pitney bowes

Spectrum™ Technology Platform

バージョン 12.0 SP1

API ガイド

目次

1 - はじめに

API の一般的な使用手順 5
データをサービスに渡す方法 6
サポートされるコンパイラ 7
サードパーティ ライブラリ 11
ネットワーク プロトコルとポート 11
サンプル アプリケーションの使用 12
HTTPS の使用 12
タイムアウト値を増やす 14

2 - C API

C API の概要 16
Server 32
Service 37
Message 38
DataTable 52
DataRow 60

3 - C++ API

C++ API の概要 74
Server 88
Service 92
Message 92
DataTable 103
DataRow 111

4 - COM API

はじめに 123
Server 128
Service 130
Message 131
DataTable 139
DataRow 145
Map 153

5 - Java API

はじめに 158
Server 163
Service 169
Message 171
DataTable 177
DataRow 183

6 - .NET API

はじめに 191
Server 195
Service 198
Message 199
EnhancedDataTable 206

7 - ManagementAPI メソッド (非推奨)

はじめに 211
GetLicenseInfo 211
GetVersionInfo 212

8 - モジュール サービス

Address Now モジュール 215
Enterprise Geocoding モジュール 270
GeoConfidence モジュール 358
Universal Addressing モジュール 361
Universal Name モジュール 568

9 - Spectrum™ Technology Platform について

Spectrum™ Technology Platform とは 580
エンタープライズ データ管理アーキテクチャ 581
Spectrum™ Technology Platformのアーキテクチャ 585
モジュールとコンポーネント 590

第 章 : 付録

付録 A :
ISO 国コードとモジュール サポート 597

1 - はじめに

このセクションの構成

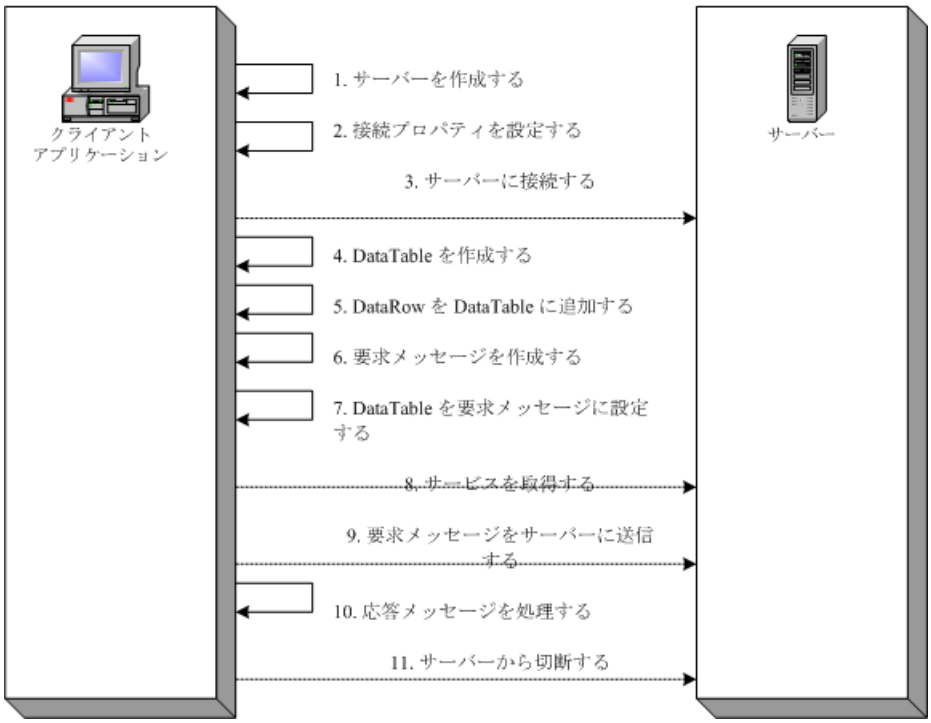
API の一般的な使用手順 5
データをサービスに渡す方法 6
サポートされるコンパイラ 7
サードパーティ ライブラリ 11
ネットワーク プロトコルとポート 11
サンプル アプリケーションの使用 12
HTTPS の使用 12
タイムアウト値を増やす 14

API の一般的な使用手順

以下に、Spectrum™ Technology Platform API の基本的な使用手順を示します。

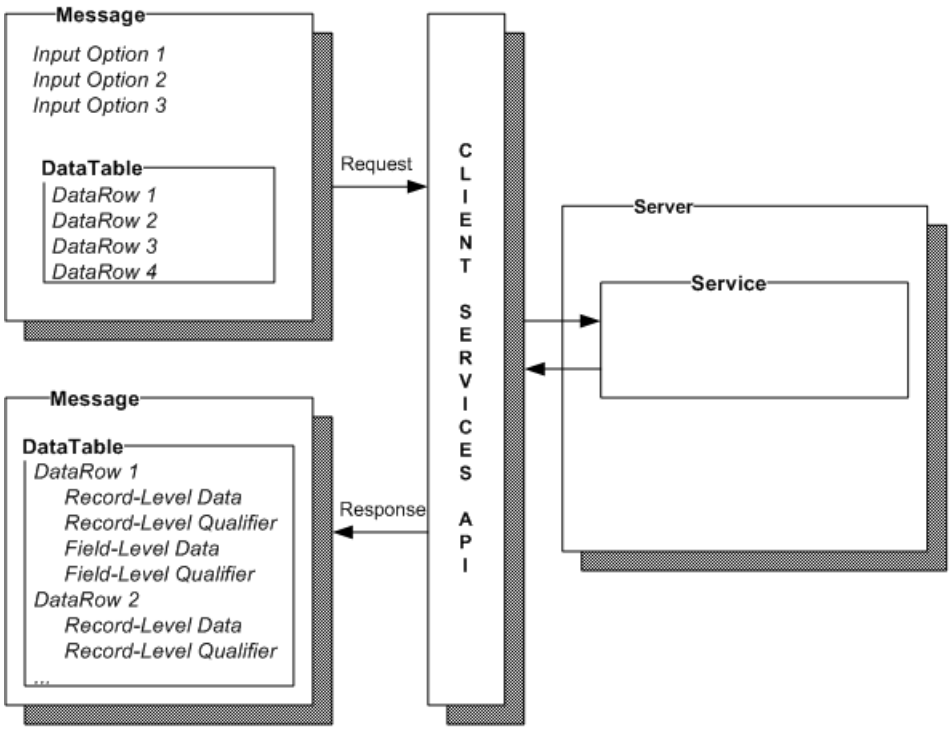
1. Server インスタンスを作成します。
2. 接続プロパティ (接続タイプ、ホスト、ポートなど) を設定します。
3. サーバーに接続します。
4. DataTable を作成します。
5. レコードを DataTable に追加します。
6. 要求メッセージを作成します。
7. DataTable を要求メッセージに設定します。
8. サービスを取得します。
9. 要求メッセージをサーバーに送信します。
10. 応答メッセージを処理します。
11. サーバーから切断します。

Client API の使用手順



データをサービスに渡す方法

以下の図に、API を介してデータをサービスに渡す方法を示します。



メッセージ

Message コンポーネントを使って、入力データを Spectrum™ Technology Platform サービスに送信し、出力データをサービスから受信します。

また、Message コンポーネントを使うと、サービスのデフォルトの処理オプションよりも優先することもできます。 サービスのデフォルトのオプションは Management Console で設定されます。 例えば、ValidateAddress サービスは、出力をすべて大文字で生成するか小文字と大文字の混在するスタイルで生成できます。 ほとんどの場合は、すべて大文字で出力を生成します。 ただし、アプリケーションによっては小文字と大文字が混在する出力が必要とされることがあります。この場合、ValidateAddress サービスに対しては大文字と小文字の区別に関するデフォルトを大文字のみに設定し、その単一のアプリケーションに対しては API を使用して大文字と小文字の区別に関するデフォルトよりも優先します。 指定されているデフォルトに応じて要求を処理する必要がある場合、要求に入力オプションを指定する必要はありません。

Message のプロパティには、コンテキスト プロパティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、オプション プロパティ (サービス固有の実行時オプション)、エラー プロパティ (エラー クラス、エラー メッセージ、エラー スタックトレース) などがあります。

DataTable

DataTable コンポーネントには、入出力データのレコードが格納されます。 このクラスに関連付けられたメソッドを使って、出力用の列名を定義しレコードをデータセットに追加します。 Reset メソッドと Next メソッドは、サーバーからの応答で返された結果に対して反復処理を実行するために使用します。

DataRow

DataRow には、スキーマ情報と一連のデータ行が格納されます。 個々のレコードは、データ行に収められます。 各出力のデータ行には、レコードレベル判定情報、フィールドレベル データ、およびフィールドレベル判定情報があります。

レコードレベル判定情報は、レコードの処理を定義します。 レコードレベル判定情報には、要求のステータス (成功、失敗、またはエラー) や、出力レコードの精度に関する確信度などが含まれます。

フィールドレベル データには、検証、正規化、または拡張されたレコードが含まれます。

フィールドレベル判定情報には、特定のフィールドに関する追加のデータが含まれます。 例えば、USPS の分類法に従う私設私書箱タイプは、フィールドレベル判定情報です。

サーバー

Server コンポーネントは、Spectrum™ Technology Platform サーバーを表します。 Server コンポーネントを使うと、サーバー上の特定のサービスに接続し、アクセスし、接続を切断することができます。

サービス

Service コンポーネントは、送信するメッセージの処理 (入力メッセージの送信と応答メッセージの返却) に使用します。 Service コンポーネントの唯一のメソッドは、Process メッセージです。

サポートされるコンパイラ

Spectrum™ Technology Platform クライアント SDK は、ここに挙げるバージョン以降のコンパイラとランタイムでサポートされます。

Java

クライアント SDK パッケージ ディレクトリ: `clientSDK/platforms/java`

クライアント SDK には Java JDK バージョン 1.4 以上が必要です。これはクライアント SDK と共にインストールされません。

Windows 32 ビット

- JDK: 1.4
- C コンパイラ: MSVC 6.0 SP3、MSVC 2003、MSVC 2005、MSVC 2008
- C++ コンパイラ: MSVC 6.0 SP3、MSVC 2003、MSVC 2005、MSVC 2008
- C# .NET: Microsoft .NET Framework 1.1
- Visual Basic: MS Visual Basic 6.0

Windows 64 ビット

- JDK: 1.4
- C コンパイラ: MSVC 2005、MSVC 2008
- C++ コンパイラ: MSVC 2005、MSVC 2008

HP-UX RISC

- JDK: 1.4
- C コンパイラ: cc: HP92453-01 A.11.01.21 HP C (バンドル) コンパイラ
- C++ コンパイラ: aCC: HP aC++ B3910B A.03.30 HP aC++ B3910B A.03.27

clientSDK 32 bit lib は、次のライブラリにリンクされます。

- libpthread.1
- librt.2
- libnsl.1
- libxti.2

clientSDK 64 bit lib は、次のライブラリにリンクされます。

- libpthread.1
- libnsl.1
- librt.2
- libdl.1
- libc.2
- libxti.2
- libdl.1

HP-UX Itanium

- JDK: 1.4

- C コンパイラ: cc: HP aC++/ANSI C B3910B A.06.05
- C++ コンパイラ: aCC: HP aC++/ANSI C B3910B A.06.05

clientSDK 32 bit lib は、次のライブラリにリンクされます。

- libpthread.so.1
- libnsl.so.1
- librt.so.1
- libxti.so.1
- libdl.so.1

clientSDK 64 bit lib は、次のライブラリにリンクされます。

- libpthread.so.1
- libnsl.so.1
- librt.so.1
- libxti.so.1
- libdl.so.1

Red Hat (32 ビット)

- オペレーティング システム: Red Hat Linux 2.4.9-e.65smp
- C コンパイラ: gcc バージョン 2.96 (Address Now モジュールには gcc 4.1 が必要)
- C++ コンパイラ: g++ バージョン 2.96

clientSDK lib は、次のライブラリにリンクされます。

- libstdc++-libc6.2-2.so.3
- libm.so.6
- libc.so.6
- ld-linux.so.2

Red Hat (64 ビット)

- オペレーティング システム: Red Hat Linux バージョン 2.6.9-34.0.2.ELsmp
- C コンパイラ: gcc バージョン 3.4.5
- C++ コンパイラ: g++ バージョン 3.4.5

clientSDK lib は、次のライブラリにリンクされます。

- libstdc++.so.6
- libm.so.6
- libgcc_s.so.1
- libpthread.so.0
- libc.so.6
- ld-linux-x86-64.so.2

SuSE

- オペレーティング システム: SuSE SLES 8 (UnitedLinux 1.0 によって動作) (i586)\nKernel 2.4.21-295-smp (0).
- C コンパイラ: gcc バージョン 3.2.2
- C++ コンパイラ: g++ バージョン 3.2.2

clientSDK lib (32 ビット) は、次のライブラリにリンクされます。

- libstdc++.so.5
- libm.so.6
- libgcc_s.so.1
- libc.so.6
- ld-linux.so.2

Solaris

- オペレーティング システム: Solaris 5.8
- C コンパイラ: cc: Forte Developer 7 C 5.4 2002/03/09
- C++ コンパイラ: CC: Forte Developer 7 C++ 5.4 Patch 111715-16 2005/04/28

clientSDK 32 bit lib は、次のライブラリにリンクされます。

- libpthread.so.1
- libsocket.so.1
- libnsl.so.1
- librt.so.1
- libc.so.1
- libdl.so.1
- libmp.so.2
- libaio.so.1
- libc_psr.so.1

clientSDK 64 bit lib は、次のライブラリにリンクされます。

- libpthread.so.1
- libsocket.so.1
- libnsl.so.1
- librt.so.1
- libc.so.1
- libmp.so.2
- libmd5.so.1
- libscf.so.1
- libaio.so.1
- libdoor.so.1
- libuutil.so.1

- libm.so.2
- libc_psr.so.1
- libmd5_psr.so.1

AIX

- オペレーティング システム: AIX バージョン 5.1.0.0
- C コンパイラ: xlc 6.0 Visual Age C 6.0
- C++ コンパイラ: xlC 6.0 Visual Age C++ 6.0

clientSDK 32 bit および 64 bit lib は、次のライブラリにリンクされます。

- libC.a
- libc_r.a
- libpthread.a
- librtl.a

サードパーティ ライブラリ

Spectrum™ Technology Platform API では、次のサードパーティ ライブラリが使用されます。

- Apache Commons Pool 1.6
- ICU 3.2.0
- Jakarta Commons HttpClient 3.1
- OpenSSL 1.0.2L
- OpenTop 1.5.3
- POCO 1.3

ネットワーク プロトコルとポート

API は、HTTP、HTTPS、またはソケット使用して Spectrum™ Technology Platform サーバーと通信します。Spectrum™ Technology Platform では、通常、ポート 8080 で HTTP 要求を待機し、ポート 443 で HTTPS 要求を待機します。HTTP および HTTPS 機能も C、C++、COM、Java、および .NET の API でサポートされています。.NET、Java、および COM の API は Unicode をサポートし、C と C++ の API は ASCII と Unicode の両方をサポートします。

HTTP のほかに、Spectrum™ Technology Platform は永続的なソケット接続もサポートしています。この高速ソケット接続は、従来の HTTP をはるかにしのぐパフォーマンスを発揮します。Spectrum™ Technology Platform では、通常、ポート 10119 でソケット要求を待機します。

サンプル アプリケーションの使用

クライアント SDK には、サポートされるすべての言語のサンプル アプリケーションが付属しています。このサンプル アプリケーションは、入力データの文字をすべて大文字または小文字に変換するサンプル サービスを Spectrum™ Technology Platform サーバー上で呼び出します。

1. `casing-<version>.car` ファイルを `ClientAPI\common\lib` から `server\app\deploy` サーバーの Spectrum™ Technology Platform フォルダにコピーします。

   これで、サンプル サービスで使用する文字変換サービスが Spectrum™ Technology Platform サーバーに展開されます。

2. `ClientAPI\platforms` フォルダで、使用しているプラットフォーム用の `samples` サブフォルダを見つけ、`readme.txt` ファイルを開いて、サンプル アプリケーションの詳しい使用方法を参照します。

   注： ライセンスを受けているいずれかのサービスを使用するようにサンプル アプリケーションを変更し、サンプルを再コンパイルして実行できます。

HTTPS の使用

この手順では、アプリケーションと Spectrum™ Technology Platform サーバーの間で HTTPS 通信を使用する方法について説明します。

1. アプリケーションと Spectrum™ Technology Platform サーバーの間の通信で使用するルート CA を指定します。そのためには、以下のいずれかの操作を実行します。

オプション	説明
使用するルート CA が不明な場合	`ca-bundle.pem` ファイルをワーキング ディレクトリにコピーします。C/C++、COM、および ASP の場合、.pem ファイルは Client SDK をインストールした場所の次のフォルダにあります。 `Spectrum Client SDK\ClientAPI\platforms\windows\c-c++\<32or64>\<version>\lib\openssl` ASP の場合、ワーキング ディレクトリの例がいくつかあります。 • Internet Information Services を使って ASP を実行する場合は、`ca-bundle.pem` を Windows システム ディレクトリ (例えば `C:\Windows\system32`) にコピーします。 • Internet Explorer を使って ASP を実行する場合は、`ca-bundle.pem` を Internet Explorer のデフォルト ワーキング ディレクトリ (例えば `C:\Documents and Settings\<user>\Desktop`) にコピーします。
使用するルート CA がわかっている場合	CA バンドル ファイルにルート CA 証明書を指定します。

2. アプリケーションで、サーバーに接続するときに、接続タイプ HTTPS に設定します。

タイムアウト値を増やす

クライアントとサーバー間でタイムアウトが発生するときは、クライアントのタイムアウト値を増やします。

- タイムアウト値の設定には `setConnectionProperty` メソッドを使用します。

2 - C API

このセクションの構成

C API の概要	16
Server	32
Service	37
Message	38
DataTable	52
DataRow	60

C API の概要

C API は、次の要素で構成されます。

- Server
- Service
- Message
- DataTable
- DataRow

> **注：** C API は、C++ コードを C でラップしたものです。Unix では C++ コンパイラを使って C アプリケーションをビルドできます。通常はこの方法を使います。ただし、Linux や Solaris では C コンパイラを直接使うこともできます。HP-UX または AIX で C コンパイラを使うには、すべての C++ 必須ライブラリにリンクする必要があります。この操作を行うには、 `ldd ./batch` を `..../samples/batch/bin/` の下で実行してすべての依存ライブラリを取得し、それらを makefile 内のリンク セクションに追加します。

サポートされるライブラリ

Spectrum™ Technology Platform は、ASCII バージョンと Unicode バージョンの C API を提供します。Unicode バージョンでは、元の ASCII バージョンの API 設計との互換性が極力維持されます。 Spectrum™ Technology Platform は、Unicode 機能をサポートするために International Components for Unicode (ICU) が API に適用されます。 ICU は、長年にわたって広く利用されている Unicode サポート用の C/C++ ライブラリであり、IBM で開発されました。

Unicode 規格は、16 ビットのコード単位に基づいてデフォルトのエンコーディングを定義します。 ICU では、UChar を符号なしの 16 ビット整数タイプ (unsigned short *) として定義することで Unicode がサポートされます。 これが、ICU で文字列を表す文字配列の基本型です。 Spectrum™ Technology Platform は、C API で Unicode 文字列を表現するために UChar を使用します。

> **注：** 一部のサービスは、Unicode 文字セットを完全にはサポートしません。 例えば、ValidateAddress サービスは、米国入力/国際入出力用に ISO 8859-1 文字セットをサポートし、カナダ入出力用に CP 850 文字セットをサポートします。 ただし、入力データに ASCII ではない文字が含まれる場合は、基本サービスが Unicode 文字セットを完全にサポートしていなくても Unicode ライブラリが使用されます。

UChar の詳細については、次の 2 つのサイトを参照してください。

- **icu.sourceforge.net/userguide**

- **www.ibm.com/software/globalization/icu**

Windows でサポートされる C ライブラリ

各 API 設定から生成されるライブラリ ファイルの名前は、共通の基本名 (g1client) に固有の接尾文字と、場合によってはさらに接頭文字 (静的ライブラリであれば "lib") が付加されたフォーマットになります。 ライブラリの接尾文字は、次の意味を持ちます。

```
<lib>g1client<S><U><D>.<lib|dll>
```

- lib — 静的ライブラリ
- dll — 動的 (共有) ライブラリ
- S — 単一スレッド ビルド。 この文字がないのは、マルチスレッド ビルドであることを意味します。
- U — UNICODE バージョン ビルド。 この文字がないのは、ASCII ビルドであることを意味します。
- D — デバッグ用ビルド。 この文字がないのは、最適化されたリリース用ビルドであることを意味します。

UNICODE バージョンを有効にするには、LIB_UNICODE マクロ定義がプロジェクトに存在する必要があります。

静的 C/C++ API ライブラリ UNICODE バージョンを使うには、U_STATIC_IMPLEMENTATION をプロジェクトで定義する必要があります。

動的バージョンを使うには、G1CLIENT_DLL をプロジェクトで定義する必要があります。

また、"auto_link.h" というファイルをヘッダー ファイル ディレクトリに配置します。このファイルは、プロジェクト設定に従ってすべての対応するライブラリに自動的にリンクします。

Windows で 64 ビット ライブラリを呼び出すには、VER_64 をプロジェクトで定義する必要があります。

静的ライブラリ

> **注 :** このセクションに記載されている名前は 32 ビット ライブラリ用です。 64 ビット ライブラリ用は、ライブラリ名の "32" を "64" に置き換えてください。

表 1 : 単一スレッド/リリース

	ASCII	Unicode
g1	libg1client_S.lib	libg1client_SU.lib

	ASCII	Unicode
openssl	otlibeay32.lib otlibssl32.lib	otlibeay32.lib otlibssl32.lib
opentop	opentop.lib	opentopw.lib
icu		libicuuc.lib libicudt.lib libicuin.lib libicuio.lib
Poco	PocoXML32.lib	PocoXML32w.lib

表 2 : 単一スレッド/デバッグ

	ASCII	Unicode
g1	libg1client_SD.lib	libg1client_SUD.lib
openssl	otlibeay32d.lib otlibssl32d.lib	otlibeay32d.lib otlibssl32d.lib
opentop	opentopd.lib	opentopwd.lib
icu		libicuucd.lib libicudtd.lib libicuind.lib libicuiod.lib
Poco	PocoXML32d.lib	PocoXML32wd.lib

表 3 : マルチ/リリース (マルチスレッド CRT 使用)

	ASCII	Unicode
g1	libg1client.lib	libg1client_U.lib

	ASCII	Unicode
openssl	otlibeay32mt.lib otlibssl32mt.lib	otlibeay32mt.lib otlibssl32mt.lib
opentop	opentopmt.lib	opentopmtw.lib
icu		libicuucmt.lib libicudtmt.lib libicuinmt.lib libicuiomt.lib
Poco	PocoXMLmt32.lib	PocoXML32mtw.lib

表 4 : マルチ/デバッグ (マルチスレッド CRT 使用)

	ASCII	Unicode
g1	libg1client_D.lib	libg1client_UD.lib
openssl	otlibeay32mtd.lib otlibssl32mtd.lib	otlibeay32mtd.lib otlibssl32mtd.lib
opentop	opentopmtd.lib	opentopmtwd.lib
icu		libicuucmtd.lib libicudtmtd.lib libicuinmtd.lib libicuiomtd.lib
Poco	PocoXMLmt32d.lib	PocoXML32mtwd.lib

