`

1. IgnoreBlanks
2. CountAs100
3. CountAs0
4. CompareBlanks

列挙 JoinDetail.JoinType

パッケージ: `com.pb.bdq.dim.api.detai`

クラス: `JoinDetail`

1. 全体
2. 内部
3. LeftOuter

列挙 JoinType

パッケージ: `com.pb.bdq.api.consolidation`

クラス: `ConjoinedRule`

1. または
2. AND

列挙 IncludeTerm
パッケージ: `com.pb.bdq.api.advtransformer`

クラス: `TableDataExtraction`

1. ExtractedData
2. NonExtractedData
3. TermNeither

列挙 Extract
パッケージ: `com.pb.bdq.api.advtransformer`

クラス: `TableDataExtraction`

1. ExtractTerm
2. ExtractNWordsLeft
3. ExtractNWordsRight

列挙 AdvTransformerExtractionType
パッケージ: `com.pb.bdq.api.advtransformer`

クラス: `AbstractAdvancedTransformerRules`

1. TableData
2. RegularExpression

列挙 MatchRuleType
パッケージ: `com.pb.bdq.api.matcher`

クラス: `MatchRule`

1. Parent
2. Child

列挙 SortInput
パッケージ: `com.pb.bdq.api.matcher`

クラス: `MatchRule`

1. CHARS
2. TERMS

列挙 TableLookupAction
パッケージ: `com.pb.bdq.api.tablelookup`

クラス: `AbstractTableLookupRule`

1. 正規化
2. 分類
3. 特定

Universal Addressing 列挙体

列挙 `DatabaseType`

パッケージ: `com.pb.bdq.api.uam.global`

クラス: `GlobalAddressingEngineConfiguration`

1. BATCH_INTERACTIVE
2. FASTCOMPLETION
3. CERTIFIED

列挙 `PreloadingType`

パッケージ: `com.pb.bdq.api.uam.global`

クラス: `GlobalAddressingEngineConfiguration`

1. NONE
2. FULL
3. PARTIAL

列挙 `CountryCodes`

パッケージ: `com.pb.bdq.api.uam`

説明: サポートされているすべての国に割り当てられる英文字のコード。

列挙 `StateProvinceType`

パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. COUNTRY_STANDARD
2. ABBREVIATION
3. EXTENDED

列挙 `CountryType`

パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `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

列挙 `PreferredScript`

パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. DATABASE
2. POSTAL_ADMIN_PREF
3. POSTAL_ADMIN_ALT
4. LATIN
5. LATIN_ALT
6. ASCII_SIMPLIFIED
7. ASCII_EXTENDED

列挙 `PreferredLanguage`

パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. DATABASE
2. ENGLISH

列挙 Casing
パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. NATIVE
2. UPPER
3. LOWER
4. MIXED
5. NOCHANGE

列挙 OptimizationLevel
パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. NARROW
2. 標準
3. WIDE

列挙 Mode
パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. BATCH
2. CERTIFIED
3. FASTCOMPLETION
4. 対話型
5. PARSE

列挙 MatchingScope
パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. ALL
2. LOCALITY_LEVEL
3. STREET_LEVEL
4. DELIVERYPOINT_LEVEL