動的ライブラリ

注 : このセクションに記載されている名前は 32 ビット ライブラリ用です。 64 ビット ライブラリ用は、ライブラリ名の "32" を "64" に置き換えてください。

表 5 : マルチ/リリース (マルチスレッド CRT 使用)

	ASCII	Unicode
g1	g1client.dll	g1client_U.dll
openssl	otlibeay32mts.dll otlibssl32mts.dll	otlibeay32mts.dll otlibssl32mts.dll
opentop	opentopmts.dll	opentopmtws.dll
icu		icuuc32.dll icuio32.dll icuin32.dll icudt32.dll
Poco	PocoXML32mts.dll	PocoXML32mtws.dll

表 6 : マルチ/デバッグ (マルチスレッド CRT 使用)

	ASCII	Unicode
g1	g1client_D.dll	g1client_UD.dll
openssl	otlibeay32mtds.dll otlibssl32mtds.dll	otlibeay32mtds.dll otlibssl32mtds.dll
opentop	opentopmtds.dll	opentopmtwds.dll
icu		icuuc32d.dll icuio32d.dll icuin32d.dll icudt32d.dll
Poco	PocoXML32mtds.dll	PocoXML32mtwds.dll

Unix でサポートされる C ライブラリ

各 ClientSDK 設定から生成されるライブラリ ファイルの名前は、共通の基本名 (libg1client) に固有の接尾文字が付加されたフォーマットになります。 Spectrum™ Technology Platform は、ASCII バージョンと UNICODE バージョンのマルチスレッド/リリース ビルドを提供します。

ライブラリの接尾文字は、次の意味を持ちます。

```
libg1client<U>.<so|sl|a>
```

ここで U は、UNICODE バージョン ビルドのことです。 この文字がないのは、ASCII ビルドであることを意味します。

UNICODE バージョンを使うには、LIB_UNICODE をプロジェクトで定義する必要があります。

UNICODE バージョンの C++ API では、すべてのクラスのネームスペースが g1client になります。

表 7 : AIX

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libssl.so	libcrypto.so libssl.so
opentop	libopentop-xlCmt.so	libopentop-xlCmtw.so libotxml-xlCmtw.so
icu		libicudata34.a libicui18n34.a libicuio34.a libicuuc34.a
Poco	libPocoXML.so	

表 8 : HP-UX RISC

	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl

	ASCII	Unicode
openssl	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicui18n.sl libicui18n.sl.34 libicuio.sl libicuio.sl.34 libicuuc.sl libicuuc.sl.34
Poco	libPocoXML.sl	

表 9 : HP-UX Itanium

	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.a libssl.a	libcrypto.a libssl.a
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicudata.sl.34.0 libicui18n.sl libicui18n.sl.34 libicui18n.sl.34.0 libicuio.sl libicuio.sl.34 libicuio.sl.34.0 libicuuc.sl libicuuc.sl.34 libicuuc.sl.34.0
Poco	libPocoXML.sl	

表 10 : Linux

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-gccmt.so	libopentop-gccmtw.so libotxml-gccmtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	

表 11 : Solaris SPARC

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-fortemt.so	libopentop-fortemtw.so libotxml-fortemtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34

	ASCII	Unicode
Poco	libPocoXML.so	

定数

C API では、2 組の定数が使用されます。最初の 1 組は `Server` コンポーネント用です。以下の表に説明します。

表 12 : `Server` コンポーネントの定数

定数名	説明/デフォルト	例
SERVER_HOST	サーバーのホスト名を表す文字列。デフォルトは "localhost" です。	65.89.200.89
SERVER_PORT	サーバーのポートを表す文字列。デフォルトは "8080" です。	10119
SERVER_ACCOUNT_ID	サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1
SERVER_ACCOUNT_PASSWORD	サーバーのアカウントパスワードを表す文字列。デフォルト値はありません。	user1
SERVER_CONNECTION_TIMEOUT	サーバーの接続タイムアウトをミリ秒単位で表す文字列。デフォルトは "5000" です。	50000

定数名	説明/デフォルト	例
SERVER_CONNECTION_TYPE	サーバーの接続タイプを表す文字列。現在は HTTP、HTTPS、または SOCKET のみがサポートされています。デフォルトは "HTTP" です。	HTTP(S)
SERVER_PROXY_HOST	プロキシ サーバーのホスト名を表す文字列。デフォルト値はありません。	192.168.1.77
SERVER_PROXY_PORT	プロキシ サーバーのポートを表す文字列。デフォルト値はありません。	8080
SERVER_PROXY_USER	プロキシ サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1
SERVER_PROXY_PASSWORD	プロキシ サーバーのアカウント パスワードを表す文字列。デフォルト値はありません。	user1

2 組目の定数は `Message` コンポーネント用です。

表 13 : `Message` コンポーネントの定数

定数名	説明	例
MESSAGE_CONTEXT_ACCOUNT_ID	メッセージ コンテキストのアカウント ID を表す文字列。	user1
MESSAGE_CONTEXT_ACCOUNT_PASSWORD	メッセージ コンテキストのアカウント パスワードを表す文字列。	user1

定数名	説明	例
MESSAGE_CONTEXT_SERVICE_NAME	メッセージコンテキストのサービス名を表す文字列。	echoservice

エラー メッセージ

成功時に SUCCESSFUL_RETURN または 0 (ゼロ) の値を返す関数もあります。成功しなかった場合、関数はエラー コードを返します。エラー メッセージを取得するには、getErrorMessage(int errorCode) を呼び出します。例:

```
Server *server = NULL;
int nRet;
//Create Server
server = createServer();
//set the property to the server
...
//Connect to server
printf("Making connection to the server...\n");
nRet = serverConnect(server);
if(nRet != SUCCESSFUL_RETURN)
{
// ASCII Version-use the following code
printf(getErrorMessage(nRet));
//Unicode Version -use the following code
UChar * error = getErrorMessage(nRet));
// more code to print out the error messageÂ….
 return ;
 }
```

C API では、次のエラー メッセージが使用されます。

- NULL 構造体渡しのエラー メッセージ:
  - "Input null DataRow"
  - "Input null DataTable"
  - "Input null Message"
  - "Input null Server"
- 接続エラー メッセージ:
  - "Connection type not supported"
  - "Client timeout"
  - "Blank connection property name"

  - "Blank property name"
- DataTable 作成時のエラー メッセージ:
  - "Blank column name"
  - "Duplicated column name"
- MessagePackaging 例外のエラー メッセージ:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect from Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

サンプル アプリケーション

以下のサンプル コードに、ASCII バージョンの C API の使い方を示します。

```
// Declarations
Server *server = NULL;
Message *request = NULL;
DataTable *dataTable = NULL;
Message *reply = NULL;
Service *service = NULL;
int nRet;
DataRow *row1 = NULL;
DataRow *row2 = NULL;
DataTable *returnDataTable= NULL;
char** columnNames;
DataRow** rows;
DataRow*dataRow;
int i;
int j;
char* value;

//Create Server
server = createServer();

//Set server connection properties
nRet = setConnectionProperty(server, SERVER_HOST, "localhost");
nRet = setConnectionProperty(server, SERVER_PORT, "10119 ");
nRet = setConnectionProperty(server, SERVER_CONNECTION_TYPE,"SOCKET");

nRet = setConnectionProperty(server, SERVER_ACCOUNT_ID,"guest");
nRet = setConnectionProperty(server, SERVER_ACCOUNT_PASSWORD,"");
```

```
//Connect to server
nRet = serverConnect(server);
if(nRet != SUCCESSFUL_RETURN)
{
printf( getErrorMessage(nRet));
// free memory
if(server)
nRet = deleteServer(server);
return ;
}

 //Get Service From Server
 service = getServiceFromServer(server,"ValidateAddress" );

//Create Input Message
request = createMessage();

//Fill DataTable in the input message
dataTable = getDataTable(request);
nRet= addColumn( dataTable, "AddressLine1", &nRet);
nRet= addColumn( dataTable, "City", &nRet);
nRet= addColumn( dataTable, "StateProvince", &nRet);

row1 = newRow( dataTable );
setByIndex (row1, 0 , "4200 Parliament Place");
setByIndex (row1, 1 ,"Lanham");
setByIndex (row1, 2 , "Maryland");

addRow( dataTable, row1);

row2 = newRow( dataTable );
setByIndex (row2, 0 , "10535 Boyer Blvd");
setByIndex (row2, 1 ,"Austin");
setByIndex (row2, 2 , "Texas");

addRow( dataTable, row2);

//Set"option" Properties to the Input Message
nRet = putOption(request, "OutputCasing","M");
nRet = putOption(request, "OutputRecordType","A");

//Process Input Message, return output Message
nRet = processMessage(service, request, &reply);
if(nRet != SUCCESSFUL_RETURN)
{
printf("Error Occurred, " );
printf(getErrorMessage(nRet));

// free memory
if(request)
nRet = deleteMessage(request);
if(reply)
```

```
nRet = deleteMessage(reply);
if(server)
nRet = deleteServer(server);

return ;

}

//Disconnect from server
nRet = serverDisconnect(server);

//Get the result from the response message
returnDataTable = getDataTable(reply );
columnNames = getColumnNames(returnDataTable);

rows = getDataRows( returnDataTable);

for( i=0; i < getRowCount( returnDataTable); i++)
{
dataRow = rows[i];

for(j=0; j < getColumnCount(returnDataTable); j++)
{
value = (char*)getByIndex( dataRow, j);
printf(value);
printf("\n");
}
}

//Free Memory
if(request)
nRet = deleteMessage(request);
if(reply)
nRet = deleteMessage(reply);
if(server)
nRet = deleteServer(server);
}
```

以下のサンプル コードに、Unicode バージョンの C API の使い方を示します。 ここでは文字列を表すのに UChar*(または unsigned short*) を使います。これは Unicode 文字列を表す 16 ビット タイプです。ICU には、8 ビット文字列を 16 ビット文字列に変換する、u_charsToUChars という関数が用意されています。 この例は、Unicode バージョンの C API を呼び出す方法を示しています。 入力文字列はすべて ASCII なので、u_charsToUChars を使って 16 ビット文字列に変換します。 また、Unicode 文字列を C API に直接渡すこともできます。

```
UChar* convertcharToUChar( char* name, UChar* value)
  {
   int lenName= strlen(name);
   u_charsToUChars(name, value, lenName );
   value[ lenName]=0;
   return value;
```

```
  }

   // Declarations
   Server *server = NULL;
   Message *request = NULL;
   DataTable *dataTable = NULL;
   DataTable *returnDataTable= NULL;
   Message *reply = NULL;
   Service *service = NULL;
   int nRet;
   DataRow* newDataRow;
   UChar    name[128];
   UChar value[128];
   UChar** columnNames;
   DataRow** rows;
   DataRow* dataRow;
   int i, j;
   UChar* columnValue;
   UChar*   errorMsg;

   //Create Server
   server = createServer();

   //Set server connection properties
   setConnectionProperty(server, convertcharToUChar( SERVER_HOST, name)
 , convertcharToUChar( "localhost", value));
   setConnectionProperty(server, convertcharToUChar( SERVER_PORT, name)
 , convertcharToUChar( "10119", value));
   setConnectionProperty(server, convertcharToUChar(
SERVER_CONNECTION_TYPE, name) , convertcharToUChar( "SOCKET", value));

  setConnectionProperty(server, convertcharToUChar( SERVER_ACCOUNT_ID,
 name) , convertcharToUChar( "guest", value));
   setConnectionProperty(server, convertcharToUChar(
SERVER_ACCOUNT_PASSWORD, name) , convertcharToUChar( "", value));


   //Connect to server
   nRet = serverConnect(server);
   if(nRet != SUCCESSFUL_RETURN)
   {
   // error handling
   errorMsg = getErrorMessage(nRet);
   // free memory
   if(server)
   nRet = deleteServer(server);
   return ;
   }

   //Get Service From Server
   service = getServiceFromServer(server,convertcharToUChar(
"ValidateAddress", name));
```

```
   //Create Input Message
   request = createMessage();

   //Fill DataTable in the input message
   dataTable = getDataTable(request);
   addColumn( dataTable,convertcharToUChar( "AddressLine1", name),
&nRet);
   addColumn( dataTable,convertcharToUChar( "City", name),  &nRet);
  addColumn( dataTable,convertcharToUChar( "PostalCode", name),  &nRet);

   addColumn( dataTable,convertcharToUChar( "StateProvince", name),
&nRet);

   newDataRow = newRow( dataTable );

   setByIndex (newDataRow, 0 , convertcharToUChar( "74, Rue Octave
Bénard", name) );
   setByIndex (newDataRow, 1 , convertcharToUChar( "Etang-Salé-les-
Bains", name) );
   setByIndex (newDataRow, 2 , convertcharToUChar( "97427", name) );
   setByIndex (newDataRow, 3 , convertcharToUChar( "Reunion Island",
name) );

   addRow( dataTable, newDataRow);

   //Set"option" Properties to the Input Message
  nRet = putOption(request, convertcharToUChar( "OutputCasing", name),
 convertcharToUChar( "M", value));
   nRet = putOption(request, convertcharToUChar( "OutputRecordType",
name), convertcharToUChar( "A", value));


   //Process Input Message, return output Message
   nRet = processMessage(service, request, &reply);
    if(nRet != SUCCESSFUL_RETURN)
   {
   // error handling
   errorMsg = getErrorMessage(nRet);
   // free memory
   if(request)
   nRet = deleteMessage(request);
   if(reply)
   nRet = deleteMessage(reply);
   if(server)
   nRet = deleteServer(server);

   return ;
   }

   //Disconnect from server
   nRet = serverDisconnect(server);

   //Get the result from the response message
```

```
returnDataTable = getDataTable(reply );
columnNames = getColumnNames(returnDataTable);
rows = getDataRows( dataTable);
for( i=0; i < getRowCount( dataTable); i++)
{
dataRow =  rows[i];
for(j=0; j < getColumnCount(dataTable); j++)
{
columnValue = (UChar*)getByIndex( dataRow, j);
}
}

//Free Memory
if(request)
nRet = deleteMessage(request);
if(reply)
nRet = deleteMessage(reply);
if(server)
nRet = deleteServer(server);
```

Server

Server構造体は、サーバーへの接続、サーバーからの切断、およびサーバーからのサービスの取得に使用されます。

CreateServer

サーバーを作成します。

構文

```
Server* createServer()
```

パラメータ

なし

結果

サーバーが作成されます。

例

```
Server *server = NULL;
//Create Server
server = createServer();
```

DeleteServer

サーバーを削除します。

構文

```
int deleteServer(Server* server)
```

パラメータ

- Server — 削除するサーバー。

結果

0 (正常終了) またはエラー コードを返します。

例

```
int nRet;
nRet = deleteServer(server);
```

SetConnectionProperty

ホスト名、タイアウト時間など、サーバー接続設定プロパティを設定します。

構文

ASCII バージョン

```
int setConnectionProperty(Server* server, const char* name, const char*
 value)
```

Unicode バージョン

```
int setConnectionProperty(Server* server, const UChar* name, const UChar*
 value)
```

パラメータ

- Server — クライアントが接続するサーバー。
- Name — 接続プロパティの名前。HOST など。
- Value — 接続プロパティの値。"www.myhost.com" など。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
int nRet;
Server *server = NULL;
nRet = createServer(&server);
nRet = setConnectionProperty(server, SERVER_HOST,
"localhost");
```

Unicode バージョン

```
int nRet;
// construct 16-bit string
UChar serverHost[32];
char* SERVER_HOST= SERVER_HOST;
u_charsToUChars(SERVER_HOST, serverHost, strlen(SERVER_HOST));
serverHost [ strlen(SERVER_HOST)]=0;
// construct 16-bit string
UChar hostValue [32];
char* value= "localhost";
u_charsToUChars(value, hostValue, strlen(value));
hostValue[ strlen(value)]=0;
nRet = setConnectionProperty(server, serverHost , hostValue);
```

ServerConnect

プロパティを読み取って、設定値を決定し、サーバーへの接続を確立します。

> **注：** C では、HTTP、HTTPS、またはソケット サーバー接続プロトコルを使用します。HTTP と HTTPS は、クライアント接続を論理的に確立するだけで、GetService メソッドまたは Process メソッドが呼び出されるまで実際にはサーバーに接続しません。ソケット プロトコルは、`Connect` が呼び出された時点でサーバーへの接続を確立します。

構文

```
int serverConnect(Server* server)
```

パラメータ

- Server — クライアントが接続するサーバー。

結果

0 (正常終了) またはエラー コードを返します。

例

```
int nRet;
nRet = serverConnect(server);
```

ServerDisconnect

サーバーから切断します。

構文

```
int serverDisconnect(Server* server)
```

パラメータ

- Server — クライアントが切断するサーバー。

結果

0 (正常終了) またはエラー コードを返します。

例

```
int nRet;
nRet = serverDisconnect(server);
```

GetServiceFromServer

サーバーからサービスを取得します。

構文

ASCII バージョン

```
Service* getServiceFromServer(Server* server,const char* serviceName )
```

Unicode バージョン

```
Service* getServiceFromServer(Server* server,const UChar* serviceName
)
```

パラメータ

- Server - クライアントが接続するサーバー。
- ServiceName - クライアントが要求するサービスの名前。

結果

サービスが返されます。

例

ASCII バージョン

```
Server *server= NULL;
Service *service = NULL;
//Create Server
server = createServer();
...
// get Service From Server
service = getServiceFromServer(server,"ValidateAddress" );
```

Unicode バージョン

```
// construct 16-bit string
UChar serviceName[32];
char* sName="ValidateAddress";
u_charsToUChars(sName, serviceName, strlen(sName));
serviceName [ strlen(sName)]=0;
service = getServiceFromServer(server , serviceName );
```

Service

`Service` 構造体は、メッセージを処理するために使用されます (より具体的に言えば、メッセージをサーバーに送信し、サーバーから応答を受信するために使用されます)。

ProcessMessage

入力メッセージを処理し、サーバーから応答メッセージを取得します。

> **注：** 返されたメッセージが不要になった時点で、`DeleteMessage()` を呼び出してメモリを解放する必要があります。

構文

```
int processMessage (Service* service, Message* request, Message*
returnVal)
```

パラメータ

- Service — クライアントが要求するサービス。
- Request — "option" 設定とデータセットが含まれている入力メッセージ。
- returnVal — サーバーからの応答メッセージ。

結果

0 (正常終了) またはエラー コードを返します。

例

```
Message *request = NULL;
Message *reply = NULL;
int nRet;
 ...
 // Assume that service is given here
 // Create Input Message
 request = createMessage();
 ... more code to fill dataTable information in request message
//Process Input Message, return output Message
 nRet = processMessage(service, request, &reply);
 if(nRet != SUCCESSFUL_RETURN)
 {
```

```
printf("Error Occurred, " );
printf(getErrorMessage(nRet));
return ;
}
if(request)
nRet = deleteMessage(request);
if(reply)
nRet = deleteMessage(reply);
```

Message

Message 構造体は、入力データを送信し、サービスから出力データを受け取ります。Message のプロパティには、コンテキスト プロパティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、オプション プロパティ (サービス固有の実行時オプション)、エラー プロパティ (エラー クラス、エラー メッセージ、エラー スタックトレース) などがあります。

CreateMessage

メッセージを作成します。

構文

```
Message* createMessage()
```

パラメータ

なし

結果

メッセージが作成されます。

例

```
Message* request = NULL;
request = createMessage();
```

DeleteMessage

メッセージを削除します。

構文

```
int deleteMessage(Message* message)
```

パラメータ

- Message— 削除するメッセージ。

結果

0 (正常終了) またはエラー コードを返します。

例

```
int nRet = deleteMessage(message);
```

GetContext

メッセージのコンテキスト セッションに指定された名前のコンテキスト エンティティから値を取得します。"コンテキスト" エンティティには、アカウント ID、アカウント パスワード、サービス名、サービス メソッドなどがあります。

構文

ASCII バージョン

```
const char* getContext(Message* message,const char* name)
```

Unicode バージョン

```
const UChar * getContext(Message* message,const UChar* name)
```

パラメータ

- Message - この関数で操作するメッセージ。
- Name - 関連付けられた値を返す名前。

結果

コンテキスト エンティティにある名前の値を返します。名前が存在しない場合、空白の文字列を返します。

例

ASCII バージョン

```
const char* value = getContext(message, "account.id");
```

Unicode バージョン

```
UChar* value;
// construct 16-bit string
UChar accountID[32];
char* account="account.id";
u_charsToUChars(account, accountID, strlen(account));
accountID[ strlen(account)]=0;
value = getContext(message, accountID);
```

GetContextMap

すべてのコンテキスト エントリが含まれるマップを取得します。

構文

ASCII バージョン

```
MAP_STRING**getContextMap(Message* message)
Where the MAP_STRING is defined by
typedef struct map_string{
char* key;
char* value;
}MAP_STRING;
```

Unicode バージョン

```
MAP_STRING**getContextMap(Message* message)
Where the MAP_STRING is defined by
typedef struct map_string{
UChar* key;
```

```
UChar* value;
}MAP_STRING;
```

パラメータ

- Message - この関数で操作するメッセージ。

結果

すべてのコンテキスト エントリが含まれる MAP_STRING の配列を返します。

例

ASCII バージョン

```
int i;
char* name;
char* value;
MAP_STRING** mapping;
mapping = getContextMap( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

Unicode バージョン

```
int i;
UChar* name;
UChar* value;
MAP_STRING** mapping;
mapping = getContextMap( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

PutContext

指定された名前に基づいてコンテキスト プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

ASCII バージョン

```
int putContext(Message* message,const char* name,
const char* value)
```

Unicode バージョン

```
int putContext(Message* message,const UChar* name,
const UChar* value)
```

パラメータ

- Message — この関数で操作するメッセージ。
- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
int nRet;
Message* message = createMessage();
nRet = putContext( message, "account.id", "user1") ;
```

Unicode バージョン

```
int nRet;
Message* message;
// construct 16-bit string
UChar accountID[32];
char* account="account.id";
UChar accountIDValue[32];
char* accountValue="user1";
u_charsToUChars(account, accountID, strlen(account));
```

```
accountID [ strlen(account)]=0;
u_charsToUChars(accountValue, accountIDValue, strlen(accountValue));
accountIDValue [ strlen(accountValue)]=0;
message = createMessage();
nRet = putContext( message, accountID, accountIDValue);
```

PutContextMap

新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。

構文

```
int putContextMap(Message* message, MAP_STRING** context)
```

パラメータ

- Message - この関数で操作するメッセージ。
- 現在のコンテキスト マップに追加する新しいコンテキスト マップ。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
MAP_STRING** mapping;
Message* message;
message = createMessage();
int nRet;
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet = putContextMap( message, mapping) ;
```

Unicode バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
```

```
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";

u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;

message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet = putContextMap( message, mapping) ;
```

SetContextMap

新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。

構文

```
int setContextMap(Message* message, MAP_STRING** context)
```

パラメータ

- Message - この関数で操作するメッセージ。
- 現在のコンテキスト マップを置き換えるために使用する新しいコンテキスト マップ。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
```

```
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet=setContextMap( message, mapping) ;
```

Unicode バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1 ;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet=setContextMap( message, mapping) ;
```

GetOption

メッセージのオプションセッションに指定された名前のオプションエンティティから値を取得します。"オプション" エンティティには、出力の大文字と小文字の区別、出力データのフォーマットなど、サービス固有の実行時オプションが含まれます。

構文

ASCII バージョン

```
const char* getOption(Message* message,const char* name)
```

Unicode バージョン

```
const Uchar* getOption(Message* message,const UChar* name)
```

パラメータ

- Message - この関数で操作するメッセージ。

- Name - 関連付けられた値を返す名前。

結果

メッセージの "オプション" プロパティ内の名前の値を返します。または、その名前が存在しない場合は空の文字列を返します。

例

ASCII バージョン

```
const char* value = getOption (message, " OutputCasing");
```

Unicode バージョン

```
UChar* value;
// construct 16-bit string
UChar option[32];
char* optionValue="OutputCasing";
u_charsToUChars(optionValue, option, strlen(optionValue));
option [ strlen(optionValue)]=0;
value = getOption(message, option);
```

GetOptions

すべてのオプション エントリが含まれるマップを取得します。

構文

```
MAP_STRING**        getOptions(Message* message)
```

パラメータ

- Message — この関数が適用されるメッセージ。

結果

すべてのコンテキスト エントリが含まれる MAP_STRING の配列を返します。

例

ASCII バージョン

```
int i;
char* name;
char* value;
```

```
MAP_STRING** mapping;
mapping = getOptions( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

Unicode バージョン

```
int i;
UChar* name;
UChar* value;
MAP_STRING** mapping;
mapping = getOptions( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

PutOption

指定された名前に基づいてオプション プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。オプション プロパティはサービス固有の実行時オプションです。

構文

ASCII バージョン

```
int putOption(Message* message,const char* name,
const char* value)
```

Unicode バージョン

```
int putOption(Message* message,const UChar* name,
const UChar* value)
```

パラメータ

- Message - この関数で操作するメッセージ。
- Name - 指定された値を関連付ける名前。

- Value - 指定された名前に関連付ける値。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
int nRet;
Message* message = createMessage();
nRet = putOption( message, "OutputCasing", "M");
```

Unicode バージョン

```
int nRet;
Message* message;
// construct 16-bit string
UChar option[32];
char* optionString="OutputCasing";

UChar optionValue[32];
char* optionValueString="M";

u_charsToUChars(optionString, option, strlen(optionString));
option[ strlen(optionString)]=0;

u_charsToUChars(optionValueString, optionValue,
strlen(optionValueString));
optionValue [ strlen(optionValueString)]=0;

message = createMessage();
nRet = putOption( message, option, optionValue);
```

PutOptions

新しいオプション プロパティを現在のオプション プロパティに追加します。

構文

```
int putOptions(Message* message, MAP_STRING** context)
```

パラメータ

- Message - この関数で操作するメッセージ。
- 現在のオプション プロパティに追加する新しいオプション マップ。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
MAP_STRING** mapping;
Message* message;
message = createMessage();
int nRet;
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet = putOptions( message, mapping) ;
```

Unicode バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet = putOptions ( message, mapping) ;
```

SetOptions

新しいオプション プロパティで現在のオプション プロパティを上書きします。

構文

```
int setOptions(Message* message, MAP_STRING** context)
```

パラメータ

- Message - この関数で操作するメッセージ。
- 現在のオプション マップを置き換えるために使用する新しいオプション マップ。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
message = createMessage();
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet=setOptions( message, mapping) ;
```

Unicode バージョン

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1 ;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet= setOptions ( message, mapping) ;
```

GetError

メッセージからエラー メッセージを取得します。

構文

ASCII バージョン

```
const char* getError(Message* message )
```

Unicode バージョン

```
const UChar* getError(Message* message )
```

パラメータ

- Message - この関数で操作するメッセージ。

結果

メッセージからエラー メッセージを取得して返します。

例

ASCII バージョン

```
const char* error = getError(message );
```

Unicode バージョン

```
const UChar* error = getError(message );
```

GetDataTable

DataTable をメッセージから取得します。

構文

```
DataTable* getDataTable(Message* message )
```

パラメータ

- Message - この関数で操作するメッセージ。

例

```
// Assume that message is given here
DataTable  *dataTable   ;
dataTable = getDataTable( message );
```

DataTable

DataTable には入出力データのレコードが含まれます。

CreateDataTable

DataTable を作成します。

構文

```
DataTable* createDataTable()
```

結果

作成した DataTable を返します。

例

```
DataTable* dataTable;
dataTable = createDataTable();
```

DeleteDataTable

DataTable を削除します。

構文

```
int deleteDataTable(DataTable* dataTable)
```

パラメータ

- Datatable - 削除する DataTable。

例

```
DataTable* dataTable;
dataTable = createDataTable();
...
if(dataTable) deleteDataTable(dataTable);
```

AddColumn

新しい列を追加します。

構文

ASCII バージョン

```
int addColumn(DataTable* dataTable, const char* columnName,
int* indexReturn)
```

Unicode バージョン

```
int addColumn(DataTable* dataTable, const UChar* columnName,
int* indexReturn)
```

パラメータ

- Datatable - この関数で操作する DataTable。
- 列名を DataTable に追加します。
- 対応するインデックスを返します。

結果

0 (正常終了) またはエラー コードを返します。

例外

- 列名が空白です。
- 同名の列がすでにあります。

例

ASCII バージョン

```
int nIndex;
int nRet;
nRet= addColumn( dataTable, "AddressLine1", &nIndex);
```

```
nRet= addColumn( dataTable, "City",&nIndex);
nRet= addColumn( dataTable, "State", &nIndex);
if(nRet != SUCCESSFUL_RETURN)
{
printf(getErrorMessage(nRet));
return ;
}
```

Unicode バージョン

```
int nRet;
int nIndex;
UChar* error;
UChar city[64];
char* cityString= "City"
u_charsToUChars(cityString, city, strlen(cityString));
city[ strlen(cityString)]=0;

nRet= addColumn( dataTable, city,&nIndex);
if(nRet != SUCCESSFUL_RETURN)
{
error = getErrorMessage(nRet);
//more code
}
```

GetColumnNames

すべての列名を取得します。

構文

ASCII バージョン

```
char** getColumnNames(dataTable )
```

Unicode バージョン

```
UChar** getColumnNames(dataTable )
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

列名の配列を返します。

例

ASCII バージョン

```
char* value;
char** columnNames;
int i;
columnNames =getColumnNames ( dataTable) ;
for( i=0; i < getColumncount( dataTable); i++)
{
value = columnNames[i];
}
```

Unicode バージョン

```
UChar* value;
UChar** columnNames;
int i;
columnNames =getColumnNames ( dataTable) ;
for( i=0; i < getColumncount( dataTable); i++)
{
value = columnNames[i];
}
```

GetColumnIndex

対応する列インデックスを取得します。

構文

ASCII バージョン

```
int getColumnIndex(DataTable* dataTable ,const char* columnName)
```

Unicode バージョン

```
int getColumnIndex(DataTable* dataTable ,const UChar* columnName)
```

パラメータ

- Datatable - この関数で操作する DataTable。
- 列名

結果

対応する列インデックスを返します。

例

ASCII バージョン

```
int nIndex ;
nIndex = getColumnIndex(dataTable ,"AddressLine1")
```

Unicode バージョン

```
int nIndex ;
UChar columnName[64];
char* columnNameStr= "AddressLine1" u_charsToUChars(columnNameStr,
columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
nIndex = getColumnIndex(dataTable , columnName);
```

GetColumnCount

列の数を取得します。

構文

```
int getColumnCount(DataTable* dataTable )
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

列の数を返します。

例

```
// Assume that dataTable is given here int nColumnCount ;
nColumnCount = getColumnCount( dataTable ) ;
```

Clear

DataTable 内のデータを消去します。

構文

```
int clear(DataTable* dataTable)
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

0 (正常終了) またはエラー コードを返します。

例

```
// Assume that dataTable is given here
clear(dataTable);
```

GetDataRows

DataTable 内のすべての DataRow の配列を取得します。

構文

```
DataRow** getDataRows(DataTable* dataTable)
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

DataRow の配列を返します。

例

```
// Assume that dataTable is given here
DataRows** rows;
DataRow* dataRow;
int i;
int j;
rows = getDataRows( dataTable);
for( i=0; i < getRowCount( dataTable); i++)
{
 dataRow =  rows[i];
```

```
 for(j=0; j < getColumnCount(dataTable); j++)
 {
 value = (char*)getByIndex( dataRow, j);
 }
}
```

AddRow

DataRow を DataTable に追加します。

構文

```
int addRow(DataTable* dataTable, DataRow* dataRow)
```

パラメータ

- Datatable - この関数で操作する DataTable。
- DataTable に追加する DataRow。

結果

0 (正常終了) またはエラー コードを返します。

例

```
// Assume that dataTable is given here DataRow* newDataRow;
int nRet;
newDataRow = newRow( dataTable );
setByIndex (newDataRow, 0 , "10535 Boyer Blvd");
setByIndex (newDataRow, 1 , "Austin");
setByIndex (newDataRow, 2 , "Texas");
nRet = addRow( dataTable, newDataRow);
```

NewRow

新しい DataRow を DataTable 内に作成します。

構文

```
DataRow* newRow(DataTable* dataTable )
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

作成された新しい DataRow を返します。

例

```
// Assume that dataTable is given here
DataRow* newDataRow;
int nRet;
newDataRow = newRow( dataTable );
setByIndex (newDataRow, 0 , "10535 Boyer Blvd");
setByIndex (newDataRow, 1 , "Austin");
setByIndex (newDataRow, 2 , "Texas");
nRet = addRow( dataTable, newDataRow);
```

GetRowCount

この DataTable にある DataRow の数を取得します。

構文

```
int getRowCount(DataTable* dataTable)
```

パラメータ

- Datatable - この関数で操作する DataTable。

結果

この DataTable にある DataRow の数を返します。

例

```
// Assume that dataTable is given here int nRowCount ;
nRowCount = getRowCount( dataTable);
```

MergeDataTable

指定された DataTable と現在の DataTable を結合します。

構文

```
int mergeDataTable(DataTable* dataTable ,DataTable* other )
```

パラメータ

- Datatable - この関数で操作する DataTable。
- 現在の DataTable と結合する他の DataTable。

結果

0 (正常終了) またはエラー コードを返します。

例

```
// Assume that dataTable and otherDataTable are given here
mergeDataTable (dataTable ,otherDataTableDataRow)
```

DataRow

DataRow には入出力データのレコードが含まれます。

CreateDataRow

DataRow を作成します。

構文

```
DataRow*   createDataRow()
```

結果

作成した DataRow を返します。

例

```
DataRow* dataRow;
dataRow = createDataRow();
```

DeleteDataRow

DataRow を削除します。

構文

```
int   deleteDataRow(DataRow* dataRow)
```

パラメータ

- DataRow が削除されます。

例

```
DataRow* dataRow;
dataRow = createDataRow();
...
if(dataRow)
  deleteDataRow (dataRow);
```

GetColumnNamesFromRow

すべての列名を取得します。

構文

ASCII バージョン

```
char** getColumnNamesFromRow(DataRow* dataRow)
```

Unicode バージョン

```
UChar** getColumnNamesFromRow(DataRow* dataRow)
```

パラメータ

- DataRow - この関数で操作する DataRow。

結果

列名の配列を返します。

例

ASCII バージョン

```
char* value;
char** columnNames;
int i;
columnNames = getColumnNamesFromRow (dataRow) ;
for( i=0; i < getColumnCountFromRow (dataRow); i++)
{
 value = columnNames[i];
}
```

Unicode バージョン

```
UChar* value;
UChar** columnNames;
int i;
columnNames = getColumnNamesFromRow (dataRow) ;
for( i=0; i < getColumnCountFromRow (dataRow); i++)
{
 value = columnNames[i];
}
```

GetColumnIndexFromRow

対応する列インデックスを取得します。

構文

ASCII バージョン

```
int getColumnIndexFromRow(DataRow* dataRow, const char* name)
```

Unicode バージョン

```
int getColumnIndexFromRow(DataRow* dataRow, const UChar* name)
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 列名

結果

対応する列インデックスを返します。

例

ASCII バージョン

```
int nIndex
nIndex = getColumnIndexFromRow ("AddressLine1");
```

Unicode バージョン

```
int nIndex
UChar columnName[64];
char* columnNameStr= "AddressLine1"
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
nIndex = getColumnIndexFromRow (columnName);
```

GetColumnCountFromRow

列の数を取得します。

構文

```
int  getColumnCountFromRow(DataRow* dataRow )
```

パラメータ

- DataRow - この関数で操作する DataRow。

結果

列の数を返します。

例

```
//Assume that the dataRow is given here
int nColumnCount ;
nColumnCount = getColumnCountFromRow (dataRow );
```

GetByIndex

この DataRow の列インデックスによってフィールド配列から値を取得します。

構文

ASCII バージョン

```
const char* getByIndex(DataRow* dataRow, int index)
```

Unicode バージョン

```
const UChar* getByIndex(DataRow* dataRow, int index)
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 指定された値を関連付けるインデックス。

結果

DataRow の列インデックスの値を返します。インデックスが無効の場合は空の文字列を返します。

例

ASCII バージョン

```
char* value = getByIndex( dataRow, 0);
```

Unicode バージョン

```
UChar* value = getByIndex( dataRow, 0);
```

GetByName

この DataRow の列名によってフィールド配列から値を取得します。

構文

ASCII バージョン

```
const char* getByName(DataRow* dataRow, const char* name )
```

Unicode バージョン

```
const UChar* getByName(DataRow* dataRow, const UChar* name )
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 指定された値を関連付ける名前。

結果

DataRow の列名の値を返します。列名が存在しない場合は空の文字列を返します。

例

ASCII バージョン

```
char* value = getByName ( dataRow, "City")
```

Unicode バージョン

```
UChar* value;
UChar columnName[64];
char* columnNameStr= "City"
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
value = getByName ( dataRow, columnName);
```

MergeDataRow

指定された DataRow と現在の DataRow を結合します。

構文

```
int   mergeDataRow(DataRow* dataRow, DataRow* other)
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 現在の DataRow と結合する他の DataRow。

結果

0 (正常終了) またはエラー コードを返します。

例

```
//Assume that the dataRow and otherDataRow are given here
int nRet;
nRet= mergeDataRow(dataRow, otherDataRow);
```

SetByName

DataRowの対応する列の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。

構文

ASCII バージョン

```
int setByName(DataRow* dataRow, const char* name, const char* value)
```

Unicode バージョン

```
int setByName(DataRow* dataRow, const UChar* name, const
UChar* value)
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 指定された値を関連付ける名前。
- 指定された名前に関連付ける値。

例外

空白の列名または重複する列名を入力した場合は、エラーを返します。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
int nRet;
nRet= setByName (dataRow, "City", "Austin");
if(nRet != SUCCESSFUL_RETURN)
{ printf(getErrorMessage(nRet));
```

```
//more code
}
```

Unicode バージョン

```
int nRet;
UChar* error;
UChar columnName[64];
char* columnNameStr= "City"
UChar columnValue[64];
char* columnValueStr= "Austin";
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
u_charsToUChars(columnValueStr, columnValue, strlen(columnValueStr));
columnValue [strlen(columnValueStr)]=0;
nRet= setByName (dataRow, columnName, columnValue);
if(nRet != SUCCESSFUL_RETURN)
{ error = getErrorMessage(nRet);
//more code
}
```

SetByIndex

DataRow の対応する列の値を設定します。 この名前の値が存在する場合は、古い値を置き換えます。

構文

ASCII バージョン

```
int setByIndex(DataRow* dataRow, int index, const char* value)
```

Unicode バージョン

```
int setByIndex(DataRow* dataRow, int index, const UChar* value)
```

パラメータ

- DataRow - この関数で操作する DataRow。
- 指定された値を関連付ける列インデックス。
- 指定された名前に関連付ける値。

例外

- 列インデックスが無効です。

結果

0 (正常終了) またはエラー コードを返します。

例

ASCII バージョン

```
int nRet;
nRet= setByIndex (dataRow, 1, "Austin");
if(nRet != SUCCESSFUL_RETURN)
{
printf(getErrorMessage(nRet));
 //more code
}
```

Unicode バージョン

```
int nRet;
UChar* error;
UChar columnValue[64];
char* columnValueStr= "Austin";
u_charsToUChars(columnValueStr, columnValue, strlen(columnValueStr));
columnValue [strlen(columnValueStr)]=0;
nRet= setByIndex (dataRow, 1, columnValue);
if(nRet != SUCCESSFUL_RETURN)
{
error = getErrorMessage(nRet);
//more code
}
```

AddChild

新しい DataRow を指定された親子関係に追加します。 指定された関係が存在する場合、与えられた DataRow は既存の DataRow コレクションに追加されます。 存在しない場合、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。

構文

ASCII バージョン

```
void addChild(DataRow* dataRow, const char* childName, DataRow*
childDataRow)
```

Unicode バージョン

```
void addChild(DataRow* dataRow, const UChar* childName, DataRow*
childDataRow)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。
- 関係に追加する DataRow。

例

ASCII バージョン

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();

setByName(child1DataRow1, "City", "Austin");
setByName(child1DataRow1, "State", "Texas");

addChild( dataRow, "child1", child1DataRow1);
```

Unicode バージョン

```
UChar* convertcharToUChar( char* name, UChar* value)
 {
 int lenName= strlen(name);

 u_charsToUChars(name, value, lenName );

 value[ lenName]=0;
 return value;
 } >
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
UChar     name[128];
UChar     columnValue[128];
setByName(child1DataRow1, convertcharToUChar("City", name),
  convertcharToUChar("Austin", columnValue));
setByName(child1DataRow1, convertcharToUChar("State", name),
  convertcharToUChar("Texas", columnValue));
addChild( dataRow, "child1", child1DataRow1);
```

GetChildren

指定された関係から子の行を取得します。

構文

ASCII バージョン

```
DataRow** getChildren(DataRow* dataRow,const char* childName)
```

Unicode バージョン

```
DataRow** getChildren(DataRow* dataRow,const UChar* childName)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。

結果

指定された関係から子の行を返します。

例

ASCII バージョン

```
DataRow** child1Rows;
child1Rows = getChildren(dataRow, "child1");
```

Unicode バージョン

```
DataRow** child1Rows;
UChar childName[128];
/* see convertcharToUChar in the Example section of "addChild" */
child1Rows = getChildren(dataRow, convertcharToUChar("child1",
childName));
```

ListChildNames

指定された親子関係のすべての名前を取得します。

構文

ASCII バージョン

```
char** listChildNames(DataRow* dataRow)
```

Unicode バージョン

```
UChar** listChildNames(DataRow* dataRow)
```

結果

指定された親/子関係の名前セットを返します。

例

ASCII バージョン

```
char** childsNames;
childsNames =listChildNames( dataRow);
```

Unicode バージョン

```
UChar** childsNames;
childsNames=listChildNames( dataRow);
```

SetChildren

与えられた指定の親子関係の行を設定します。 この名前で行がすでに存在する場合は、呼び出し元に返されます。

構文

ASCII バージョン

```
DataRow** setChildren(DataRow* dataRow, const char* childName, DataRow**
 dataRows)
```

Unicode バージョン

```
DataRow** setChildren(DataRow* dataRow, const UChar* childName, DataRow**
 dataRows)
```

結果

指定された親/子関係の名前セットを返します。

例

ASCII バージョン

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
DataRow* child1DataRow2 = createDataRow();
DataRow* child2DataRow = createDataRow();
DataRow** child2Rows;
DataRow** returnRows;

setByName(child1DataRow1, "Address", "200 Congress");
setByName(child1DataRow1, "City", "Austin");
```

```
setByName(child1DataRow2, "Address", "100 Congress");
setByName(child1DataRow2, "City", "Dallas");

setByName(child2DataRow, "Address", "100 Congress");
setByName(child2DataRow, "City", "Austin");

addChild( dataRow, "child1", child1DataRow1);
addChild( dataRow, "child1", child1DataRow2);
addChild( dataRow, "child2", child2DataRow );

child2Rows=getChildren(dataRow, "child2");

returnRows=setChildren( dataRow, "child1", child2Rows);
```

Unicode バージョン

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
DataRow* child1DataRow2 = createDataRow();
DataRow* child2DataRow = createDataRow();
DataRow** child2Rows;
DataRow** returnRows;
UChar name[128];
UChar columnValue[128];
UChar childName[128];

setByName(child1DataRow1, convertcharToUChar("Address", name),
convertcharToUChar("200 Congress", columnValue));
setByName(child1DataRow1, convertcharToUChar("City", name),
convertcharToUChar("Austin", columnValue));
setByName(child1DataRow2, convertcharToUChar("Address", name),
convertcharToUChar("100 Congress", columnValue));
setByName(child1DataRow2, convertcharToUChar("City", name)
convertcharToUChar("Dallas", columnValue) );
setByName(child2DataRow, convertcharToUChar("Address", name),
convertcharToUChar("100 Congress", columnValue) );
setByName(child2DataRow, convertcharToUChar("City", name),
convertcharToUChar("Austin", columnValue) );

addChild( dataRow, convertcharToUChar("child1", childName),
child1DataRow1);
addChild( dataRow, convertcharToUChar("child1",
childName),child1DataRow2);
addChild( dataRow, convertcharToUChar("child2", childName), child2DataRow
 );

child2Rows=getChildren(dataRow, convertcharToUChar("child2",childName));

returnRows=setChildren( dataRow, convertcharToUChar("child1",childName),
 child2Rows);
```

3 - C++ API

このセクションの構成

C++ API の概要	74
Server	88
Service	92
Message	92
DataTable	103
DataRow	111

C++ API の概要

C++ API は、次のクラスで構成されます。

- Server
- Service
- Message
- DataTable
- DataRow

ICU の UnicodeString は文字列クラスで、Unicode 文字を直接格納し、Java String クラスおよび StringBuffer クラスと同様の機能を提供します。Spectrum™ Technology Platform の Unicode C++ API は、このクラスを使用して Unicode 文字列を格納します。

サポートされるライブラリ

Spectrum™ Technology Platformは、ASCII バージョンと Unicode バージョンの C API を提供します。Unicode バージョンでは、元の ASCII バージョンの API 設計との互換性が極力維持されます。Spectrum™ Technology Platformは、Unicode 機能をサポートするために International Components for Unicode (ICU) が API に適用されます。ICU は、長年にわたって広く利用されている Unicode サポート用の C/C++ ライブラリであり、IBM で開発されました。

Unicode 規格は、16 ビットのコード単位に基づいてデフォルトのエンコーディングを定義します。ICU では、UChar を符号なしの 16 ビット整数タイプ (unsigned short *) として定義することで Unicode がサポートされます。これが、ICU で文字列を表す文字配列の基本型です。Spectrum™ Technology Platformは、C API で Unicode 文字列を表現するために UChar を使用します。

> **注：** 一部のサービスは、Unicode 文字セットを完全にはサポートしません。例えば、ValidateAddress サービスは、米国入力/国際入出力用に ISO 8859-1 文字セットをサポートし、カナダ入出力用に CP 850 文字セットをサポートします。ただし、入力データに ASCII ではない文字が含まれる場合は、基本サービスが Unicode 文字セットを完全にサポートしていなくても Unicode ライブラリが使用されます。

UChar の詳細については、次の 2 つのサイトを参照してください。

- icu.sourceforge.net/userguide/
- www-306.ibm.com/software/globalization/icu/index.jsp

Windows

各 API 設定から生成されるライブラリ ファイルの名前は、共通の基本名 (g1client) に固有の接尾文字と、場合によってはさらに接頭文字 (静的ライブラリであれば "lib") が付加されたフォーマットになります。ライブラリの接尾文字は、次の意味を持ちます。