列挙 FormatDelimiter
パッケージ: `com.pb.bdq.api.uam.global`

インターフェイス: `GlobalAddressingInputOption`

1. CRLF

2. LF
3. CR
4. SEMICOLON
5. COMMA
6. TAB
7. PIPE
8. SPACE

列挙 ExhaustedAction

パッケージ: `com.pb.bdq.api.uam.loqate`

クラス: `LoqateAddressingGeneralConfiguration`

1. GROW
2. BLOCK
3. FAIL

列挙 AcceptanceLevel

パッケージ: `com.pb.bdq.api.uam.loqate.validate`

クラス: `LoqateAddressingValidateConfiguration`

1. Level0
2. Level1
3. Level2
4. Level3
5. Level4
6. Level5

列挙 OutputCasing

パッケージ: `com.pb.bdq.api.uam.loqate.validate`

クラス: `LoqateAddressingValidateConfiguration`

1. 混在
2. 大文字

列挙 CountryFormat

パッケージ: `com.pb.bdq.api.uam.loqate.validate`

クラス: `LoqateAddressingValidateConfiguration`

1. ENGLISH
2. ISO
3. UPU

列挙 ScriptAlphabet
パッケージ: `com.pb.bdq.api.uam.loqate.validate`

クラス: `LoqateAddressingValidateConfiguration`

1. InputScript
2. ネイティブ
3. Latin_English

列挙 CacheSize
パッケージ: `com.pb.bdq.api.uam.global`

クラス: `GlobalAddressingGeneralConfiguration`

1. NONE
2. SMALL
3. LARGE

列挙 RangesToExpand
パッケージ: `com.pb.bdq.api.uam.global`

クラス: `GlobalAddressingGeneralConfiguration`

1. NONE
2. ONLY_WITH_VALID_ITEMS

列挙 FlexibleRangeExpansion
パッケージ: `com.pb.bdq.api.uam.global`

クラス: `GlobalAddressingGeneralConfiguration`

1. ON
2. OFF

列挙 CasingType
パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. MIXED
2. UPPER

列挙 CityNameFormat
パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. CITY_FORMAT_LONG
2. CITY_FORMAT_SHORT

3. CITY_FORMAT_STANDARD

列挙 OutputCountryFormat

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. ENGLISH
2. FRENCH
3. GERMAN
4. SPANISH
5. ISO
6. UPU

列挙 DualAddressLogic

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. DUAL_NORMAL
2. DUAL_PO_BOX
3. DUAL_STREET

列挙 StandardAddressFormat

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_FORMAT_COMBINED_UNIT
2. STANDARD_ADDRESS_FORMAT_SEPARATE_UNIT
3. STANDARD_ADDRESS_FORMAT_SEPARATE_DUAL

列挙 StreetMatchingStrictness

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

列挙 FirmMatchingStrictness

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL

2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

列挙 DirectionalMatchingStrictness

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. MATCHING_STRICTNESS_EQUAL
2. MATCHING_STRICTNESS_TIGHT
3. MATCHING_STRICTNESS_MEDIUM
4. MATCHING_STRICTNESS_LOOSE

列挙 StandardAddressPMBLine

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. STANDARD_ADDRESS_PMB_LINE_NONE
2. STANDARD_ADDRESS_PMB_LINE_1
3. STANDARD_ADDRESS_PMB_LINE_2

列挙 PreferredCity

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. CITY_OVERRIDE_NAME_ZIP4
2. CITY_USPS_STATE_FILE
3. CITY_PRIMARY_NAME

列挙 DPVFileType

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressGeneralConfiguration`

1. SPLIT
2. FULL
3. FLAT

列挙 DPVMemoryModel

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO

3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

列挙 LacsLinkMemoryModel

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

列挙 SuiteLinkMemoryModel

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressGeneralConfiguration`

1. PICO
2. MICRO
3. SMALL
4. MEDIUM
5. LARGE
6. HUGE

列挙 DPVSuccessStatusCondition

パッケージ: `com.pb.bdq.api.universaladdress`

クラス: `UniversalAddressInputConfiguration`

1. DPV_CONDITON_FULL
2. DPV_CONDITON_PARTIAL
3. DPV_CONDITON_ALWAYS

列挙 UAMCASSReportType

パッケージ: `com.pb.bdq.uam.common`

1. CASS_3553
2. CASS_DETAIL
3. CASS_DETAIL2
4. CASS_DETAIL3

Enum AddressValidationProcessType
パッケージ: com.pb.bdq.api.uam.addressvalidation

1. VALIDATE

Enum AddressValidationInputOption.MatchMode
パッケージ: com.pb.bdq.api.uam.addressvalidation