```
<lib>g1client<S><U><D>.<lib|dll>
```

- lib — 静的ライブラリ
- dll — 動的 (共有) ライブラリ
- S — 単一スレッド ビルド。この文字がないのは、マルチスレッド ビルドであることを意味します。
- U — UNICODE バージョン ビルド。この文字がないのは、ASCII ビルドであることを意味します。
- D — デバッグ用ビルド。この文字がないのは、最適化されたリリース用ビルドであることを意味します。

UNICODE バージョンを有効にするには、LIB_UNICODE マクロ定義がプロジェクトに存在する必要があります。

静的 C/C++ API ライブラリ UNICODE バージョンを使うには、U_STATIC_IMPLEMENTATION をプロジェクトで定義する必要があります。

動的バージョンを使うには、G1CLIENT_DLL をプロジェクトで定義する必要があります。

また、"auto_link.h" というファイルをヘッダー ファイル ディレクトリに配置します。このファイルは、プロジェクト設定に従ってすべての対応するライブラリに自動的にリンクします。

Windows で 64 ビット ライブラリを呼び出すには、VER_64 をプロジェクトで定義する必要があります。

静的ライブラリ

> **注 :** このセクションに記載されている名前は 32 ビット ライブラリ用です。64 ビット ライブラリ用は、ライブラリ名の "32" を "64" に置き換えてください。

単一スレッド/リリース

	Ascii	Unicode
G1	libg1client_S.lib	libg1client_SU.lib

openssl	otlibeay32.lib otlibssl32.lib	otlibeay32.lib otlibssl32.lib
opentop	opentop.lib	opentopw.lib
icu		libicuuc.lib libicudt.lib libicuin.lib libicuio.lib
Poco	PocoXML32.lib	PocoXML32w.lib
単一スレッド/デバッグ		
	Ascii	Unicode
G1	libg1client_SD.lib	libg1client_SUD.lib
openssl	otlibeay32d.lib otlibssl32d.lib	otlibeay32d.lib otlibssl32d.lib
opentop	opentopd.lib	opentopwd.lib
icu		libicuucd.lib libicudtd.lib libicuind.lib libicuiod.lib
Poco	PocoXML32d.lib	PocoXML32wd.lib
マルチ/リリース (マルチスレッド CRT 使用)		
	Ascii	Unicode
G1	libg1client.lib	libg1client_U.lib
openssl	otlibeay32mt.lib otlibssl32mt.lib	otlibeay32mt.lib otlibssl32mt.lib

opentop	opentopmt.lib	opentopmtw.lib
icu		libicuucmt.lib libicudtmt.lib libicuinmt.lib libicuiomt.lib
Poco	PocoXMLmt32.lib	PocoXML32mtw.lib

マルチ/デバッグ (マルチスレッド CRT 使用)

	Ascii	Unicode
G1	libg1client_D.lib	libg1client_UD.lib
openssl	otlibeay32mtd.lib otlibssl32mtd.lib	otlibeay32mtd.lib otlibssl32mtd.lib
opentop	opentopmtd.lib	opentopmtwd.lib
icu		libicuucmtd.lib libicudtmtd.lib libicuinmtd.lib libicuiomtd.lib
Poco	PocoXMLmt32d.lib	PocoXML32mtwd.lib

動的ライブラリ

注: このセクションに記載されている名前は 32 ビット ライブラリ用です。64 ビット ライブラリ用は、ライブラリ名の "32" を "64" に置き換えてください。

マルチ/リリース (マルチスレッド CRT 使用)

	Ascii	Unicode
G1	g1client.dll	g1client_U.dll

openssl	otlibeay32mts.dll otlibssl32mts.dll	otlibeay32mts.dll otlibssl32mts.dll
opentop	opentopmts.dll	opentopmtws.dll
icu		icuuc32.dll icuio32.dll icuin32.dll icudt32.dll
Poco	PocoXML32mts.dll	PocoXML32mtws.dll
マルチ/デバッグ (マルチスレッド CRT 使用)		
	Ascii	Unicode
G1	g1client_D.dll	g1client_UD.dll
openssl	otlibeay32mtds.dll otlibssl32mtds.dll	otlibeay32mtds.dll otlibssl32mtds.dll
opentop	opentopmtds.dll	opentopmtwds.dll
icu		icuuc32d.dll icuio32d.dll icuin32d.dll icudt32d.dll
Poco	PocoXML32mtds.dll	PocoXML32mtwds.dll

Unix

各 ClientSDK 設定から生成されるライブラリ ファイルの名前は、共通の基本名 (libg1client) に固有の接尾文字が付加されたフォーマットになります。Spectrum™ Technology Platformは、ASCII バージョンと UNICODE バージョンのマルチスレッド/リリース ビルドを提供します。

ライブラリの接尾文字は、次の意味を持ちます。

```
libg1client<U>.<so|sl|a>
```

- U — UNICODE バージョン ビルド。この文字がないのは、ASCII ビルドであることを意味します。

UNICODE バージョンを使うには、LIB_UNICODE をプロジェクトで定義する必要があります。

UNICODE バージョンの C++ API では、すべてのクラスのネームスペースが g1client になります。

AIX

	Ascii	Unicode
G1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libssl.so	libcrypto.so libssl.so
opentop	libopentop-xlCmt.so	libopentop-xlCmtw.so libotxml-xlCmtw.so
icu		libicudata34.a libicui18n34.a libicuio34.a libicuuc34.a
Poco	libPocoXML.so	

HP-UX

	Ascii	Unicode
G1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicui18n.sl libicui18n.sl.34 libicuio.sl libicuio.sl.34 libicuuc.sl libicuuc.sl.34

Poco	libPocoXML.sl	
Itanium		
	Ascii	Unicode
G1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.a libssl.a	libcrypto.a libssl.a
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicudata.sl.34.0 libicui18n.sl libicui18n.sl.34 libicui18n.sl.34.0 libicuio.sl libicuio.sl.34 libicuio.sl.34.0 libicuuc.sl libicuuc.sl.34 libicuuc.sl.34.0
Poco	libPocoXML.sl	
Linux		
	Ascii	Unicode
G1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-gccmt.so	libopentop-gccmtw.so libotxml-gccmtw.so

icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	
Solaris		
	Ascii	Unicode
G1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-fortemt.so	libopentop-fortemtw.so libotxml-fortemtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	

定数

C++ API では、2 組の定数が使用されます。最初の 1 組は `Server` クラス用です。以下の表に説明します。

表 14 : `Server` コンポーネントの定数

定数名	説明/デフォルト	例
Server::HOST	サーバーのホスト名を表す文字列。デフォルトは "localhost" です。	65.89.200.89
Server::PORT	サーバーのポートを表す文字列。デフォルトは "8080" です。	10119
Server::ACCOUNT_ID	サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1
Server::ACCOUNT_PASSWORD	サーバーのアカウント パスワードを表す文字列。デフォルト値はありません。	user1
Server::CONNECTION_TIMEOUT	サーバーの接続タイムアウトをミリ秒単位で表す文字列。デフォルトは "5000" です。	50000
Server::CONNECTION_TYPE	サーバーの接続タイプを表す文字列。現在は HTTP、HTTPS、または SOCKET のみがサポートされています。デフォルトは "HTTP" です。	HTTP(S)
Server::PROXY_HOST	プロキシ サーバーのホスト名を表す文字列。デフォルト値はありません。	192.168.1.77
Server::PROXY_PORT	プロキシ サーバーのポートを表す文字列。デフォルト値はありません。	8080
Server::PROXY_USER	プロキシ サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1

定数名	説明/デフォルト	例
Server::PROXY_PASSWORD	プロキシ サーバーのアカウント パスワードを表す文字列。デフォルト値はありません。	user1

2 組目の定数は `Message` クラス用です。

表 15 : `Message` コンポーネントの定数

定数名	説明	例
Message::CONTEXT_ACCOUNT_ID	メッセージコンテキストのアカウント ID を表す文字列。	user1
Message::CONTEXT_ACCOUNT_PASSWORD	メッセージコンテキストのアカウント パスワードを表す文字列。	user1
Message::CONTEXT_SERVICE_NAME	メッセージコンテキストのサービス名を表す文字列。	echoservice

エラー メッセージ

エラー メッセージを取得するには、Exception クラスを使用します。 try/catch 構造を使用して、エラー メッセージをキャプチャします。 例:

```
try{
  Server *server=new Server();

  //Connect to server
  server->connect();

  }catch(Exception e)
  {
   // ASCII Version-use the following code
  cout << "Error Occurs," << e.getErrorMessage();
```

```
  //Unicode Version -use the following code

  UnicodeString error = e.getErrorMessage() ;

  wcout << error.getTerminatedBuffer();
  }
```

C++ API では、次のエラー メッセージが使用されます。

- 接続エラー メッセージ:
  - "Connection type not supported"
  - "Client timeout"
  - "Blank connection property name"
  - "Blank property name"
- DataTable 作成時のエラー メッセージ:
  - "Blank column name"
  - "Duplicated column name"
  - "The column index is invalid"
- MessagePackaging 例外のエラー メッセージ:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect from Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

SmartPointer

Spectrum™ Technology Platform の SmartPointer クラスは、単純なリファレンス カウント方法を使って、動的メモリ割り当ての追跡とメモリ管理タスクの実行を容易にします。

例:

```
SmartPointer<Server> server =new Server();
server.connect();
...
server.disconnect();
```

ポインター サーバーのメモリを削除する必要はありません。 SmartPointer が代わりにすべてのメモリ管理を処理します。

サンプル アプリケーション

以下のサンプル コードに、ASCII バージョンの C++ API の使い方を示します。

```
try{

   //Create Server
   SmartPointer<Server> server =new Server();

   //Set server connection properties
   server->setConnectionProperty(Server::HOST, "localhost");
   server->setConnectionProperty(Server::PORT, "10119");
   server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
   server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
   server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

   //Connect to server
   server->connect();

   //Get Service From Server
   SmartPointer<Service> service = server-
>getService("ValidateAddress");

   //Create Input Message
   SmartPointer<Message> request = new Message();

   //Fill DataTable in the input message
   SmartPointer<DataTable> dataTable = request->getDataTable();
    SmartPointer<DataRow> row1 = dataTable->newRow();
    row1->set("AddressLine1", "4200 Parliament Place") ;
    row1->set("City", "Lanham");
    row1->set("StateProvince", "Maryland");
    dataTable->addRow(row1);

    SmartPointer<DataRow> row2 = dataTable->newRow();
    row2->set("AddressLine1", "100 Congress");
    row2->set("City", "Austin");
    row2->set("StateProvince", "Texas");
    dataTable->addRow(row2);

   //Set"option" Properties to the Input Message
    request->putOption("OutputCasing", "M");
    request->putOption("OutputRecordType", "A");

   //Process Input Message, return output Message
    SmartPointer<Message> reply = service->process(request);

    //Disconnect from server
   server->disconnect();
```

```
   //Get the result from the resonse message
   SmartPointer<DataTable> returnDataTable = reply->getDataTable();

    vector<string> columnName = returnDataTable- >getColumnNames();
      vector< SmartPointer<DataRow> >::iterator iter =
returnDataTable->iterator();

     for (int i=0; i< returnDataTable->getRowCount(); i++, iter++)
   {
   SmartPointer<DataRow> dataRow = *iter;

   for (int col = 0; col < returnDataTable->getColumnCount(); col++)
   {
   const char* value = dataRow->get(columnName[col].c_str());
   cout << value << "\n";
  }
   }
   }catch(Exception e)
   {
   cout << "Error Occurred, " << e.getErrorMessage();
   }
```

以下のサンプル コードに、Unicode バージョンの C++ API の使い方を示します。

```
try{
 //Create Server
 SmartPointer<Server> server =new Server();

 //Set server connection properties
 server->setConnectionProperty(Server::HOST,"localhost");
 server->setConnectionProperty(Server::PORT, "10119");
 server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
 server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
 server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

 //Connect to server
 server->connect();

 //Get Service From Server
 //NOTE: ValidateAddress does not support unicode, but supports
 //characters in Canadian address and International address data files.

 SmartPointer<Service> service = server- >getService("ValidateAddress");


 //Create Input Message
 SmartPointer<Message> request = new Message();

 //Fill DataTable in the input message
 SmartPointer<DataTable> dataTable = request->getDataTable();
 dataTable->addColumn("AddressLine1");
```

```
 dataTable->addColumn("City");
 dataTable->addColumn("PostalCode");
 dataTable->addColumn("Country");

 SmartPointer<DataRow> row1 = dataTable->newRow();

 UnicodeString address1 = "74, Rue Octave Bénard";
 row1->set( 0 , address1);
 UnicodeString city1 = "Etang-Salé-les-Bains";
 row1->set( 1 , city1);
 UnicodeString postalCode1 = "97427";
 row1->set( 2 , postalCode1);
 UnicodeString country1 = "Reunion Island";
 row1->set( 3 , country1);

 dataTable->addRow(row1);

 SmartPointer<DataRow> row2 = dataTable->newRow();
 UnicodeString address2 = "Final Av. Panteón Foro Libertador";
 row2->set( 0 , address2);
 UnicodeString city2 = "Caracas";
 row2->set( 1 , city2);
 UnicodeString postalCode2 = "1010";
 row2->set( 2 , postalCode2);
 UnicodeString country2 = "Venezuela";
 row2->set( 3 , country2);

 dataTable->addRow(row2);

 //Set"option" Properties to the Input Message
 request->putOption("OutputCasing", "M");
 request->putOption("OutputRecordType", "A");

 //Process Input Message, return output Message
 SmartPointer<Message> reply = service->process(request);

 //Disconnect from server
 server->disconnect();

 //Get the result from the resonse message
 SmartPointer<DataTable> returnDataTable = reply->getDataTable();

 vector<UnicodeString> columnName = returnDataTable- >getColumnNames();


 vector< SmartPointer<DataRow> >::iterator iter = returnDataTable-
>iterator();

 for (int i=0; i< returnDataTable->getRowCount(); i++, iter++)
 {
 SmartPointer<DataRow> dataRow = *iter;

 for (int col = 0; col < returnDataTable->getColumnCount(); col++)
```

```
{
UnicodeString value = dataRow->get(columnName[col]);
wcout <<value.getTerminatedBuffer() <<"\n"; }
}

}catch(Exception e)
{
UnicodeString error = e.getErrorMessage() ;

wcout << error.getTerminatedBuffer();
}
```

Server

`Server` クラスは、サーバーへの接続、サーバーからの切断、およびサーバーからのサービスの取得に使用されます。

コンストラクタ

Server クラスのコンストラクタには次のものがあります。

- Server()

デストラクタ

Server クラスのデストラクタには次のものがあります。

- ~Server()

Connect

プロパティを読み取って、設定値を決定し、サーバーへの接続を確立します。HTTP、HTTPS、またはソケットを介して接続できます。

> **注：** C++ では、HTTP、HTTPS、またはソケット サーバー接続プロトコルを使用します。HTTP と HTTPS は、クライアント接続を論理的に確立するだけで、GetService メソッド

または Process メソッドが呼び出されるまで実際にはサーバーに接続しません。ソケットプロトコルは、`Connect` が呼び出された時点でサーバーへの接続を確立します。

構文

```
void connect()
```

パラメータ

なし

結果

サーバーへのクライアント接続を確立します。

例

```
//Create Server
SmartPointer<Server> server =new Server();

//Set server connection properties
server->setConnectionProperty(Server::HOST,"localhost");
server->setConnectionProperty(Server::PORT, "10119");
server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

//Connect to server
server->connect();
```

Disconnect

サーバーから切断します。

構文

```
void disconnect()
```

パラメータ

なし

結果

クライアントがサーバーから切断されます。

例

```
SmartPointer<Server> server =new Server()
server->connect();
...
server->disconnect();
```

SetConnectionProperty

ホスト名、タイアウト時間など、サーバー接続設定プロパティを設定します。

構文

ASCII バージョン

```
void setConnectionProperty(const char* name, const char* value)
```

Unicode バージョン

```
void setConnectionProperty(const UnicodeString name, const UnicodeString
 value)
```

パラメータ

- Name — 接続プロパティの名前。HOST など。
- Value — 接続プロパティの値。"www.myhost.com" など。

結果

サーバーに接続するための設定プロパティが設定されます。

例

ASCII バージョン

```
SmartPointer<Server> server =new Server()
server->setConnectionProperty(Server::HOST,"localhost");
server->setConnectionProperty(Server::PORT, "8080");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
SmartPointer<Server> server =new Server()
UnicodeString host="localhost";// Or input unicode string
server->setConnectionProperty(Server::HOST, host);
```

GetService

サーバーからサービスを取得します。

> **注：** 使用可能なサービスのリストについては、このガイドの「コンポーネント リファレンス」を参照してください。

構文

ASCII バージョン

```
SmartPointer<Service> getService(const char* serviceName)
```

Unicode バージョン

```
SmartPointer<Service> getService(const UnicodeString serviceName)
```

パラメータ

- サービス名

結果

特定のサービスを返します。

例

ASCII バージョン

```
// Get Service From Server
SmartPointer<Service> service = server- >getService("ValidateAddress");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
// Get Service From Server
UnicodeString serviceName="ValidateAddress";// Or input unicode string
SmartPointer<Service> service = server->getService(serviceName);
```

Service

`Service` クラスは、メッセージを処理するために使用されます (より具体的に言えば、メッセージをサーバーに送信し、サーバーから応答を受信するために使用されます)。

Process

入力メッセージを処理し、応答メッセージを返します。

構文

```
SmartPointer<Message> process(Message* message)
```

パラメータ

- 入力メッセージ

結果

応答メッセージを返します。

例

```
SmartPointer<Message> reply = service->process(request);
```

Message

`Message` クラスは、入力データを送信し、サービスから出力データを受け取ります。`Message` のプロパティには、コンテキスト エンティティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、オプション エンティティ (サービス固有の実行時プロパティ)、エラー エンティティ (エラー クラス、エラー メッセージ、エラー スタックトレース) などがあります。

コンストラクタ

Message クラスのコンストラクタには次のものがあります。

- `Message()`

例:

```
Message *request = new Message():
```

- `Message(const Message&)`

例:

```
Message* request = new Message();
Message anotherMessage = request;
Message message(anotherMessage);
```

デストラクタ

Message クラスのデストラクタには次のものがあります。

- `~Message();`

以下の表に、Message クラスで実行されるメソッドの概要を示します。

表 16 : `Message` メソッドの概要

方法	関数
getContext	メッセージのコンテキスト セッションに指定された名前のコンテキスト エンティティから値を取得します。
getContext	すべてのコンテキスト エントリが含まれるマップを取得します。

方法	関数
putContext	メッセージのコンテキスト セクションに指定された名前のコンテキスト エンティティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。
putContext	新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。
setContext	新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。
getOption	メッセージのオプション セッションに指定された名前のオプション エンティティから値を取得します。
getOptions	すべてのオプション エントリが含まれるマップを取得します。
putOption	メッセージのオプション セッションに指定された名前のオプション エンティティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。
putOptions	新しいオプション プロパティを現在のオプション プロパティに追加します。
setOptions	新しいオプション プロパティで現在のオプション プロパティを上書きします。
getError	エラー メッセージを取得します。
getDataTable	DataTable をメッセージから取得します。

GetContext

メッセージのコンテキスト セッションに指定された名前のコンテキスト エンティティから値を取得します。

構文

ASCII バージョン

```
const char* getContext(const char* name)
```

Unicode バージョン

```
const UnicodeString getContext(const UnicodeString name)
```

パラメータ

- 関連付けられた値を返す名前。

結果

コンテキスト エンティティにある名前の値を返します。 名前が存在しない場合、空白の文字列を返します。

例

ASCII バージョン

```
const char* value= msg->getContext(Server::ACCOUNT_ID);
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString name= Server::ACCOUNT_ID;// Or input unicode string
UnicodeString value= msg->getContext(name);
```

GetContext

すべてのコンテキスト エントリが含まれるマップを取得します。

構文

ASCII バージョン

```
map<string , string> getContext()
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > getContext()
```

パラメータ

なし

結果

すべてのコンテキスト エントリが含まれるマップを返します。

例

ASCII バージョン

```
map<string , string> context = message->getContext();
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > context = message- >getContext();
```

PutContext

指定された名前に基づいてコンテキスト プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

ASCII バージョン

```
void putContext(const char* name, const char* value)
```

Unicode バージョン

```
void putContext(const UnicodeString name, const UnicodeString value)
```

パラメータ

- 指定された値を関連付ける名前。
- 指定された名前に関連付ける値。

例

ASCII バージョン

```
message->putContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString account="user1" ;// Or input unicode string
message->putContext(Message.CONTEXT_ACCOUNT_ID, account);
```

PutContext

新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。

構文

ASCII バージョン

```
void putContext(map<string , string> context)
```

Unicode バージョン

```
void putContext(map< UnicodeString, UnicodeString > context)
```

パラメータ

- 現在のコンテキスト マップに追加する新しいコンテキスト マップ。

例

ASCII バージョン

```
map<string , string> context ;
//more code
message->putContext(context);
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > context ;
//more code
message->putContext(context);
```

SetContext

新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。

構文

ASCII バージョン

```
void setContext(map<string , string> context)
```

Unicode バージョン

```
void setContext(map< UnicodeString, UnicodeString > context)
```

パラメータ

- 現在のコンテキスト マップを置き換えるために使用する新しいコンテキスト マップ。

例

ASCII バージョン

```
map<string , string> context ;
//more code
message->setContext(context);
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > context ;
//more code
message->setContext(context);
```

GetOption

メッセージのオプション セクションに指定された名前のオプション エンティティから値を取得します。オプション エンティティには、出力の大文字と小文字の区別、出力データのフォーマットなど、サービス固有の実行時オプションが含まれます。

構文

ASCII バージョン

```
const char* getOption(const char* name)
```

Unicode バージョン

```
const UnicodeString getOption(const UnicodeString name)
```

パラメータ

- 関連付けられた値を返す名前。

結果

コンテキスト エンティティにある名前の値を返します。名前が存在しない場合、空白の文字列を返します。

例

ASCII バージョン

```
const char* value = message->getOption("OutputCasing");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString option="OutputCasing"; // Or input unicode string
UnicodeString value= message->getOption(option);
```

GetOptions

すべてのオプション エントリが含まれるマップを取得します。

構文

ASCII バージョン

```
map<string , string> getOptions()
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > getOptions()
```

パラメータ

なし

結果

すべてのオプション エントリが含まれるマップを返します。

例

ASCII バージョン

```
const char* value = message->getOption("OutputCasing");
```

Unicode バージョン

```
UnicodeString option="OutputCasing"; //or input Unicode string
UnicodeString value= message->getOption(option);
```

PutOption

指定された名前に基づいてオプション プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。オプション プロパティはサービス固有の実行時オプションです。

構文

ASCII バージョン

```
void putOption(const char* name, const char* value)
```

Unicode バージョン

```
void putOption(const UnicodeString name, const UnicodeString value)
```

パラメータ

- 指定された値を関連付ける名前。
- 指定された名前に関連付ける値。

例

ASCII バージョン

```
message->putOption("OutputCasing", "M");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString option="M"; // Or input unicode string
message->putOption("OutputCasing", option);
```

PutOptions

新しいオプション プロパティを現在のオプション プロパティに追加します。

構文

ASCII バージョン

```
void putOptions(map<string , string> options)
```

Unicode バージョン

```
void putOptions(map< UnicodeString, UnicodeString > options)
```

パラメータ

- 現在のオプション プロパティに追加する新しいオプション マップ。

例

ASCII バージョン

```
map<string , string> options ;
//more code
message->putOptions(options);
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > options ;
//more code
message->putOptions(options);
```

SetOptions

新しいオプション プロパティで現在のオプション プロパティを上書きします。

構文

ASCII バージョン

```
void setOptions(map<string , string> options)
```

Unicode バージョン

```
void setOptions(map< UnicodeString, UnicodeString > options)
```

パラメータ

- 現在のオプション マップを置き換えるために使用する新しいオプション マップ。

例

ASCII バージョン

```
map<string , string> options ;
//more code
message->setOptions(options);
```

Unicode バージョン

```
map< UnicodeString, UnicodeString > options ;
//more code
message->setOptions(options);
```

GetError

メッセージからエラー メッセージを取得します。

構文

ASCII バージョン

```
string getError()
```

Unicode バージョン

```
UnicodeString getError()
```

パラメータ

なし

結果

メッセージからエラー メッセージを取得して返します。

例

ASCII バージョン

```
String error = message->getError();
```

Unicode バージョン

```
UnicodeString error = message->getError();
```

GetDataTable

DataTable をメッセージから取得します。

構文

```
SmartPointer<DataTable> getDataTable()
```

パラメータ

なし

例

```
SmartPointer<DataTable> dataTable
= message->getDataTable();
```

DataTable

DataTable には入出力データのレコードが含まれます。

コンストラクタ

DataTable クラスのコンストラクタには次のものがあります。

- `DataTable()`

例:

```
DataTable* dataTable = new DataTable()
```

デストラクタ

DataTable クラスのデストラクタには次のものがあります。

- `~DataTable();`

以下の表に、DataTable クラスで実行されるメソッドの概要を示します。

表 17 : DataTable メソッドの概要

方法	関数
addColumn	新しい列を追加します。
getColumnNames	すべての列名を取得します。
getColumnIndex	対応する列インデックスを取得します。
getColumnCount	列の数を取得します。
clear	DataTable 内のデータを消去します。
iterator	DataTable 内のすべての DataRow を含むイテレータです。
addRow	DataRow を DataTable に追加します。
newRow	新しい DataRow を DataTable 内に作成します。

方法	関数
getRowCount	この DataTable にある DataRow の数を取得します。
merge	指定された DataTable と現在の DataTable を結合します。

AddColumn

新しい列を追加します。

構文

ASCII バージョン

```
int addColumn(const char* columnName)
```

Unicode バージョン

```
int addColumn(const UnicodeString columnName)
```

パラメータ

- 列名

結果

- 列のインデックスを返します。

例外

- 列名が空白です。
- 同名の列がすでにあります。

例

ASCII バージョン

```
SmartPointer<DataTable> dataTable = message.getDataTable();
dataTable->addColumn("Address");
dataTable->addColumn("City");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
SmartPointer<DataTable> dataTable = message.getDataTable();
UnicodeString columnName="Address"; // Or input unicode string
dataTable->addColumn(columnName);
```

GetColumnNames

すべての列名を取得します。

構文

ASCII バージョン

```
vector<string> getColumnNames();
```

Unicode バージョン

```
vector<UnicodeString> getColumnNames();
```

パラメータ

なし

結果

列名のベクトルを返します。

例

ASCII バージョン

```
vector<string> columnNames = dataTable->getColumnNames();
```

Unicode バージョン

```
vector<UnicodeString> columnNames = dataTable->getColumnNames();
```

GetColumnIndex

対応する列インデックスを取得します。

構文

ASCII バージョン

```
int getColumnIndex(const char* columnName)
```

Unicode バージョン

```
int getColumnIndex(const UnicodeString columnName)
```

パラメータ

- 列名

結果

対応する列インデックスを返します。

例

ASCII バージョン

```
int columnIndex = dataTable->getColumnIndex ("City");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString columnName="City"; // Or input unicode string
int columnIndex = dataTable->getColumnIndex (columnName);
```

GetColumnCount

列の数を取得します。

構文

```
int getColumnCount()
```

パラメータ

なし

結果

列の数を返します。

例

```
int columnCount = dataTable->getColumnCount ();
```

Clear

DataTable 内のデータを消去します。

構文

```
void clear()
```

パラメータ

なし

例

```
dataTable->clear();
```

Iterator

DataTable 内のすべての DataRow を含むイテレータです。

構文

```
vector< SmartPointer<DataRow> >::iterator iterator()
```

パラメータ

なし

結果

DataTable 内のすべての DataRow を含むイテレータを返します。

例

```
vector<string> columnName
= returnDataTable->getColumnNames();
```

```
vector< SmartPointer<DataRow> >::iterator theIterator
= returnDataTable->iterator();

for (int i=0; i< returnDataTable->getRowCount();
i++, theIterator++)
{
SmartPointer<DataRow> dataRow = *theIterator;

for (int col = 0;
col < returnDataTable->getColumnCount(); col++)
 {
 const char* value = dataRow->get(columnName[col].c_str());
 }
 }
```

AddRow

DataRow を DataTable に追加します。

構文

```
void addRow( SmartPointer<DataRow> dataRow)
```

パラメータ

- DataTable に追加する DataRow。

例

```
SmartPointer<DataRow> newRow = dataTable->newRow();
 newRow->set( 0 , "10535 Boyer");
 newRow->set( 1 , "Austin");
 newRow->set( 2 , "Texas");
 dataTable->addRow(newRow);
```

NewRow

新しい DataRow を DataTable 内に作成します。

構文

```
SmartPointer<DataRow> newRow()
```

結果

作成された新しい DataRow を返します。

例

```
SmartPointer<DataRow> newRow = dataTable->newRow();
 newRow->set( 0 , "10535 Boyer");
 newRow->set( 1 , "Austin");
 newRow->set( 2 , "Texas");
 dataTable->addRow(newRow);
```

GetRowCount

この DataTable にある DataRow の数を取得します。

構文

```
int getRowCount()
```

結果

この DataTable にある DataRow の数を返します。

例

```
int rowCount = dataTable->getRowCount();
```

Merge

指定された DataTable と現在の DataTable を結合します。

構文

```
void merge(DataTable* other)
```

パラメータ

- 現在の DataTable と結合する他の DataTable。

例

```
DataTable* otherDataTable = new DataTable();
dataTable->merge(otherDataTable);
```

DataRow

DataRow には入出力データのレコードが含まれます。

コンストラクタ

DataRow クラスのコンストラクタには次のものがあります。

- `DataRow ()`

例:

```
DataRow * dataRow = new DataRow();
```

- `DataRow(const DataRow&)`

例:

```
DataRow* dataRow = new DataRow();
DataRow anotheDataRow = dataRow;
DataRow newDataRow(anotheDataRow);
```

デストラクタ

DataRow クラスのデストラクタには次のものがあります。

- `~ DataRow();`

以下の表に、DataRow クラスのメソッドが実行する関数の概要を示します。

表 18 : DataRow メソッドの概要

方法	関数
getColumnNames	すべての列名を取得します。
getColumnIndex	対応する列インデックスを取得します。
getColumnCount	列の数を取得します。
get	この DataRow の列インデックスによってフィールド配列から値を取得します。
get	この DataRow の列名によってフィールド配列から値を取得します。
merge	指定された DataTable と現在の DataTable を結合します。
set	DataRow の対応する列名の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。
set	DataRow の対応する列インデックスの値を設定します。この名前の値が存在する場合は、古い値を置き換えます。
addChild	新しい DataRow を指定された親子関係に追加します。指定された関係が存在する場合は、与えられた DataRow は既存の DataRow コレクションに追加されます。存在しない場合は、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。
getChildren	指定された関係から子の行を取得します。
listChildNames	指定された親子関係のすべての名前を取得します。

方法	関数
setChildren	与えられた指定の親子関係の行を設定します。この名前で行がすでに存在する場合は、呼び出し元に返されます。

GetColumnNames

すべての列名を取得します。

構文

ASCII バージョン

```
vector<string> getColumnNames()
```

Unicode バージョン

```
vector<UnicodeString> getColumnNames()
```

パラメータ

なし

結果

列名のベクトルを返します。

例

ASCII バージョン

```
vector<string> columnNames = dataRow->getColumnNames();
```

Unicode バージョン

```
vector<UnicodeString> columnNames = dataRow->getColumnNames();
```

GetColumnIndex

対応する列インデックスを取得します。

構文

ASCII バージョン

```
int getColumnIndex(const char* columnName)
```

Unicode バージョン

```
int getColumnIndex(const UnicodeString columnName)
```

パラメータ

- 列名

結果

対応する列インデックスを返します。

例

ASCII バージョン

```
int columnIndex = dataRow->getColumnIndex ("City");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString columnName="City"; // Or input unicode string
int columnIndex = dataRow->getColumnIndex (columnName);
```

GetColumnCount

列の数を取得します。

構文

```
int getColumnCount()
```

パラメータ

なし

結果

列の数を返します。

例

```
int columnCount = dataRow->getColumnCount ();
```

Get

この DataRow の列インデックスによってフィールド配列から値を取得します。

構文

ASCII バージョン

```
const char* get(int index)
```

Unicode バージョン

```
const UnicodeString get(int index)
```

パラメータ

- 指定された値を関連付けるインデックス。

結果

DataRow の列インデックスの値を返します。インデックスが無効の場合は空の文字列を返します。

例

ASCII バージョン

```
const char* value = dataRow->get(1);
```

Unicode バージョン

```
const UnicodeString value = dataRow->get(1);
```

Get

この DataRow の列名によってフィールド配列から値を取得します。

構文

ASCII バージョン

```
const char* get(const char* columnName)
```

Unicode バージョン

```
const UnicodeString get(const UnicodeString columnName)
```

パラメータ

- 指定された値を関連付ける名前。

結果

DataRow の列名の値を返します。列名が存在しない場合は空の文字列を返します。

例

ASCII バージョン

```
const char* value = dataRow->get("City");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
UnicodeString columnName="City"; // Or input unicode string
const UnicodeString value = dataRow->get(columnName);
```

Merge

指定された DataRow と現在の DataRow を結合します。

構文

```
void merge(DataRow* other)
```

パラメータ

- 現在の DataRow と結合する他の DataRow。

例

```
DataRow* otherDataRow = new DataRow();
DataRow->merge(otherDataRow);
```

Set

DataRow の対応する列の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。

構文

ASCII バージョン

```
void set(const char* columnName, const char* value)
```

Unicode バージョン

```
void set(const UnicodeString columnName, const UnicodeString value)
```

パラメータ

- 指定された値を関連付ける名前。
- 指定された名前に関連付ける値。

Exceptions

- 列名が空白です。
- 同名の列がすでにあります。

例

ASCII バージョン

```
SmartPointer<DataRow> newRow = dataTable->newRow();
newRow->set( "AddressLine1" , "10535 Boyer");
newRow->set( "City" , "Austin");
newRow->set( "State" , "Texas");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
SmartPointer<DataRow> newRow = dataTable->newRow();
UnicodeString address="10535 Boyer"; // Or input unicode string
newRow->set( "AddressLine1" , address);
```

Set

DataRow の対応する列の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。

構文

ASCII バージョン

```
void set(int index, const char* value)
```

Unicode バージョン

```
void set(int index, const UnicodeString value)
```

パラメータ

- 指定された値を関連付ける列インデックス。
- 指定された名前に関連付ける値。

Exceptions

- 列インデックスが無効です。

例

ASCII バージョン

```
SmartPointer<DataRow> newRow = dataTable->newRow();
newRow->set( 0 , "10535 Boyer");
newRow->set( 1 , "Austin");
newRow->set( 2 , "Texas");
```

Unicode バージョン

ASCII バージョンと同じです。または、以下のようになります。

```
SmartPointer<DataRow> newRow = dataTable->newRow();
UnicodeString address="10535 Boyer"; // Or input unicode string
newRow->set( 0 , address);
```

AddChild

新しい DataRow を指定された親子関係に追加します。 指定された関係が存在する場合、与えられた DataRow は既存の DataRow コレクションに追加されます。 存在しない場合、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。

構文

ASCII バージョン

```
void addChild(const char* childName, SmartPointer<DataRow> childDataRow)
```

Unicode バージョン

```
void addChild(const UnicodeString childName, SmartPointer<DataRow>
childDataRow)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。
- 関係に追加する DataRow。

例

```
SmartPointer<DataRow> childDataRow =new DataRow();
childDataRow ->set("Address", "100 Congress");
childDataRow ->set("City", "Austin");
SmartPointer<DataRow> dataRow =new DataRow();
dataRow->addChild("child1", childDataRow );
```

GetChildren

指定された関係から子の行を取得します。

構文

ASCII バージョン

```
list< SmartPointer<DataRow> > getChildren(const char* childName)
```

Unicode バージョン

```
list< SmartPointer<DataRow> > getChildren(const UnicodeString childName)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。

結果

指定された関係から子の行を返します。

例

```
list< SmartPointer<DataRow> > rowsChild2= dataRow-
>getChildren("child2");
```

ListChildNames

指定された親子関係のすべての名前を取得します。

構文

ASCII バージョン

```
list<string> listChildNames()
```

Unicode バージョン

```
list<UnicodeString> listChildNames()
```

結果

指定された親/子関係の名前セットを返します。

例

```
list<G1CLIENT_STRING> names = dataRow->listChildNames();
```

SetChildren

与えられた指定の親子関係の行を設定します。 この名前で行がすでに存在する場合は、呼び出し元に返されます。

構文

ASCII バージョン

```
list< SmartPointer<DataRow> > setChildren(const char* childName, list<
 SmartPointer<DataRow> > dataRows)
```

Unicode バージョン

```
list< SmartPointer<DataRow> > setChildren(const UnicodeString childName,
 list< SmartPointer<DataRow> > dataRows)
```

結果

指定された親/子関係の名前セットを返します。

例

```
SmartPointer<DataRow> dataRow1=new DataRow();
dataRow1->set("Address", "100 Congress");
dataRow1->set("City", "Austin");
SmartPointer<DataRow> dataRow2=new DataRow();
dataRow2->set("Address", "200 Congress");
dataRow2->set("City", "Austin");
list< SmartPointer<DataRow> > rows ;
rows.push_back(dataRow1);
rows.push_back(dataRow2);
list< SmartPointer<DataRow> > rowsNewChildren = dataRowSpt-
>setChildren("child1", rows);
```

4 - COM API

このセクションの構成

はじめに	123
Server	128
Service	130
Message	131
DataTable	139
DataRow	145
Map	153

はじめに

Component Object Model (COM) という語は、オブジェクト指向テクノロジーを基盤とするクライアント/サーバー アプリケーションのクロスプラットフォーム開発用のオープン アーキテクチャを指します。COM は、再利用可能なソフトウェア コンポーネントを作成する方法の 1 つです。クライアントは、オブジェクトに実装されたインターフェイスを通じてオブジェクトにアクセスします。つまり、オブジェクトはクライアントとサーバー間の通信媒体となります。Component Object Model によって、分散オブジェクト指向システムを柔軟に作成することができます。COM オブジェクトは言語に依存せず、バイナリ形式で出荷できます。また、既存の統合コードを変更することなくアップグレードでき、ネットワーク上に透過的に再配置できます。このような性質を持つため、COM オブジェクトは極めて柔軟性が高く、ほぼすべての Windows ベースのクライアント/サーバー システムに特定の機能を追加するのに適しています。

注： この章に示す例は、Visual Basic で書かれています。

Spectrum™ Technology Platform COM API は、次のインターフェイスで構成されます。

- サーバー
- サービス
- メッセージ
- DataTable
- DataRow
- マップ

定数

.COM API では、2 組の定数が使用されます。最初の 1 組は `Server` オブジェクト用です。以下の表に説明します。

表 19：`Server` コンポーネントの定数

定数名	説明/デフォルト	例
SERVER.HOST	サーバーのホスト名を表す文字列。デフォルトは "localhost" です。	65.89.200.89

定数名	説明/デフォルト	例
SERVER.PORT	サーバーのポートを表す文字列。デフォルトは "8080" です。	10119
SERVER.ACCOUNT_ID	サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1
SERVER.ACCOUNT_PASSWORD	サーバーのアカウント パスワードを表す文字列。デフォルト値はありません。	user1
SERVER.CONNECTION_TIMEOUT	サーバーの接続タイムアウトをミリ秒単位で表す文字列。デフォルトは "5000" です。	50000
SERVER.CONNECTION_TYPE	サーバーの接続タイプを表す文字列。現在は HTTP、HTTPS、または SOCKET のみがサポートされています。デフォルトは "HTTP" です。	HTTP(S)
SERVER.PROXY_HOST	プロキシ サーバーのホスト名を表す文字列。デフォルト値はありません。	192.168.1.77
SERVER.PROXY_PORT	プロキシ サーバーのポートを表す文字列。デフォルト値はありません。	8080
SERVER.PROXY_USER	プロキシ サーバーのアカウント ID を表す文字列。デフォルト値はありません。	user1
SERVER.PROXY_PASSWORD	プロキシ サーバーのアカウント パスワードを表す文字列。デフォルト値はありません。	user1

2 組目の定数は `Message` コンポーネント用です。

表 20 : `Message` コンポーネントの定数

定数名	説明	例
MESSAGE.CONTEXT_ACCOUNT_ID	メッセージ コンテキストのアカウント ID を表す文字列。	user1
MESSAGE.CONTEXT_ACCOUNT_PASSWORD	メッセージ コンテキストのアカウント パスワードを表す文字列。	user1
MESSAGE.CONTEXT_SERVICE_NAME	メッセージ コンテキストのサービス名を表す文字列。	echoservice

エラー メッセージ

COM API では、次のエラー メッセージが使用されます。

- 接続エラー メッセージ:
  - "Connection type not supported"
  - "Client timeout"
- DataTable 作成時のエラー メッセージ:
  - "Blank column name"
  - "Duplicated column name"
  - "The column index is invalid"
- Message Packaging 例外のエラー メッセージ:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect to Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

例:

```
On Error GoTo ErrorHandler
  Dim server As New G1CLIENTLib.server
  server.setConnectionProperty server.HOST, "localhost"
  server.setConnectionProperty server.Port, "8080"
  'Making connection to the server
  server.Connect
  ...
  Exit Sub
  ErrorHandler:
  MsgBox Err.Description
```

サンプル アプリケーション

以下のサンプル コードに、COM API の使い方を示します。

```
On Error GoTo ErrorHandler

 Dim server As New G1CLIENTLib.server
 Dim service As G1CLIENTLib.service
 Dim requestMsg As New G1CLIENTLib.Message
 Dim replyMsg As G1CLIENTLib.Message
 Dim dataTable As G1CLIENTLib.dataTable
 Dim newRow As G1CLIENTLib.dataRow
 Dim returnDataTable As G1CLIENTLib.dataTable
 Dim row As G1CLIENTLib.DataRow
 Dim sColumnNames() As String
 Dim sColumnName As String
 Dim sFieldValue As String
 Dim rows() As Variant
 Dim nRow As Integer
 Dim nColumn As Integer
 'Set server connection properties
 server.setConnectionProperty server.HOST, "localhost"
 server.setConnectionProperty server.Port, "10119"
 server.setConnectionProperty server.CONNECTION_TYPE, "SOCKET"
 server.setConnectionProperty server.ACCOUNT_ID, "guest"
 server.setConnectionProperty server.ACCOUNT_PASSWORD, ""

 'Connect to server
 server.Connect

 'Get the service from the server
 Set service = server.getService("ValidateAddress")

 'Fill DataTable in the input message
 Set dataTable = requestMsg.getDataTable
 dataTable.addColumn ("AddressLine1")
```

```
 dataTable.addColumn ("City")
 dataTable.addColumn ("StateProvince")

 Set newRow = dataTable.newRow
 newRow.setByIndex 0, "10535 Boyer"
 newRow.setByIndex 1, "Austin"
 newRow.setByIndex 2, "Texas"
 dataTable.addRow newRow

 'Set"option" Properties to the Input Message
 requestMsg.putOption "OutputCasing", "M"
 requestMsg.putOption "OutputRecordType", "A"

 'Process Input Message, return output Message
 Set replyMsg = service.process(requestMsg)

 'Disconnect from the server
 server.disconnect

 'Get the result from the resonse message
 Set returnDataTable = replyMsg.getDataTable
 ReDim rows(returnDataTable.getRowCount) As Variant

 rows = returnDataTable.iterator

 ReDim sColumnNames(returnDataTable.getColumnCount) As String
 sColumnNames = returnDataTable.getColumnNames

 For nRow = 0 To returnDataTable.getRowCount - 1
Set row = rows(nRow)

For nColumn = 0 To row.getColumnCount - 1
  sColumnName = sColumnNames(nColumn)
  sFieldValue = row.getByName(sColumnName)
Next

 Next

  Exit Sub

  ErrorHandler:

MsgBox Err.Description
```

Server

`Server` オブジェクトを使用して、サーバーへの接続、サーバーからの切断、およびサーバーからのサービスの取得を行います。

Connect

サーバーに接続します。HTTP またはソケットを介して接続できます。

> **注:** COM では、HTTP、HTTPS、またはソケット サーバー接続プロトコルを使用します。HTTP と HTTPS は、クライアント接続を論理的に確立するだけで、GetService メソッドまたは Process メソッドが呼び出されるまで実際にはサーバーに接続しません。ソケットプロトコルは、Connect が呼び出された時点でサーバーへの接続を確立します。

構文

```
Sub connect()
```

パラメータ

なし

結果

なし

Exception

接続タイプがサポートされていません。

例

```
Dim server As New G1CLIENTLib.server
server.connect
```

Disconnect

サーバーから切断します。

構文

```
Sub disconnect()
```

パラメータ

なし

結果

なし

例

```
Dim server As New G1CLIENTLib.server
server.disconnect
```

GetService

サーバーからサービス (ValidateAddress など) を取得します。

構文

```
Function getService(serviceName As String) As Service
```

パラメータ

- serviceName - クライアントが要求するサービスの名前。

結果

要求したサービス、またはサービスが存在しない場合は NULL。

Exceptions

- ERROR_FAIL_TO_GET_SERVICE — サーバーへの接続がない場合。

例

```
Dim server As New G1CLIENTLib.server
Dim service As G1CLIENTLib.service
...
'get the service from the server
Set service = server.getService("ValidateAddress")
```

SetConnectionProperty

ホスト名、タイアウト時間など、サーバー接続設定プロパティを設定します。

構文

```
Sub setConnectionProperty(name As String, value As String)
```

パラメータ

- Name — 接続プロパティの名前。HOST など。
- Value — 接続プロパティの値。"www.myhost.com" など。

結果

リターン コード — なし

例外

- ERROR-INVALID-COLUMN_NAME — 列名が空または NULL。
- ERROR_INVALID_VALUE — 値が NULL。

例