1. カスタム
2. 完全一致
3. 緩和
4. 標準

C - ISO 国コードとモジュール サポート

このセクションの構成

ISO 国コードとコーダー サポート 417

ISO 国コードとコーダー サポート

次の表に、各国の 2 桁と 3 桁の ISO コードと、Validate Address International (VAI)、Validate Address Global (VAG)、Validate Address Loqate (VAL) の各コーダーに対するサポートのレベルを示します。レベルは次のように定義されています。

- レベル A — 通りレベルのデータが提供されています。
- レベル B — 都市、郵便番号、またはその両方のデータが提供されています。
- レベル C — 国データが提供されています。
- '-' — サポートされていない国

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Afghanistan	AF	AFG	B	B	A
Aland Islands	AX	ALA	B	-	A
アルバニア	AL	ALB	B	B	A
アルジェリア	DZ	DZA	B	B	A
American Samoa	AS	ASM	B	-	A
Andorra	AD	AND	A	A	A
Angola	AO	AGO	B	B	A
Anguilla	AI	AIA	B	B	B
Antarctica	AQ	ATA	C	B	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Antigua And Barbuda	AG	ATG	A	B	B
アルゼンチン	AR	ARG	A	A	A
Armenia	AM	ARM	B	A	A
Aruba	AW	ABW	A	B	A
オーストラリア	AU	AUS	A	A	A
オーストリア	AT	AUT	A	A	A
Azerbaijan	AZ	AZE	B	B	A
バハマ	BS	BHS	A	B	A
Bahrain	BH	BHR	A	A	A
Bangladesh	BD	BGD	B	B	A
Barbados	BB	BRB	A	B	A
Belarus	BY	BLR	A	A	A
Belgium	BE	BEL	A	A	A
Belize	BZ	BLZ	A	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Benin	BJ	BEN	B	B	A
Bermuda	BM	BMU	A	A	A
Bhutan	BT	BTN	B	B	B
Bolivia, Plurinational State Of	BO	BOL	B	B	A
Bonaire, Saint Eustatius And Saba	BQ	BES	B	-	B
Bosnia And Herzegovina	BA	BIH	B	B	A
Botswana	BW	BWA	B	B	A
Bouvet Island	BV	BVT	C	-	-
ブラジル	BR	BRA	A	A	A
British Indian Ocean Territory	IO	IOT	B	B	B
Brunei Darussalam	BN	BRN	A	A	A
Bulgaria	BG	BGR	A	A	A
Burkina Faso	BF	BFA	A	B	A
Burundi	BI	BDI	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Cambodia	KH	KHM	B	B	A
Cameroon	CM	CMR	B	B	A
カナダ	CA	CAN	C	A	A
Cape Verde	CV	CPV	B	B	A
Cayman Islands	KY	CYM	A	B	A
Central African Republic	CF	CAF	B	B	A
Chad	TD	TCD	B	B	A
チリ	CL	CHL	A	A	A
中国	CN	CHN	B	A	A
Christmas Island	CX	CXR	B	-	B
Cocos (Keeling) Islands	CC	CCK	B	-	B
コロンビア	CO	COL	B	A	A
Comoros	KM	COM	B	B	B
Congo	CG	COG	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Congo, The Democratic Republic Of The	CD	COD	B	B	A
Cook Islands	CK	COK	A	B	B
Costa Rica	CR	CRI	B	B	A
Côte d'Ivoire	CI	CIV	B	B	A
Croatia	HR	HRV	A	A	A
Cuba	CU	CUB	A	B	A
Curacao	CW	CUW	B	B	B
キプロス	CY	CYP	A	A	A
チェコ共和国	CZ	CZE	A	A	A
デンマーク	DK	DNK	A	A	A
Djibouti	DJ	DJI	B	B	B
Dominica	DM	DMA	B	B	B
Dominican Republic	DO	DOM	B	A	A
Ecuador	EC	ECU	A	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Egypt	EG	EGY	B	B	A
El Salvador	SV	SLV	A	B	A
Equatorial Guinea	GQ	GNQ	B	B	A