```
set connection properties
Dim server As New G1CLIENTLib.server

server.setConnectionProperty server.HOST,"localhost"
server.setConnectionProperty server.PORT, "8080"
```

Service

`Service` はサービスを呼び出し、送信メッセージを処理します (言い換えると入力メッセージを送信し、応答を受信します)。

Process

入力メッセージを処理し、サーバーから応答メッセージを取得します。

構文

```
Function process(IRequest As Message) As Message
```

パラメータ

- iRequest — "オプション" 設定と DataTable が格納された入力メッセージ オブジェクト。

結果

要求に対する応答メッセージを返します。

例外:

- ERROR_NULL_INPUT_MESSAGE — 要求メッセージが NULL です。

例

```
Dim service As New G1CLIENTLib.service
Dim replyMsg As G1CLIENTLib.Message
...
'Process the message and return back the response message
Set replyMsg = service.process(requestMsg)
```

Message

Message は、入力データを送信し、出力データをサービスから受信するのに使用されます。Message のプロパティには、"コンテキスト" エンティティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、"オプション" エンティティ (サービス固有の実行時プロパティ)、"エラー" エンティティ (エラー クラス、エラー メッセージ、エラー スタックトレース) などがあります。

GetContext

メッセージのコンテキスト セクションに指定された名前のコンテキスト エンティティから値を取得します。 "コンテキスト" エンティティには、アカウント ID、アカウント パスワード、サービス名、サービス メソッドなどがあります。

構文

```
Function getContext(name As String) As String
```

パラメータ

- Name — 関連付けられた値を返す名前。

結果

String — 指定されたエンティティの値、または指定されたエンティティが存在しない場合は空の文字列。

例

```
Dim msg As New G1CLIENTLib.Message
Dim accountID As String

accountID = msg.getContext(msg.CONTEXT_ACCOUNT_ID)
```

GetContextMap

すべてのコンテキスト エントリが含まれるマップを取得します。

構文

```
Function getContextMap() As Map
```

パラメータ

- なし

結果

すべてのコンテキスト エントリが含まれるマップを返します。

例

```
Dim map As G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message
Dim sKey As String
Dim sValue As String

requestMsg.putContext
 requestMsg.CONTEXT_ACCOUNT_ID, "admin"
```

```
requestMsg.putContext
 requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

PutContext

指定された名前に基づいてコンテキスト プロパティに値を設定します。 指定された名前のエンティティに既存の値がある場合、その値は上書きされます。 "コンテキスト" プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス メソッドなどがあります。

構文

```
Sub putContext(name As String, value As String)
```

パラメータ

- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

結果

なし

例

```
Dim requestMsg As New G1CLIENTLib.Message

requestMsg.putContext
 requestMsg.CONTEXT_ACOUNT_ID, "admin"
requestMsg.putContext
 requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"
```

PutContextMap

新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。

構文

```
Sub putContextMap(context As Map)
```

パラメータ

- 現在のコンテキスト マップに追加する新しいコンテキスト マップ。

結果

なし

例

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1UBCAPICOMLib.Message

map.Insert requestMsg.CONTEXT_ACCOUNT_ID, "admin"
map.Insert requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"

requestMsg.putContextMap map
```

SetContextMap

新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。

構文

```
Sub setContextMap(context As Map)
```

パラメータ

- 現在のコンテキスト マップを置き換える新しいコンテキスト マップ。

結果

なし

例

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1UBCAPICOMLib.Message

map.Insert requestMsg.CONTEXT_ACCOUNT_ID, "admin"
map.Insert requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"
```

```
requestMsg.setContextMap map
```

GetOption

メッセージのオプション セクションに指定された名前のオプション エンティティから値を取得します。オプション エンティティには、出力の大文字と小文字の区別、出力データのフォーマットなど、サービス固有の実行時オプションが含まれます。

構文

```
Function getOption(name As String) As String
```

パラメータ

- Name — 関連付けられた値を返す名前。

結果

- String — 指定されたエンティティの値、または指定されたエンティティが存在しない場合は空の文字列。

例

```
Dim msg As New G1CLIENTLib.Message
Dim optionValue As String

OptionValue = msg.getOption("OutputCasing")
```

GetOptions

すべてのオプション エントリが含まれるマップを取得します。

構文

```
Function getOptions() As Map
```

パラメータ

- なし

結果

すべてのオプション エントリが含まれるマップを返します。

例

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message
Dim sKey As String
Dim sValue As String

requestMsg.putOption "OutputCasing", "M"
requestMsg.putOption "OutputRecordType", "A"

Set map = requestMsg.getOptions

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

PutOption

指定された名前に基づいてオプション プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。"オプション" プロパティはサービス固有の実行時オプションです。

構文

```
Sub putOption(name As String, value As String)
```

パラメータ

- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

例

```
Dim requestMsg As New G1CLIENTLib.Message

requestMsg.putOption "OutputCasing", "M"
requestMsg.putOption "OutputRecordType", "A"
```

PutOptions

新しいオプション プロパティを現在のオプション プロパティに追加します。

構文

```
Sub putOptions(options As Map)
```

パラメータ

- 現在のオプション プロパティに追加する新しいオプション マップ。

例

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message

map.Insert "OutputCasing", "M"
map.Insert "OutputRecordType", "A"

requestMsg.putOptions map
```

SetOptions

新しいオプション プロパティで現在のオプション プロパティを上書きします。

構文

```
Sub setOptions(options As Map)
```

パラメータ

- 現在のオプション マップを置き換える新しいオプション マップ。

例

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message

map.Insert "OutputCasing", "M"
map.Insert "OutputRecordType", "A"
```

```
requestMsg.setOptions map
```

GetError

エラー メッセージからエラーを取得します。

構文

```
Function getError() As String
```

パラメータ

- なし

結果

メッセージからエラー メッセージを取得して返します。

例

```
Dim sErrorMessage As String
...
sErrorMessage = replyMsg.getError()
```

GetDataTable

DataTable をメッセージから取得します。

構文

```
Function getDataTable() As DataTable
```

パラメータ

- なし

例

```
Dim DataTable AS G1CLIENTLib.dataTable

Set DataTable = message.getDataTable
```

DataTable

DataTableには入出力データのレコードが含まれます。このオブジェクトに関連付けられたメソッドを使って、出力用の列名を定義し、行をDataTableに追加します。

AddColumn

新しい列をDataTableに追加します。

構文

```
Function addColumn(columnName As String) As Integer
```

パラメータ

- 列名

結果

列のインデックスを返します。

例外

- 列名が空白です。
- 同名の列がすでにあります。

例

```
Dim dataTable As G1CLIENTLib.dataTable

dataTable.addcolumn "AddressLine1"
dataTable.addcolumn "City"
```

GetColumnNames

すべての列名を取得します。

構文

```
Syntax Function getColumnNames() As String()
```

パラメータ

- なし

結果

列名の配列を返します。

例

```
Dim sColumnNames() As String
Dim sColumnName As String
Dim nColumn As Integer

ReDim sColumnNames(returnDataTable.getColumnCount) As String
sColumnNames = returnDataTable.getColumnNames

For nColumn = 0 To dataRow.getColumnCount - 1
sColumnName = sColumnNames (nColumn)
Next
```

GetColumnIndex

対応する列インデックスを取得します。

構文

```
Function getColumnIndex(columnName As String) As Integer
```

パラメータ

- 列名

結果

対応する列インデックスを返します。

例

```
Dim nIndex As Integer
nIndex = dataTable.getColumnIndex("AddressLine1")
```

GetColumnCount

DataTable にある列の数を取得します。

構文

```
Function getColumnCount() As Integer
```

パラメータ

- なし

結果

列の数を返します。

例

```
Dim nColumnCount As Integer
nColumnCount = dataTable.getColumnCount()
```

Clear

DataTable のデータを消去します。

構文

```
Sub clear()
```

パラメータ

- なし

例

```
dataTable.clear()
```

Iterator

DataTable 内のすべての DataRow を含むイテレータです。

構文

```
Syntax Function iterator() As DataRow()
```

パラメータ

- なし

結果

DataTable 内のすべての DataRow を含むイテレータを返します。

例

```
Dim returnDataTable As G1CLIENTLib.dataTable
  Dim row As G1CLIENTLib.DataRow
  Dim sColumnName As String
  Dim sFieldValue As String
  Dim rows() As Variant
  Dim nRow As Integer
  Dim nColumn As Integer

  'Get the result from the resonse message
  Set returnDataTable = replyMsg.getDataTable
  ReDim rows(returnDataTable.getRowCount) As Variant

  rows = returnDataTable.iterator

  For nRow = 0 To returnDataTable.getRowCount - 1
 Set row = rows(nRow)

 For nColumn = 0 To row.getColumnCount - 1
sColumnName = row.getColumnNames(nColumn)
sFieldValue = row.getByName(sColumnName)
 Next

 Next
```

AddRow

DataRow を DataTable に追加します。

構文

```
Sub addRow(DataRow As DataRow)
```

パラメータ

- DataTable に追加する DataRow。

結果

なし

例

```
Dim dataTable As G1CLIENTLib.dataTable
Dim newRow As G1CLIENTlib.DataRow

Set dataTable=requestMsg.getDataTable
dataTable.addColumn("AddressLine1")
dataTable.addColumn("City")
dataTable.addColumn("State")
Set newRow=dataTable.newRow
newRow.setByIndex 0, "10535 Boyer"
newRow.setByIndex 1, "Austin"
newRow.setByIndex 2, "Texas"
dataTable.addRow newRow
```

NewRow

新しい DataRow を DataTable 内に作成します。

構文

```
Function newRow() As DataRow
```

パラメータ

- なし

結果

新規作成された DataRow を返します。

例

```
Dim dataTable As G1CLIENTLib.dataTable
Dim newRow As G1CLIENTlib.DataRow

Set dataTable=requestMsg.getDataTable

Set newRow=dataTable.newRow
newRow.setByName "AddressLine1","10535 Boyer"
newRow.setByName "City", "Austin"
newRow.setByName "State", "Texas"
dataTable.addRow newRow
```

GetRowCount

DataTable にある DataRow の数を取得します。

構文

```
Function getRowCount() As Integer
```

パラメータ

- なし

結果

DataTable にある DataRow の数を返します。

例

```
Dim nRowCount As Integer
nRowCount = dataTable.getRowCount
```

Merge

指定された DataTable と現在の DataTable を結合します。

構文

```
Sub merge(other As DataTable)
```

パラメータ

- 現在の DataTable と結合する他の DataTable。

結果

なし

例

```
Dim otherDataTable As New G1CLIENTlib.dataTable
...
dataTable.merge(otherDataTable)
```

DataRow

DataRow には入出力データの個々のレコードが含まれます。このクラスに関連付けられたメソッドを使って、出力用の列名を定義しレコードを DataTable に追加します。

GetColumnNames

すべての列名を取得します。

構文

```
Function getColumnNames() As String()
```

パラメータ

- なし

結果

列名の配列を返します。

例

```
Dim sColumnNames() As String
Dim sColumnName As String
Dim nColumn As Integer
ReDim sColumnNames(dataRow.getColumnCount) As String
sColumnName = sColumnNames(nColumn)
For nColumn = 0 To dataRow.getColumnCount -1
sColumnName = sColumnNames(nColumn)
Next
```

GetColumnIndex

対応する列インデックスを取得します。

構文

```
Function getColumnIndex(columnName As String) As Integer
```

パラメータ

- 列名

結果

対応する列インデックスを返します。

例

```
Dim nIndex As Integer
nIndex = dataRow.getColumnIndex("AddressLine1")
```

GetColumnCount

DataRow にある列の数を取得します。

構文

```
Function getColumnCount() As Integer
```

パラメータ

- なし

結果

列の数を返します。

例

```
Dim nColumnCount As Integer
nColumnCount = dataRow.getColumnCount()
```

GetByIndex

この DataRow の列インデックスによってフィールド配列から値を取得します。

構文

```
Function getByIndex(index As Integer) As String
```

パラメータ

- 指定された値を関連付けるインデックス。

結果

この DataRow の列インデックスの値を返します。 インデックスが無効な場合は空の文字列を返します。

例

```
Dim sValue As String
sValue = dataRow.getByIndex(1)
```

GetByName

この DataRow の列名によってフィールド配列から値を取得します。

構文

```
Function getByName(columnName As String) As String
```

パラメータ

- 指定された値を関連付ける名前。

結果

この DataRow の列名の値を返します。列名が存在しない場合は空の文字列を返します。

例

```
Dim sValue As String
sValue = dataRow.getByName("City")
```

Merge

指定された DataRow と現在の DataRow を結合します。

構文

```
Sub merge(other As DataRow)
```

パラメータ

- 現在の DataRow と結合する他の DataRow。

結果

なし

例

```
Dim otherDataRow As New G1CLIENTlib.DataRow
...
dataRow.merge(otherDataRow)
```

SetByName

DataRowの対応する列の値を設定します。 この名前の値が存在する場合は、古い値を置き換えます。

構文

```
Sub setByName(columnName As String, value As String)
```

パラメータ

- 指定された値を関連付ける名前。
- 指定された名前に関連付ける値。

結果

なし

例外

- 列名が空白です。
- 同名の列がすでにあります。

例

```
Dim newRow As G1CLIENTLib.DataRow
Set newRow= dataTable.netRow
newRow.setByName "AddressLine1", "100 Congress"
newRow.setByName "City", "Austin"
newRow.setByName "State", "Texas"
dataTable.addRow newRow
```

SetByIndex

DataRowの対応する列の値を設定します。 この名前の値が存在する場合は、古い値を置き換えます。

構文

```
Sub setByIndex(index As Integer, value As String)
```

パラメータ

- 指定された値を関連付ける列インデックス。
- 指定された名前に関連付ける値。

結果

なし

例外

- 列インデックスが無効です。

例

```
Dim newRow As G1CLIENTLib.DataRow
Set newRow= dataTable.netRow
newRow.setByIndex 0, "100 Congress"
newRow.setByIndex 1, "Austin"
newRow.setByIndex 2, "Texas"
dataTable.addRow newRow
```

AddChild

新しい DataRow を指定された親子関係に追加します。 指定された関係が存在する場合、与えられた DataRow は既存の DataRow コレクションに追加されます。 存在しない場合、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。

構文

```
Sub addChild( childName As String, childDataRow As DataRow)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。
- 関係に追加する DataRow。

結果

なし

例

```
Dim dataRow As New G1CLIENTLib.dataRow
Dim childDataRow As New G1CLIENTLib.dataRow

childDataRow .setByName "Address", "100 Congress"
childDataRow .setByName "City", "Austin"

dataRow.addChild "child1", dataRow
```

GetChildren

指定された関係から子の行を取得します。

構文

Function getChildren(childName As String) As DataRow()

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。

結果

指定された関係から子の行を返します。

例

```
Dim dataRow As New G1CLIENTLib.dataRow
' Assume that dataRow has children ......
' Or more code to be needed
Dim rowsChild1() As Variant
rowsChild1 = dataRow.getChildren("child1")
```

ListChildNames

指定された親子関係のすべての名前を取得します。

構文

```
Function listChildNames() As String()
```

パラメータ

なし

結果

指定された親/子関係の名前セットを返します。

例

```
Dim dataRow As New G1CLIENTLib.dataRow
' Assume that dataRow has children ......
' Or more code to be needed
Dim sChildNames() As String
sChildNames = dataRow.listChildNames
```

SetChildren

与えられた指定の親子関係の行を設定します。 この名前で行がすでに存在する場合は、呼び出し元に返されます。

構文

```
Function setChildren(childName As String, DataRows As DataRow()) As
DataRow()
```

パラメータ

なし

結果

指定された親/子関係の名前セットを返します。

例

```
Dim dataRow1 As New G1CLIENTLib.dataRow
Dim dataRow2 As New G1CLIENTLib.dataRow
dataRow1.setByName "Address", "100 Congress"
dataRow1.setByName "City", "Austin"
dataRow2.setByName "Address", "200 Congress"
dataRow2.setByName "City", "Austin"

Dim rows(1) As G1CLIENTLib.dataRow

Set rows(0) = dataRow1
Set rows(1) = dataRow2

Dim newRows() As Variant
newRows = dataRowSpt.setChildren("child1", rows())
```

Map

Map はキーを値にマップするオブジェクトです。マップに重複キーを含めることはできません。各キーは 1 つの値にのみマップすることができます。

Reset

カーソルを最初のマップの前にセットします。

構文

Sub Reset()

パラメータ

- なし

結果

なし

例

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

Next

カーソルを現在の位置から 1 マップ下に移動します。

構文

```
Sub Next()
```

パラメータ

- なし

例

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

GetKey

現在のマップのキーを取得します。

構文

```
Function getKey() As String
```

パラメータ

- なし

結果

現在のマップのキーを返します。

例

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

GetValue

現在のマップの値を取得します。

構文

```
Function getValue() As String
```

パラメータ

- なし

結果

現在のマップの値を返します。

例

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"
```

```
Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

5 - Java API

このセクションの構成

はじめに	158
Server	163
Service	169
Message	171
DataTable	177
DataRow	183

はじめに

Java クラスは、あるタイプのすべてのオブジェクトに共通する変数およびメソッドを定義する設計図またはプロトタイプです。 Java クラスは、特定の種類のオブジェクトの実装も定義します。これらのクラスを使用して、Java アプリケーションを作成することができます。 一般的に、Java オブジェクトは Java クラスから作成されます。

Java オブジェクトは関連する変数およびメソッドのコレクションで、Java Virtual Machine (JVM) を利用して Java 言語で書かれています。 クラスまたはオブジェクトと関連付けられたデータは、変数に格納されます。 クラスまたはオブジェクトと関連付けられた動作は、メソッドを使用して実行されます。 メソッドは、C などの手続き言語における関数またはプロシージャに似ています。

Java ソフトウェア オブジェクトは、メッセージを使用して互いにやりとりしたり、通信を行います。 受信側のオブジェクトがタスクの実行に必要とすることがあるその他の情報は、パラメータによって渡されます。

Java テクノロジーの詳細については、**www.oracle.com/java** を参照してください。

定数

Java API では、2 組の定数が使用されます。最初の 1 組は `Server` コンポーネント用です。以下の表に説明します。

表 21 : `Server` コンポーネントの定数

定数名	説明	例
Server.HOST	サーバーのホスト名を表す文字列。デフォルトは "localhost" です。	65.89.200.89
Server.PORT	サーバーのポートを表す文字列。デフォルトは "8080" です。	10119

定数名	説明	例
Server.ACCOUNT_ID	サーバーのアカウント ID を表す文字列。デフォルトは NULL です。	user1
Server.ACCOUNT_PASSWORD	サーバーのアカウント パスワードを表す文字列。デフォルトは NULL です。	user1
Server.CONNECTION_TIMEOUT	サーバーの接続タイムアウトをミリ秒単位で表す文字列。デフォルトは "10000" です。	50000
Server.CONNECTION_TYPE	サーバーの接続タイプを表す文字列。現在は HTTP、HTTPS、または SOCKET のみがサポートされています。デフォルトは "HTTP" です。	HTTP
Server.PROXY_HOST	プロキシ サーバーのホスト名を表す文字列。デフォルトは NULL です。	192.168.1.77
Server.PROXY_PORT	プロキシ サーバーのポートを表す文字列。デフォルトは NULL です。	8080
Server.PROXY_USER	プロキシ サーバーのアカウント ID を表す文字列。デフォルトは NULL です。	user1
Server.PROXY_PASSWORD	プロキシ サーバーのアカウント パスワードを表す文字列。デフォルトは NULL です。	user1

定数名	説明	例
Server.INPUT_CLEANUP	入力データの中の特殊文字を削除する必要があるかどうかを示す boolean 値。デフォルトは false です。 **注 :** この属性が false に設定されている場合に、入力データに特殊文字が含まれていると、例外が発生します。 **重要 :** 入力データに特殊文字が存在することがわかっている場合のみ、これに true を設定してください。それ以外の場合にこの属性を true にすると、パフォーマンスが低下します。	true

2 組目の定数は `Message` コンポーネント用です。

表 22 : `Message` コンポーネントの定数

定数名	説明/デフォルト	例
Message.CONTEXT_ACCOUNT_ID	メッセージ コンテキストのアカウント ID を表す文字列。	user1
Message.CONTEXT_ACCOUNT_PASSWORD	メッセージ コンテキストのアカウント パスワードを表す文字列。	user1
Message.CONTEXT_SERVICE_NAME	メッセージ コンテキストのサービス名を表す文字列。	echoservice
Message.CONTEXT_SPECTRUM_DISPLAY_VERSION	メッセージ コンテキストの Spectrum 表示バージョンを表す文字列。	12.1
Message.CONTEXT_SPECTRUM_SERVER_VERSION	メッセージ コンテキストの Spectrum サーバー バージョンを表す文字列	12.1

エラー メッセージ

Java API では、次のエラー メッセージが使用されます。

- 接続エラー メッセージ
  - "Connection type not supported."
  - "Client timeout"
- DataTable 作成時のエラー メッセージ:
  - "Blank column name"
  - "Duplicated column name"
  - "Index is out of bounds"
- Message Packaging 例外のエラー メッセージ:
  - "Cannot pack null Message"
  - "Input Message is null"
  - "Unable to connect to Server:"
  - "Failed to get Service"
  - "Unknown serialization type:"
  - "Unknown encoding type:"
  - "Gateway is not connected" (for SOCKET)

サンプル アプリケーション

以下のサンプル コードに、Java API の使い方を示します。

```
try
{

    // Create Server
    Server server = new Server();

    // Set server connection properties
    server.setConnectionProperty(Server.HOST, "localhost");
    server.setConnectionProperty(Server.PORT, "10119");
    server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
    server.setConnectionProperty(Server.ACCOUNT_ID, "guest");
    server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "");

    // Connect to server
    server.connect();
```

```
    // Get Service From Server
    Service service = server.getService("ValidateAddress");

    // Create Input Message
    Message request = new Message();

    // Fill DataTable in the input message
    DataTable dataTable = request.getDataTable();
    DataRow row1 = dataTable.newRow();
    row1.set("AddressLine1", "4200 Parliament Place");
    row1.set("City", "Lanham");
    row1.set("StateProvince", "Maryland");
    dataTable.addRow(row1);
    DataRow row2 = dataTable.newRow();
    row2.set("AddressLine1", "100 Congress");
    row2.set("City", "Austin");
    row2.set("StateProvince", "Texas");
    dataTable.addRow(row2);

    // Set"option" Properties to the Input
    Message request.putOption("OutputCasing", "M");
    request.putOption("OutputRecordType", "A");

    // Process Input Message, return output Message
    Message reply = service.process(request);

    // Disconnect from server
    server.disconnect();

    // Get the result from the response message
    DataTable returnDataTable = reply.getDataTable();
    String[] columnNames = returnDataTable.getColumnNames();
    Iterator iter = returnDataTable.iterator();
    while (iter.hasNext())
    {
        DataRow row = (DataRow) iter.next();
        for (int col = 0; col & lt; returnDataTable.getColumnCount();
col++)
        {
            String value = row.get(columnNames[col]);
            System.out.println(value);
        }
    }
}
catch (Exception e)
{
```

```
    System.out.println("Error Occurred, " + e.getMessage());
}
```

Server

`Server` クラスは、サーバーへの接続、サーバーからの切断、およびサーバーからのサービスの取得に使用されます。

Connect

プロパティを読み取って、使用するゲートウェイ接続を決定し、サーバーへの接続を確立します。HTTP、HTTPS、またはソケットを介して接続できます。ただし、HTTP と HTTPS は GetService メソッドまたは Process メソッドが呼び出されるまで実際にはサーバーに接続しません。ソケット接続タイプの使用時は、`Connect`メソッドは完全に機能します。

構文

```
public void connect()
```

パラメータ

なし

結果

例外:

- ConfigurationException: 無効な設定が原因で、サーバーに接続できません。例えば、不明なプロトコルは ConfigurationException を発生させます。このエラーが発生した場合は connect() を再実行しても無意味です。
- ConnectionException: サーバーに接続できないときに発生します。例外の根本的な原因によっては、再実行で接続に成功する可能性があります。
- MessageProcessingException: 設定や接続の問題ではない原因でサーバー側にエラーが発生したことを意味します。

例

```
Server server = new Server();

server.setConnectionProperty(Server.HOST, "localhost");
server.setConnectionProperty(Server.PORT, "10119");
server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
server.setConnectionProperty(Server.ACCOUNT_ID, "guest");
server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "");

try
{
 //Connect to server
 server.connect();
}
catch (ConfgurationException e)
{
 // indicate an error with configuration
}
catch (ConnectionException e)
{
 // handle connection issue (retry, report error, etc.)
}
catch (MessageProcessingException e)
{
 // report error
}
```

コネクション プーリング

ソケット接続タイプのコネクション プーリングが、Java クライアントで利用できます。ここでは、コネクション プーリングを有効または無効にする手順を説明します。デフォルトで、コネクション プーリングは無効です。

コネクション プーリングを有効にする:

```
Server server = new Server();
Server.setConnectionProperty(Connection.SOCKET_POOL, "true");
```

コネクション プーリングを無効にする:

```
Server server = new Server();
Server.setConnectionProperty(Connection.SOCKET_POOL, "false");
```

コネクション プーリングを有効にすると、connect() メソッドはプールから接続を借用し、disconnect() メソッドは接続をプールに返却します。プーリングの使用時は、接続をプールに返すたびに必ず disconnect() を呼び出します。

各スレッドは、独自のサーバーを保持する必要があります。次に、使用例を示します。

```
{
 ...
 Server server = new Server();
 server.setConnectionProperty(Server.HOST, "localhost");
 server.setConnectionProperty(Server.PORT, "10119");
 server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
 server.setConnectionProperty(Server.ACCOUNT_ID, "yourID");
 server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "pwd");
 server.setConnectionProperty(Connection.SOCKET_POOL, "true");
 server.setConnectionProperty(Connection.SOCKET_POOL_MAX_ACTIVE, "20");

 server.setConnectionProperty(Connection.SOCKET_POOL_MIN_IDLE, "10");
 server.setConnectionProperty(Connection.SOCKET_POOL_MAX_TOTAL, "25");

 server.connect();
 ...
 service = server.getService(serviceName);
 reply = service.process(requestMessage);
 server.disconnect();
 ...
}
```

以下の表に、コネクション プーリングに使用できる定数を示します。

表 23 : コネクション プーリングの定数

定数名	説明
SOCKET_POOL	ソケット接続タイプの使用時にコネクション プーリングを使うかどうかを示します。値は True または False です。デフォルトは false です。
SOCKET_POOL_MAX_ACTIVE*	プールから借用できるアクティブなソケット接続の最大数。デフォルトは -1 です。この値は最大数がないことを意味します。
SOCKET_POOL_MAX_IDLE*	プールに残っているアイドル状態のソケット接続の最大数。デフォルトは -1 です。この値は最大数がないことを意味します。

定数名	説明
SOCKET_POOL_MAX_TOTAL*	プールに存在するソケット接続の最大数の合計(アクティブな接続とアイドル状態の接続の両方)。デフォルトは-1です。この値は最大数がないことを意味します。
SOCKET_POOL_MAX_WAIT*	"when exhausted" アクションがWHEN_EXHAUSTED_BLOCKに設定されている場合に、プールが空になってから例外をスローするまでの最大待機時間(ミリ秒)。デフォルトは -1 です。この値は最大数がないことを意味します。
SOCKET_POOL_MIN_EVICTABLE_IDLE_TIME_MILLIS*	借用の可能な状態になるまで接続をアイドルさせる最小時間。デフォルトは 1800000 (30分) です。
SOCKET_POOL_MIN_IDLE*	evictor スレッド (アクティブな状態にある場合)が新しい接続を作成するまでにプールに存在できる接続の最小数。デフォルトは0です。
SOCKET_POOL_NUM_TESTS_PER_EVICTION_RUN*	evictor スレッド (アクティブな状態にある場合)の実行時にチェックされるアイドル接続の数を設定します。デフォルトは -1 です。この値は、すべてのアイドル接続をチェックすることを意味します。
SOCKET_POOL_TEST_ON_BORROW*	プールから借用される前に接続を検証するかどうかを示します。デフォルトは true です。
SOCKET_POOL_TEST_ON_RETURN*	プールに返却される前に接続を検証するかどうかを示します。デフォルトは false です。
SOCKET_POOL_TEST_WHILE_IDLE*	アイドル接続をプールから退出させるスレッドによって接続を検証するかどうかを示します。デフォルトは false です。

定数名	説明
SOCKET_POOL_TIME_BETWEEN_EVICTION_RUNS_MILLIS*	アイドル接続をプールから退出させるスレッドの実行間隔 (スリープ時間) をミリ秒で設定します。0 または負の値に設定すると、アイドル接続を退出させるスレッドは実行されません。デフォルトは 300000 (5 分) です。
SOCKET_POOL_WHEN_EXHAUSTED_ACTION*	接続を借用しようとしたが使用可能な接続がなかった場合に実行する "when exhausted action" を設定します。デフォルトは SOCKET_POOL_WHEN_EXHAUSTED_BLOCK です。
SOCKET_POOL_WHEN_EXHAUSTED_BLOCK*	接続を借用しようとしたが使用可能な接続がなかった場合に、新しいオブジェクトが使用可能になるか最大待機時間が経過するまで、呼び出し元がブロックすることを指定する "when exhausted action" タイプ。
SOCKET_POOL_WHEN_EXHAUSTED_FAIL*	接続を借用しようとしたが使用可能な接続がなかった場合に、呼び出し元が失敗し、ConnectionException をスローすることを示す "when exhausted action" タイプ。
SOCKET_POOL_WHEN_EXHAUSTED_GROW*	接続を借用しようとしたが使用可能な接続がなかった場合に、新しい接続が作成されることを示す "when exhausted action" タイプ。

* ソケット接続タイプを使用し、コネクション プーリングが有効な場合のみ適用できます。

Disconnect

サーバーから切断します。

構文

```
public void disconnect()
```

パラメータ

なし

結果

クライアントがサーバーから切断されます。

例

```
...
//Disconnect from server
server.disconnect();
```

SetConnectionProperty

ホスト名、タイアウト時間など、サーバー接続設定プロパティを設定します。

構文

```
public void setConnectionProperty(String name, String value)
```

パラメータ

- Name — 接続プロパティの名前。HOST など。
- Value — 接続プロパティの値。"www.myhost.com" など。

結果

なし

Exceptions

- ERROR-INVALID-COLUMN_NAME — 列名が空または NULL。
- ERROR_INVALID_VALUE — 値が NULL。

例

```
Server server = new Server();
```

```
server.setConnectionProperty(Server.HOST, "localhost");
server.setConnectionProperty(Server.PORT, "8080");

//Connect to server
server.connect();
```

GetService

サーバーからサービスを取得します。

構文

```
public Service getService(String serviceName)
```

パラメータ

- Name - サービスの名前。

結果

特定のサービスを返します。

Exceptions

ServiceNotFoundException、ServiceCreationException をスローします。

例

```
Service service = server.getService("ValidateAddress");
```

Service

`Service` クラスは、メッセージを処理するために使用されます (より具体的に言えば、メッセージをサーバーに送信し、サーバーから応答を受信するために使用されます)。

Process

入力メッセージを処理し、応答メッセージを返します。

構文

```
public Message process (Message message)
```

パラメータ

- 入力メッセージ

結果

応答メッセージを返します。

Exceptions

- TimeoutException: 無効な設定が原因で、サーバーに接続できません。例えば、不明なプロトコルは ConfigurationException を発生させます。このエラーが発生した場合は connect() を再実行しても無意味です。
- ConnectionException: サーバーに接続できないときに発生します。例外の根本的な原因によっては、再実行で接続に成功する可能性があります。
- MessageProcessingException: 設定や接続の問題ではない原因でサーバー側にエラーが発生したことを意味します。

例

```
try
{
 //Process Input Message, return output Message
 Message response = service.process(message);
}
catch (ConnectionException e)
{
 // handle connection issue (retry, report error, etc.)
}
catch (TimeoutException e)
{
 // handle timeout issue (retry, report error, etc.)
}
catch (MessageProcessingException e)
{
 // report error
}
```

Message

Message クラスは、入力データを送信し、サービスから出力データを受け取ります。Message のプロパティには、コンテキスト プロパティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、オプション プロパティ (サービス固有の実行時プロパティ) などがあります。

GetContext

"コンテキスト" プロパティの名前で値を取得します。 コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

```
public String getContext(String name)
```

パラメータ

- Name - 関連付けられた値を返す名前。

結果

コンテキスト プロパティの名前に関連付けられた値を返します。 名前が存在しない場合、NULL を返します。

例

```
String value = message.getContext(Message.CONTEXT_ACCOUNT_ID);
```

GetContext

すべてのコンテキスト エントリが含まれるマップを取得します。

構文

```
public Map getContext()
```

パラメータ

- なし

結果

すべてのコンテキスト エントリが含まれるマップを返します。

例

```
Map context = message.getContext();
```

PutContext

指定された名前に基づいてコンテキスト プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

```
public void putContext(String name, String value)
```

パラメータ

- Name - 指定された値を関連付ける名前。
- Value - 指定された名前に関連付ける値。

結果

なし

例

```
message.putContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

PutContext

新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。

構文

```
public void putContext(Map map)
```

パラメータ

- 現在のコンテキスト ハッシュテーブルに追加する新しいコンテキスト ハッシュテーブル。

結果

なし

例

```
Map context = new HashMap();
...
message.putContext(context);
```

SetContext

新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。

構文

```
public void setContext(Map map)
```

パラメータ

- 現在のコンテキスト マップを置き換える新しいコンテキスト マップ。

結果

なし

例

```
Map context = new Map ();
...
message.setContext(context);
```

GetOption

オプション プロパティの名前で値を取得します。 オプション プロパティはサービス固有の実行時オプションです。

構文

```
public String getOption(String name)
```

パラメータ

- Name - 関連付けられた値を返す名前。

結果

メッセージの オプション プロパティ内の名前の値を返します。または、その名前が存在しない場合は NULL を返します。

例

```
String value = message.getOption("OutputCasing");
```

GetOptions

すべてのオプション エントリが含まれるマップを取得します。

構文

```
public Map getOptions();
```

パラメータ

- なし

結果

すべてのオプション エントリが含まれるマップを返します。

例

```
Map options = message.getOptions();
```

PutOption

指定された名前に基づいてオプション プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。オプション プロパティはサービス固有の実行時オプションです。

構文

```
public void setOption(String name, String value)
```

パラメータ

- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

結果

なし

例

```
message.setOption("OutputCasing", "M");
```

PutOptions

新しいオプション プロパティを現在のオプション プロパティに追加します。

構文

```
public void putOptions(Map map)
```

パラメータ

- 現在のオプション プロパティに追加する新しいオプション マップ。

例

```
Map options = new HashMap();
...
message.putOptions(options);
```

SetOptions

新しいオプション プロパティで現在のオプション プロパティを上書きします。

構文

```
public void setOptions(Map map)
```

パラメータ

- 現在のオプション マップを置き換える新しいオプション マップ。

結果

なし

例

```
Map options = new HashMap();
...
message.setOptions(options);
```

GetError

メッセージからエラー メッセージを取得します。

構文

```
public String getError()
```

パラメータ

- なし

結果

メッセージからエラー メッセージを取得して返します。

例

```
String error = message.getError();
```

GetDataTable

DataTable をメッセージから取得します。

構文

```
public DataTable getDataTable()
```

パラメータ

なし

結果

なし

例

```
DataTable dataTable = message.getDataTable();
```

DataTable

`DataTable` には入出力データのレコードが含まれます。このクラスに関連付けられたメソッドを使って、出力用の列名を定義しレコードを `DataTable` に追加します。

AddColumn

新しい列を DataTable に追加します。

構文

```
public int addColumn(String columnName)
```

パラメータ

- columnName

結果

列のインデックスを返します。

例

```
DataTable dataTable = message.getDataTable();
int columnIndex = dataTable.addColumn("AddressLine1");
columnIndex = dataTable.addColumn("City")
```

GetColumnNames

すべての列名を取得します。

構文

```
public String[] getColumnNames()
```

パラメータ

- なし

結果

列名の文字列配列を返します。

例

```
String[] columnNames = dataTable.getColumnNames();
```

GetColumnIndex

対応する列インデックスを取得します。

構文

```
public int getColumnIndex(String columnName)
```

パラメータ

- 列名

結果

対応する列インデックスを返します。

例

```
int columnIndex = dataTable.getColumnIndex("City");
```

GetColumnCount

DataTable にある列の数を取得します。

構文

```
public int getColumnCount()
```

パラメータ

- なし

結果

列の数を返します。

例

```
int columnCount = dataTable.getColumnCount();
```

Clear

DataTable のデータを消去します。

構文

```
public void clear()
```

パラメータ

- なし

結果

なし

例

```
dataTable.clear();
```

Iterator

DataTable 内のすべての DataRow を含むイテレータです。

構文

```
public Iterator iterator()
```

パラメータ

- なし

結果

DataTable 内のすべての DataRow を含むイテレータを返します。

例

```
Iterator iter = dataTable.iterator();
while (iter.hasNext())
{
 DataRow row = (DataRow)iter.next();
}
```

AddRow

行を DataTable に追加します。

構文

```
public void addRow(DataRow row)
```

パラメータ

- Row - DataTable に追加する DataRow。

結果

なし

例

```
DataTable dataTable = message.getDataTable();

DataRow row = dataTable.newRow();
row.set("AddressLine1","4203 Greenridge");

dataTable.addRow(row);
```

NewRow

新しい DataRow を DataTable に作成します。

構文

```
public DataRow newRow()
```

パラメータ

- なし

結果

新規作成された DataRow を返します。

例

```
DataRow row = dataTable.newRow();
row.set("AddressLine1", "4203 Greenridge");

dataTable.addRow(row);
```

GetRowCount

DataTable にある DataRow の数を取得します。

構文

```
public int getRowCount()
```

パラメータ

- なし

結果

DataTable にある DataRow の数を返します。

例

```
int rowCount = dataTable.getRowCount();
```

Merge

指定された DataTable と現在の DataTable を結合します。

構文

```
public void merge(DataTable other)
```

パラメータ

- 現在の DataTable と結合する他の DataTable。

結果

なし

例

```
DataTable otherDataTable = new DataTable();
dataTable.merge(otherDataTable);
```

DataRow

DataRowには入出力データの個々のレコードが含まれます。このクラスに関連付けられたメソッドを使って、出力用の列名を定義しレコードを DataTable に追加します。

GetColumnNames

すべての列名を取得します。

構文

```
public String[] getColumnNames()
```

パラメータ

- なし

結果

列名の文字列配列を返します。

例

```
String[] columnNames = dataRow.getColumnNames();
```

GetColumnIndex

対応する列インデックスを取得します。

構文

```
public int getColumnIndex(String columnName)
```

パラメータ

- Name - 列名。

結果

対応する列インデックスを返します。

例

```
int columnIndex = dataRow.getColumnIndex("City");
```

Get

この DataRow の列インデックスによってフィールド配列から値を取得します。

構文

```
public String get(int index)
```

パラメータ

- 指定された値を関連付けるインデックス。

結果

この DataRow の列インデックスの値を返します。

例

```
String value = dataRow.get(1);
```

Get

この DataRow の列名によってフィールド配列から値を取得します。

構文

```
public String get(String columnName)
```

パラメータ

- Name — 指定された値を関連付ける名前。

結果

この DataRow の列名の値を返します。列名が存在しない場合は空の文字列を返します。

例

```
String value = dataRow.get("City");
```

Merge

指定された DataRow と現在の DataRow を結合します。

構文

```
public void merge(DataRow other)
```

パラメータ

- 現在の DataRow と結合する他の DataRow。

結果

なし

例

```
DataRow otherDataRow = new DataRow();
dataRow.merge(otherDataRow);
```

Set

DataRow の対応する列の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。

構文

```
public void set(int Index, String value)
```

パラメータ

- 指定された値を関連付ける列インデックス。

- 指定された名前に関連付ける値。

結果

なし

Exceptions

- IndexOutOfBounds — 列インデックスが無効です。

例

```
DataRow row = dataTable.newRow();
row.set(0,"4203 Greenridge");
row.set(1,"Austin");
row.set(2,"Texas")
dataTable.addRow(row);
```

AddChild

新しい DataRow を指定された親子関係に追加します。 指定された関係が存在する場合、与えられた DataRow は既存の DataRow コレクションに追加されます。 存在しない場合、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。

構文

```
public void addChild(String childName, DataRow childDataRow)
```

パラメータ

- Name - 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。
- Value - 関係に追加する DataRow。

結果

なし

例

```
DataRow childDataRow = new DataRow();
childDataRow.set("Address", "100 Congress");
...
DataRow dataRow = new DataRow();
...
dataRow.addChild("child1", childDataRow);
```

GetChildren

指定された関係から子の行を取得します。

構文

```
public List getChildren(String childName)
```

パラメータ

- 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。

結果

指定された関係から子の行を返します。

例

```
List childRows = row.getChildren("child1");
```

ListChildNames

指定された親子関係のすべての名前を取得します。

構文

```
public Set listChildNames()
```

パラメータ

なし

結果

指定された親/子関係の名前セットを返します。

例

```
Set childNames = row.listChildNames();
```

SetChildren

与えられた指定の親子関係の行を設定します。 この名前で行がすでに存在する場合は、呼び出し元に返されます。

構文

```
public List setChildren(String childName, List DataRows)
```

パラメータ

なし

結果

指定された親/子関係の名前セットを返します。

例

```
List rows = dataRow.getChildren("child1");
parentRow.setChildren("child2", rows);
```

Set

DataRow の対応する列の値を設定します。この名前の値が存在する場合は、古い値を置き換えます。

構文

```
public void set(int Index, String value)
```

パラメータ

- 指定された値を関連付ける列インデックス。
- 指定された名前に関連付ける値。

結果

なし

Exceptions

- IndexOutOfBounds — 列インデックスが無効です。

例

```
DataRow row = dataTable.newRow();
row.set(0,"4203 Greenridge");
row.set(1,"Austin");
row.set(2,"Texas")
dataTable.addRow(row);
```

6 - .NET API

このセクションの構成

はじめに	191
Server	195
Service	198
Message	199
EnhancedDataTable	206

はじめに

.NET は、Microsoft® オペレーティング システム プラットフォームであり、アプリケーションと、Web サービスやアプリケーション開発を強化する一連のツールとサービスが組み込まれています。

.NET Framework では、共通言語ランタイム (CLR)、Framework クラス ライブラリ (FCL)、および ASP.NET と呼ばれるコンポーネントが使用されます。 CLR は、実行するコンピュータのネイティブ言語でコードを管理および実行する点で、Java 仮想マシンに相当します。 Framework クラス ライブラリは、再利用可能なオブジェクト タイプを収めた巨大なライブラリであり、多数のプログラム機能を網羅します。 ASP.NET は、従来の ASP ページをはるかにしのぐ速度で Web ページやサービスをロードできるサーバー側テクノロジです。 .NET Framework を構成するこれらの 3 つのコンポーネントによって、アプリケーションや Web の開発作業が容易になり、開発工程が簡素化され、既存の環境への統合が容易になります。 異なるプラットフォーム上で、さまざまなプログラミング言語で書かれたサービスを実行するクライアントとサーバーは、相互に迅速かつ容易に通信できます。

.NET テクノロジの詳細については、**msdn.microsoft.com/netframework** を参照してください。

定数

.NET API では、2 組の定数が使用されます。最初の 1 組は `Server`コンポーネント用です。以下の表に説明します。

表 24 : `Server`コンポーネントの定数

定数名	説明	例
Server.HOST	サーバーのホスト名を表す文字列。デフォルトは "localhost" です。	65.89.200.89
Server.PORT	サーバーのポートを表す文字列。デフォルトは "8080" です。	10119

定数名	説明	例
Server.ACCOUNT_ID	サーバーのアカウント ID を表す文字列。デフォルトは NULL です。	user1
Server.ACCOUNT_PASSWORD	サーバーのアカウント パスワードを表す文字列。デフォルトは NULL です。	user1
Server.CONNECTION_TIMEOUT	サーバーの接続タイムアウトをミリ秒単位で表す文字列。デフォルトは "10000" です。	50000
Server.CONNECTION_TYPE	サーバーの接続タイプを表す文字列。現在は HTTP、HTTPS、または SOCKET のみがサポートされています。デフォルトは "HTTP" です。	HTTP(S)
Server.PROXY_HOST	プロキシ サーバーのホスト名を表す文字列。デフォルトは NULL です。	192.168.1.77
Server.PROXY_PORT	プロキシ サーバーのポートを表す文字列。デフォルトは NULL です。	8080
Server.PROXY_USER	プロキシ サーバーのアカウント ID を表す文字列。デフォルトは NULL です。	user1
Server.PROXY_PASSWORD	プロキシ サーバーのアカウント パスワードを表す文字列。デフォルトは NULL です。	user1

2 組目の定数は `Message`コンポーネント用です。

表 25 : Messageコンポーネントの定数

定数名	説明	例
Message.CONTEXT_ACCOUNT_ID	メッセージ コンテキストのアカウント ID を表す文字列。	user1
Message.CONTEXT_ACCOUNT_PASSWORD	メッセージ コンテキストのアカウント パスワードを表す文字列。	user1
Message.CONTEXT_SERVICE_NAME	メッセージ コンテキストのサービス名を表す文字列。	echoservice

エラー メッセージ

.NET API では、次のエラー メッセージが使用されます。

- 接続エラー メッセージ
  - "Connection type not supported."
  - "Client timeout"
- Message Packaging 例外のエラー メッセージ:
  - "Input Message is null."