Eritrea	ER	ERI	B	B	A
Estonia	EE	EST	A	A	A
Ethiopia	ET	ETH	B	B	A
Falkland Islands (Malvinas)	FK	FLK	A	B	A
Faroe Islands	FO	FRO	A	B	B
Fiji	FJ	FJI	A	B	B
フィンランド	FI	FIN	A	A	A
フランス	FR	FRA	A	A	A
French Guiana	GF	GUF	A	-	A
French Polynesia	PF	PYF	B	-	B
French Southern Territories	TF	ATF	C	-	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Gabon	GA	GAB	B	B	A
Gambia	GM	GMB	B	B	A
Georgia	GE	GEO	B	A	A
ドイツ	DE	DEU	A	A	A
Ghana	GH	GHA	B	B	A
Gibraltar	GI	GIB	A	A	A
Greece	GR	GRC	B	A	A
Greenland	GL	GRL	B	A	B
Grenada	GD	GRD	B	B	B
Guadeloupe	GP	GLP	A	-	A
Guam	GU	GUM	C	-	A
Guatemala	GT	GTM	B	B	A
Guernsey	GG	GGY	C	-	A
Guinea	GN	GIN	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Guinea-Bissau	GW	GNB	B	B	A
Guyana	GY	GUY	B	B	A
Haiti	HT	HTI	A	A	A
Heard Island and McDonald Islands	HM	HMD	C	-	-
Holy See (Vatican City State)	VA	付加価値税	A	A	A
Honduras	HN	HND	B	B	A
Hong Kong	HK	HKG	A	A	A
ハンガリー	HU	HUN	A	A	A
アイスランド	IS	ISL	A	A	A
インド	IN	IND	A	A	A
Indonesia	ID	IDN	A	A	A
Iran, Islamic Republic Of	IR	IRN	B	B	A
イラク	IQ	IRQ	B	B	A
アイルランド	IE	IRL	A	A	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Isle Of Man	IM	IMN	C	-	A
Israel	IL	ISR	B	A	A
Italy	IT	ITA	A	A	A
Jamaica	JM	JAM	B	B	A
日本	JP	JPN	A	A	A
Jersey	JE	JEY	C	-	A
ヨルダン	JO	JOR	B	B	A
Kazakhstan	KZ	KAZ	B	A	A
Kenya	KE	KEN	B	B	A
Kiribati	KI	KIR	B	B	B
Korea, Democratic People's Republic Of	KP	PRK	B	B	A
Korea, Republic Of	KR	KOR	B	A	A
Kuwait	KW	KWT	A	A	A
Kyrgyzstan	KG	KGZ	A	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Lao People's Democratic Republic	LA	LAO	B	B	A
Latvia	LV	LVA	A	A	A
Lebanon	LB	LBN	A	B	A
Lesotho	LS	LSO	B	B	A
Liberia	LR	LBR	A	B	B
Libyan Arab Jamahiriya	LY	LBY	B	B	B
Liechtenstein	LI	LIE	B	A	A
Lithuania	LT	LTU	A	A	A
Luxembourg	LU	LUX	A	A	A
Macao	MO	MAC	A	A	A
Macedonia, Former Yugoslav Republic Of	MK	MKD	A	B	A
Madagascar	MG	MDG	A	B	A
Malawi	MW	MWI	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
マレーシア	MY	MYS	A	A	A
Maldives	MV	MDV	A	A	A
Mali	ML	MLI	B	B	A
マルタ	MT	MLT	A	A	A
Marshall Islands	MH	MHL	C	-	A
Martinique	MQ	MTQ	A	-	A
Mauritania	MR	MRT	B	B	A
Mauritius	MU	MUS	A	B	A
Mayotte	YT	MYT	A	-	B
メキシコ	MX	MEX	B	A	A
Micronesia, Federated States Of	FM	FSM	C	-	A
Moldova, Republic Of	MD	MDA	A	A	A
Monaco	MC	MCO	A	A	A
Mongolia	MN	MNG	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
モンテネグロ	ME	MNE	B	B	A
Montserrat	MS	MSR	A	B	B
Morocco	MA	MAR	A	A	A
Mozambique	MZ	MOZ	B	B	A
Myanmar	MM	MMR	B	B	A
Namibia	NA	NAM	A	B	A
Nauru	NR	NRU	B	B	A
Nepal	NP	NPL	B	B	A
Netherlands	NL	NLD	A	A	A
New Caledonia	NC	NCL	B	-	B
ニュージーランド	NZ	NZL	A	A	A
Nicaragua	NI	NIC	B	B	B
Niger	NE	NER	A	B	A
Nigeria	NG	NGA	A	A	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Niue	NU	NIU	C	B	B
Norfolk Island	NF	NFK	B	B	B
Northern Mariana Islands	MP	MNP	C	-	A
ノルウェー	NO	NOR	A	A	A
Oman	OM	OMN	B	B	A
Pakistan	PK	PAK	A	B	A
Palau	PW	PLW	C	-	A
Palestinian Territory, Occupied	PS	PSE	C	-	B
Panama	PA	PAN	B	A	A
Papua New Guinea	PG	PNG	B	B	B
Paraguay	PY	PRY	A	A	A
Peru	PE	PER	B	B	A
Philippines	PH	PHL	A	A	A
Pitcairn	PN	PCN	B	B	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
ポーランド	PL	POL	A	A	A
ポルトガル	PT	PRT	A	A	A
Puerto Rico	PR	PRI	C	-	A
Qatar	QA	QAT	A	A	A
Reunion	RE	REU	A	-	A
ルーマニア	RO	ROU	A	A	A
Russian Federation	RU	RUS	B	A	A
Rwanda	RW	RWA	A	B	A
Saint Barthelemy	BL	BLM	A	-	A
Saint Helena, Ascension & Tristan Da Cunha	SH	SHE	B	B	A
Saint Kitts and Nevis	KN	KNA	B	A	A
Saint Lucia	LC	LCA	B	B	A
Saint Martin (French Part)	MF	MAF	A	-	B
Saint Pierre and Miquelon	PM	SPM	B	-	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Saint Vincent and the Grenadines	VC	VCT	A	B	A
Samoa	WS	WSM	B	B	A
San Marino	SM	SMR	B	A	A
Sao Tome and Principe	ST	STP	B	B	A
Saudi Arabia	SA	SAU	A	A	A
Senegal	SN	SEN	A	B	A
セルビア	RS	SRB	A	A	A
Seychelles	SC	SYC	B	B	B
Sierra Leone	SL	SLE	A	B	A
シンガポール	SG	SGP	A	A	A
Sint Maarten (Dutch Part)	SX	SXM	B	B	B
Slovakia	SK	SVK	A	A	A
Slovenia	SI	SVN	A	A	A
Solomon Islands	SB	SLB	A	B	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Somalia	SO	SOM	B	B	A
South Africa	ZA	ZAF	A	A	A
South Georgia And The South Sandwich Islands	GS	SGS	B	B	B
South Sudan	SS	SSD	B	B	A
Spain	ES	ESP	A	A	A
Sri Lanka	LK	LKA	B	B	A
Sudan	SD	SDN	B	B	A
Suriname	SR	SUR	B	B	A
Svalbard And Jan Mayen	SJ	SJM	A	-	B
Swaziland	SZ	SWZ	B	B	A
スウェーデン	SE	SWE	A	A	A
Switzerland	CH	CHE	A	A	A
Syrian Arab Republic	SY	SYR	B	B	A
Taiwan, Province of China	TW	TWN	A	A	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Tajikistan	TJ	TJK	B	B	B
Tanzania, United Republic Of	TZ	TZA	B	B	A
Thailand	TH	THA	B	A	A
Timor-Leste	TL	TLS	B	A	B
Togo	TG	TGO	B	B	A
Tokelau	TK	TKL	C	B	B
Tonga	TO	TON	B	B	A
Trinidad and Tobago	TT	TTO	B	B	A
Tunisia	TN	TUN	B	B	A
Turkey	TR	TUR	A	A	A
Turkmenistan	TM	TKM	B	B	B
Turks And Caicos Islands	TC	TCA	A	B	B
Tuvalu	TV	TUV	B	B	B
Uganda	UG	UGA	B	B	A

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Ukraine	UA	UKR	B	A	A
United Arab Emirates	AE	ARE	A	A	A
英国	GB	GBR	A	A	A
米国	US	USA	C	A	A
United States Minor Outlying Islands	UM	UMI	C	-	B
ウルグアイ	UY	URY	A	A	A
Uzbekistan	UZ	UZB	B	B	A
Vanuatu	VU	VUT	B	B	B
Venezuela, Bolivarian Republic Of	VE	VEN	B	A	A
Viet Nam	VN	VNM	B	A	A
Virgin Islands, British	VG	VGB	B	B	B
Virgin Islands, U.S.	VI	VIR	C	-	A
Wallis and Futuna	WF	WLF	A	-	B

ISO 国名	ISO 3166-1 Alpha 2	ISO 3166-1 Alpha 3	VAI	VAG	VAL
Western Sahara	EH	ESH	C	B	B
イエメン	YE	YEM	B	B	B
Zambia	ZM	ZMB	A	A	A
Zimbabwe	ZW	ZWE	B	B	A

著作権に関する通知

USPS® 情報

Pitney Bowes Inc. は、ZIP + 4® データベースを光学および磁気媒体に発行および販売する非独占的ライセンスを所有しています。CASS、CASS 認定、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、および ZIP + 4 の各商標は United States Postal Service が所有します。United States Postal Service に帰属する商標はこれに限りません。

Pitney Bowes Inc. は、NCOA^{Link}® 処理に対する USPS® の非独占的ライセンスを所有しています。

Pitney Bowes Software の製品、オプション、およびサービスの価格は、USPS® または米国政府によって規定、制御、または承認されるものではありません。RDI™ データを利用して郵便送料を判定する場合に、使用する郵便配送業者の選定に関するビジネス上の意思決定が USPS® または米国政府によって行われることはありません。

データ プロバイダおよび関連情報

このメディアに含まれて、Pitney Bowes Software アプリケーション内で使用されるデータ製品は、各種商標によって、および次の 1 つ以上の著作権によって保護されています。

© Copyright United States Postal Service.All rights reserved.

© 2014 TomTom.All rights reserved.TomTom および TomTom ロゴは TomTom N.V の登録商標です。

© 2016 HERE

Fuente: INEGI (Instituto Nacional de Estadística y Geografía)

電子データに基づいています。© National Land Survey Sweden.

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

このプログラムの一部は著作権で保護されています。© Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Second Decimal, LLC

© Copyright Canada Post Corporation

この CD-ROM には、Canada Post Corporation が著作権を所有している編集物からのデータが収録されています。

© 2007 Claritas, Inc.

Geocode Address World データ セットには、**http://creativecommons.org/licenses/by/3.0/legalcode** に存在するクリエイティブ コモンズ アトリビューション ライセンス (「アトリビューション ライセンス」) の下に提供されている GeoNames Project (**www.geonames.org**) からライセンス供与されたデータが含まれています。お客様による GeoNames データ (Spectrum™ Technology Platform ユーザ マニュアルに記載) の使用は、アトリビューション ライセンスの条件に従う必要があり、お客様と Pitney Bowes Software, Inc. との契約と、アトリビューション ライセンスの間に矛盾が生じる場合は、アトリビューション ライセンスのみに基づいてそれを解決する必要があります。お客様による GeoNames データの使用に関しては、アトリビューション ライセンスが適用されるためです。

3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com