もう 1 つのエラー メッセージは、.NET Framework クラス ライブラリが正しく使用されていない場合に表示されます。

サンプル アプリケーション

以下のサンプル コードに、.NET API の使い方を示します。

```
using System;
using System.IO;
using System.Collections;
using System.Text;
using System.Data;
```

```
using g1client;

try
 {
 //Create Server
 Server server = new Server();

 //Set connect property to the server
 server.SetConnectionProperty(Server.HOST, "localhost");
 server.SetConnectionProperty(Server.PORT, "10119");
 server.SetConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
 server.SetConnectionProperty(Server.ACCOUNT_ID, "guest");
 server.SetConnectionProperty(Server.ACCOUNT_PASSWORD, "");

 //Connect to server
 server.Connect();

 //Get Service From Server
 Service service = server.GetService("ValidateAddress");

 //Create Input Message
 Message request = new Message();

 //Fill dataTable in the input message
 //Datatable is the .net Framework class
 DataTable dataTable = request.GetDataTable();

 DataColumn column1 = new DataColumn();
 column1.DataType = System.Type.GetType("System.String");
 column1.ColumnName = "AddressLine1";
 dataTable.Columns.Add(column1);

 DataColumn column2 = new DataColumn();
 column2.DataType = System.Type.GetType("System.String");
 column2.ColumnName = "City";
 dataTable.Columns.Add(column2);

 DataColumn column3 = new DataColumn();
 column3.DataType = System.Type.GetType("System.String");
 column3.ColumnName = "StateProvince";
 dataTable.Columns.Add(column3);

 DataRow newRow = dataTable.NewRow();
 newRow[0]="4200 Parliament Place";
 newRow[1]="Lanham";
 newRow[2]="Maryland";

 dataTable.Rows.Add(newRow);

 //Set "option" Properties to the Input Message
 request.PutOption("OutputCasing", "M");
 request.PutOption("OutputRecordType", "A");
```

```
//Process Input Message, return output Message
Message reply = service.Process(request);

//Disconnect from server
server.Disconnect();

//Get the result from the response message
DataTable returnDataTable = reply.GetDataTable();

foreach(DataColumn dc in returnDataTable.Columns)
{
// more code to be added
string columnName = dc.ColumnName;
}
foreach(DataRow dr in returnDataTable.Rows)
{
for (int col = 0; col < returnDataTable.Columns.Count; col++)
{
// more code to be added
string value = (String)dr[col] ;
Console.WriteLine(value);
}
}
}
catch (Exception e)
{

//Error handling
Console.WriteLine("Error Ocurred, " + e.ToString());
}
```

Server

`Server`クラスは、サーバーへの接続、サーバーからの切断、およびサーバーからのサービスの取得に使用されます。

Connect

プロパティを読み取って、使用するゲートウェイ接続を決定し、サーバーへの接続を確立します。

> **注:** .NET では、HTTP、HTTPS、またはソケット サーバー接続プロトコルを使用します。HTTP と HTTPS は、クライアント接続を論理的に確立するだけで、GetService メソッド

または Process メソッドが呼び出されるまで実際にはサーバーに接続しません。ソケットプロトコルは、Connectが呼び出された時点でサーバーへの接続を確立します。

構文

```
public void Connect()
```

パラメータ

なし

結果

なし

Exceptions

- "接続タイプがサポートされていません。"

例

```
Server server = new Server();

// set connect property to the server
 server.SetConnectionProperty(Server.HOST, "localhost");
server.SetConnectionProperty(Server.PORT, "8080");
// more connection properties to be set
// Connect to server
server.Connect();
```

Disconnect

サーバーから切断します。

構文

```
public void Disconnect()
```

パラメータ

なし

結果

クライアントがサーバーから切断されます。

例

```
//Disconnect from server
server.Disconnect();
```

SetConnectionProperty

ホスト名、タイアウト時間など、サーバー接続設定プロパティを設定します。

構文

```
public void SetConnectionProperty(String name, String value)
```

パラメータ

- Name — 接続プロパティの名前。HOST など。
- Value — 接続プロパティの値。"www.myhost.com" など。

結果

なし

例

```
Server server = new Server();

server.SetConnectionProperty(Server.HOST, "localhost");
server.SetConnectionProperty(Server.PORT, "8080");

//Connect to server
server.Connect();
```

GetService

サーバーからサービスを取得します。

> **注：** 使用可能なサービスのリストについては、このガイドの「コンポーネント リファレンス」を参照してください。

構文

```
public Service getService(String serviceName)
```

パラメータ

- 名前 — サービスの名前

結果

特定のサービスを返します。

例

```
Service service = server.GetService("ValidateAddress");
```

Service

`Service` クラスは、メッセージを処理するために使用されます (より具体的に言えば、メッセージをサーバーに送信し、サーバーから応答を受信するために使用されます)。

Process

入力メッセージを処理し、応答メッセージを返します。

構文

```
public Message Process(Message, message)
```

パラメータ

- 入力メッセージ

結果

応答メッセージを返します。

Exceptions

MessageProcessingException:

例

```
//Process Input Message, return output Message
Message reply = service.Process(request);
```

Message

Message クラスは、入力データを送信し、サービスから出力データを受け取ります。Message のプロパティには、コンテキスト プロパティ (アカウント ID、アカウント パスワード、サービス名、サービス メソッド)、オプション プロパティ (サービス固有の実行時プロパティ) などがあります。

GetContext

コンテキスト プロパティの名前で値を取得します。コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

```
public String GetContext(String name)
```

パラメータ

なし

結果

"コンテキスト" プロパティの名前に関連付けられた値を返します。名前が存在しない場合、NULL を返します。

例

```
String value = message.GetContext(Message.CONTEXT_ACCOUNT_ID);
```

GetContext

すべてのコンテキスト エントリが含まれるハッシュテーブルを取得します。ハッシュテーブルは .NET Framework クラスです。

構文

```
public Hashtable GetContext()
```

パラメータ

- なし

結果

すべてのコンテキスト エントリが含まれるハッシュテーブルを返します。

例

```
Hashtable context = message.GetContext();
```

PutContext

指定された名前に基づいてコンテキスト プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。コンテキスト プロパティには、アカウント ID、アカウント パスワード、サービス名、サービス キー、要求 ID などがあります。

構文

```
public void PutContext(String name, String value)
```

パラメータ

- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

例

```
message.PutContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

PutContext

新しいコンテキスト プロパティを現在のコンテキスト プロパティに追加します。

構文

```
public void PutContext(Hashtable context)
```

パラメータ

- 現在のコンテキスト ハッシュテーブルに追加する新しいコンテキスト ハッシュテーブル。

結果

なし

例

```
//Hashtable is the .NET Framework class
Hashtable context = new Hashtable();
//more code
message.PutContext(context);
```

SetContexts

新しいコンテキスト プロパティで現在のコンテキスト プロパティを上書きします。

構文

```
public void SetContexts(Hashtable context)
```

パラメータ

- Context - 現在のコンテキスト ハッシュテーブルを置き換える新しいコンテキスト ハッシュテーブル。

結果

なし

例

```
//Hashtable is the .NET Framework class
Hashtable context = new Hashtable();
//more code
message.SetContexts(context);
```

GetOption

オプション プロパティの名前で値を取得します。 オプション プロパティはサービス固有の実行時オプションです。

構文

```
public String GetOption(String name)
```

パラメータ

- Name - 関連付けられた値を返す名前。

結果

メッセージの "オプション" プロパティ内の名前の値を返します。または、その名前が存在しない場合は NULL を返します。

例

```
String value = message.GetOption("OutputCasing");
```

GetOptions

すべてのオプション エントリが含まれるハッシュテーブルを取得します。 ハッシュテーブルは .NET Framework クラスです。

構文

```
public Hashtable GetOptions();
```

パラメータ

- なし

結果

すべてのオプション エントリが含まれるハッシュテーブルを返します。

例

```
Hashtable options = message.GetOptions();
```

PutOption

指定された名前に基づいてオプション プロパティに値を設定します。指定された名前のエンティティに既存の値がある場合、その値は上書きされます。オプション プロパティはサービス固有の実行時オプションです。

構文

```
public void PutOption(String name, String value)
```

パラメータ

- Name — 指定された値を関連付ける名前。
- Value — 指定された名前に関連付ける値。

例

```
message.PutOption("OutputCasing", "M");
```

PutOptions

新しいオプション プロパティを現在のオプション プロパティに追加します。

構文

```
public void PutOptions(Hashtable options)
```

パラメータ

- Option - 現在のオプション ハッシュテーブルに追加する新たなオプション ハッシュテーブル。

結果

なし

例

```
//Hastable is the .NET Framework class
Hashtable options = new Hashtable();
// more code
message.PutOptions(options);
```

SetOptions

新しいオプション プロパティで現在のオプション プロパティを上書きします。

構文

```
public void SetOptions(Hashtable options)
```

パラメータ

- Options - 新たなオプション ハッシュテーブルを、現在のオプション ハッシュテーブルで置き換えます。

結果

なし

例

```
//Hashtable is the .NET Framework class
Hashtable options = new Hashtable();
//more code
message.SetOptions(options);
```

GetError

メッセージからエラー メッセージを取得します。

構文

```
public String GetError()
```

パラメータ

• なし

結果

メッセージからエラー メッセージを取得して返します。

例

```
String error = message.GetError();
```

GetDataTable

DataTable をメッセージから取得します。 DataTable は .NET Framework クラスです。

構文

```
public DataTable GetDataTable()
```

パラメータ

なし

結果

なし

例

```
//DataTable is the .net Framework class
DataTable dataTable = message.GetDataTable();

DataColumn column1 = new DataColumn();
column1.DataType = System.Type.GetType("System.String");
column1.ColumnName = "AddressLine1";
dataTable.Columns.Add(column1);

DataColumn column2 = new DataColumn();
column2.DataType = System.Type.GetType("System.String");
column2.ColumnName = "City";
dataTable.Columns.Add(column2);

DataRow newRow = dataTable.NewRow();
newRow[0]="4203 Greenridge";
newRow[1]="Austin";
```

```
dataTable.Rows.Add(newRow);
```

EnhancedDataTable

`EnhancedDataTable` は .NET クラスの DataTable を拡張するクラスです。

AddChild

新しい DataRow を指定された親子関係に追加します。 指定された関係が存在する場合、与えられた DataRow は既存の DataRow コレクションに追加されます。 存在しない場合、与えられた DataRow を唯一の要素として新しいコレクションが作成されます。

構文

```
public void AddChild(DataRow parentRow, string name, DataRow newChild)
```

パラメータ

- Name - 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。
- DataRow - 関係に追加する DataRow。

結果

なし

例

```
EnhancedDataTable dataTable = new EnhancedDataTable();

dataTable.Columns.Add(new DataColumn("AddressLine1",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("City",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("StateProvince",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("PostalCode",
System.Type.GetType("System.String")));
```

```
DataRow row = dataTable.NewRow();

row[0] = "510 S Coit St";
row[1] = "Florence";
row[2] = "SC";
row[3] = "29501-5221";

EnhancedDataTable childDataTable = new EnhancedDataTable();

childDataTable.Columns.Add(new DataColumn("AddressLine2",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("City",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("StateProvince",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("PostalCode",
System.Type.GetType("System.String")));


DataRow childRow = childDataTable.NewRow();

childRow[0] = "241 Ne C St";
childRow[1] = "Willamina";
childRow[2] = "OR";
childRow[3] = "97396-2714";

dataTable.AddChild(row, "Child1", childRow);
dataTable.Rows.Add(row);
```

GetChildren

指定された関係から子の行を取得します。

構文

```
public EnhancedDataTable GetChildren(DataRow parentRow, string name)
```

パラメータ

- ParentRow - 親の行。
- Name - 親子関係の名前 ("Flood Plain Data"、"References"、"Used By" など)。

結果

指定された関係から子の行を返します。

例

```
EnhancedDataTable childRows = dataTable.GetChildren(parentRow, "child1");
```

ListChildNames

指定された親子関係のすべての名前を取得します。

構文

```
public string[] ListChildrenNames(DataRow parentRow)
```

パラメータ

なし

結果

指定された親/子関係の名前セットを返します。

例

```
string[] childNames = dataTable.ListChildrenNames( parentRow);
```

SetChildren

与えられた指定の親子関係の行を設定します。 この名前で行がすでに存在する場合は、呼び出し元に返されます。

構文

```
public void SetChildren(DataRow parentRow, string name, EnhancedDataTable
 newTable)
```

結果

指定された親/子関係の名前セットを返します。

例

```
EnhancedDataTable childRows = dataTable1.GetChildren(parentRow,
"child1");
dataTable2.SetChildren(otherParentRow, "child1", childRows);
```

7 - ManagementAPI メソッド (非推奨)

このセクションの構成

はじめに 211
GetLicenseInfo 211
GetVersionInfo 212

はじめに

重要： ManagementAPI Web サービスは非推奨となり、将来のリリースで削除される予定です。システムに関するライセンスとバージョンの情報を取得するには、管理ユーティリティを使用してください。管理ユーティリティの詳細については、『*管理ガイド*』を参照してください。

ManagementAPI Web サービスを介してパブリックに使用できる管理 API メソッドは、`getLicenseInfo`と`getVersionInfo` の 2 つです。ManagementAPI Web サービスの WSDL URL は以下のとおりです。

`http://`*`SpectrumServer`*`:8080/managers/ManagementAPIService?wsdl`

SpectrumServer は、Spectrum™ Technology Platform サーバーのホスト名または IP アドレスを表します。

GetLicenseInfo

重要： ManagementAPI Web サービスは非推奨となり、将来のリリースで削除される予定です。システムに関するライセンスとバージョンの情報を取得するには、管理ユーティリティを使用してください。管理ユーティリティの詳細については、『*管理ガイド*』を参照してください。

GetLicenseInfo メソッドはライセンス オブジェクトを返します。ライセンス オブジェクトには、マシン タイプ、オペレーティング システム タイプ、ホスト名、CPU 制限を表すプロパティがあります。また、feature オブジェクトの配列と restriction オブジェクトの配列も含まれています。これらの配列は、機能と制約に関する情報を確認するために使用できます。feature には、ID、名前、および有効フラグがあります。restriction には、ID、制限値、および開始日があります。

Web サービス

ManagementAPIService

パラメータ

なし

結果

ライセンス オブジェクトを返します。

例

```
License
 string machineType
 string osType
 string hostName
 string CPULimit
 Feature[] features
 Restriction[] restrictions

Feature
 string ID;
 string name;
 Restriction[] restrictions

Restriction
 string ID
 long limit
 datetime startDate
 Feature[] features

ExpirationRestriction extends Restriction

UsageRestriction extends Restriction
 long usages
```

GetVersionInfo

重要： ManagementAPI Web サービスは非推奨となり、将来のリリースで削除される予定です。システムに関するライセンスとバージョンの情報を取得するには、管理ユーティリティを使用してください。管理ユーティリティの詳細については、『*管理ガイド*』を参照してください。

GetVersionInfo メソッドは、VersionInfo オブジェクトの配列を返します。VersionInfo オブジェクトには、名前、バージョン番号、および VersionAttribute オブジェクトのリストがあります。VersionAttribute オブジェクトは、ラベルと値を保持する単純なクラスです。GetVersionInfo 属性は製品固有の属性であり、製品自体によって収集された情報がこの属性に返されます。また、この情報は、Management Console の [バージョン情報] ノードにも表示されます。

> **注：** GetVersionInfo を一度実行し、返される値を確認してからその情報をパースして特定のデータ要素を取得する必要があります。

Web サービス

ManagementAPIService

パラメータ

なし

結果

VersionInfo オブジェクトを返します。

例

```
VersionInfo
 string name
 string version
 VersionAttribute[] attributes

VersionAttribute
 string label
 string value
```

8 - モジュール サービス

このセクションの構成

Address Now モジュール 215
Enterprise Geocoding モジュール 270
GeoConfidence モジュール 358
Universal Addressing モジュール 361
Universal Name モジュール 568

Address Now モジュール

Address Now モジュール

Address Now モジュールは、住所の正規化およびバリデーション ツールで、米国とカナダ以外の住所を広範囲にカバーします。Spectrum™ Technology Platform では、住所の正規化と検証 (バリデーション) を行う 2 つのモジュールを使用でき、Address Now はその 1 つです。もう 1 つは Universal Addressing モジュールです。Address Now モジュールは、米国とカナダ以外の住所について、Universal Addressing モジュールよりも次の点で優れています。

- **データの品質が高い** — Address Now モジュールで使用するデータベースは、Universal Addressing モジュールで使用するデータベースよりも、多くの国について、より新しく、より詳細な情報を提供します。なぜこのような違いがあるかと言うと、Universal Addressing モジュールは、国際データについては、万国郵便連合 (UPU) が提供するデータを利用し、そのデータの対象には多数の国が含まれていますが、UPU は、住所情報の更新と住所情報の詳細度を積極的に管理していません。一方、Address Now モジュールは、(ほとんどの国の) 郵便当局と他のサード パーティのデータ プロバイダが提供するデータを直接利用しています。つまり、最新情報が反映された、より詳細なデータを利用しているということです。
- **ドリル ダウン機能** — Address Now モジュールは、国の住所データに対するドリル ダウン機能も備えています。この機能を使用すると、住所情報をすばやく入力できます。構造に注意する必要はなく、データの入力ミスもありません。
- **2 バイトのサポート** — Address Now モジュールは Unicode 対応で、漢字等の 2 バイト文字を認識します。

Address Now コンポーネント

Address Now は、以下のコンポーネントで構成されます。これらのコンポーネントは、米国、カナダ、および国際住所に対して使用できます。

- **BuildGlobalAddresses** — 個々の住所要素を検索することにより、対話的に住所を作成することができます。
- **GetGlobalCandidateAddresses** — 与えられた住所にマッチすると思われる住所のリストを返します。
- **ValidateGlobalAddress** — 国際郵便データを使用して住所を正規化します。ValidateGlobalAddress 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データが大量に存在する場合は、ValidateGlobalAddress の使用を検討してください。

与えられた入力住所に対し、ValidateGlobalAddress が複数の一致住所を返す場合は、GetGlobalCandidateAddresses を使用して、住所スタックを返すことができます。GetGlobalCandidateAddresses は、返された住所のうち、どれが最良のマッチ結果であるかを判断するための、郵便データベースからの追加情報を返します。

Address Now データベース

Address Now データベースには、サポートされるすべての国の郵便データが含まれています。データベース全体、または特定の国のデータのみをインストールできます。データベースは、サーバーにインストールされます。このデータベースは、Pitney Bowes からのサブスクリプションによって提供され、毎月更新されます。

BuildGlobalAddress

BuildGlobalAddress では、単一または数個の住所要素のみから有効な住所を作成することができます。BuildGlobalAddress は、Address Now モジュールに含まれています。

BuildGlobalAddress の使用

住所の作成は対話的なプロセスであり、住所作成プロセスの各ステップで住所要素を選択することが必要になります。つまり、住所を作成するために、BuildGlobalAddress を 1 度ではなく複数回呼び出す必要があります。まず最初に、BuildGlobalAddress に対する初期化呼び出しを実行します。この呼び出しにより、セッション ID が返されます。以降の呼び出しでは、このセッション ID を使用します。各呼び出しにおいて、BuildGlobalAddress は、住所要素に対する選択肢となる値のリストを提示します。値を選択して、次の住所要素への処理を進めます。この処理を住所全体が作成されるまで続けます。一部の例外を除き、各住所要素に対して個別に呼び出しを行う必要があります。

全体的なプロセスは、次のようになります。

- まず、初期化呼び出しによって、セッションを開始し、システムによって割り当てられたセッション ID を取得します。
- 与えられた住所要素に対する、候補となる値を見つけるための検索呼び出しを行います。
- 必要な値を選択したら、与えられた住所要素に対して確定した値を通知するための確定呼び出しを行います。
- すべての住所要素が確定するまで、検索/確定呼び出しを続行します。
- 最後に、セッションを終了するための終了呼び出しを行います。

プロセスの動作方法を理解するには、Management Console の [プレビュー] タブを使用して、以下の処理を順に実行します。

1. Management Console を開きます。
2. [サービス] タブで **[Address Now]** を選択します。
3. ウィンドウ左側のサービス一覧から、**[Build Global Address]** を選択します。
4. **[オプション]** タブで、必要なオプションを指定します。オプションの詳細については、オプション（222ページ） を参照してください。
5. **[プレビュー]** タブをクリックします。
6. **[Action]** フィールドに、`"init"` と入力します。
7. **[Country]** フィールドに、作成する住所の国を入力します。
8. **[プレビューを実行]** をクリックします。
9. [プレビュー出力] で **[SessionId]** フィールドを探し、値を右クリックしてハイライト表示し、ポップアップ メニューから［コピー］を選択します。
10. [プレビュー入力] で **[SessionId]** フィールドを右クリックしてハイライト表示し、［貼り付け］を選択します。
11. 入力フィールドに以下の値を入力します。
    - Action — "search" と入力します。
    - Country — このフィールドはそのままにします。
    - FieldIndex — 検索する最初のフィールドのインデックス値を入力します。例えば、シカゴの住所を検索する場合は "1" と入力します。米国住所では、フィールド インデックス 1 が、都市フィールドに対応するためです。
    - SearchValue — 検索する値を入力します。例えば、シカゴの住所を作成する場合は、"chicago" と入力します。
    - SessionId — 同じ値のままとします。

      注： 他の入力フィールドの値は無視されます。

12. **[プレビューを実行]** を再度クリックします。
13. 検索結果は、**[Alternatives.InContext]** と **[AlternativesOutContext]** の最大 2 つの出力フィールドに表示されます。インコンテキスト結果とアウトオブコンテキスト結果の違いについては、コンテキストとは（229ページ）を参照してください。
14. 必要な値が見つかった場合は、以下の値を入力フィールドに入力します。
    - **Action** — **"commit"** と入力します。
    - **AlternativeIndex** — 選択する選択肢のインデックス番号を入力します。インデックス値の最小値は 1 ではなく、0 です。例えば、シカゴを検索する場合、BuildGlobalAddress によっ

て返される選択肢には、以下のようにインデックスが付与されます。"CHICAGO" という値を確定する場合は、[AlternativeIndex] フィールドに "0" と入力します。

- 0—CHICAGO
- 1—CHICAGO HTS
- 2—CHICAGO PARK
- 3—CHICAGO RIDGE
- 4—EAST CHICAGO
- 5—NORTH CHICAGO
- 6—WEST CHICAGO

- **AlternativeContext** — `"in"` または `"out"` と入力し、**[AlternativeIndex]** で指定したインデックス値が **[Alternatives.InContext]** フィールドと **[Alternatives.OutContext]** フィールドのどちらの選択肢リストのものであるかを示します。
- **SessionId** — この値は同じままにします。

  注： 他の入力フィールドの値は無視されます。

15. **[プレビューを実行]** を再度クリックします。指定した値が、適切な住所要素の [Field.n.Value] フィールドに表示されます。
16. 住所が作成されるまで、検索と確定のステップを必要な時だけ繰り返します。
17. 入力フィールドに以下の値を入力することにより、セッションを終了します。

    - **Action — "close"** と入力します。
    - **SessionId** — この値は同じままにします。

      注： 他の入力フィールドの値は無視されます。

入力

表 26 : BuildGlobalAddress の入力

フィールド名 パラメータ	書式	説明
Action	String	実行するアクションを指定します。次のいずれかです。 **init** 初期化。このアクションは、セッションを開始し、他のすべてのアクションで必要となるセッション ID を返します。init アクションでは、[Country] 入力フィールドが必須です。 **search** 特定の住所要素に対する値を検索し、選択肢となる値のリストを返します。search アクションでは、次の入力フィールドが必須です。 • FieldIndex • SearchValue • SessionId **commit** search アクションで返された値のうちの 1 つをフィールドに代入します。commit アクションでは、次の入力フィールドが必須です。 • AlternativeIndex • AlternativeContext • SessionId **clear** [FieldIndex] フィールドに指定されたフィールドの確定を取り消します。clear アクションでは、次の入力フィールドが必須です。 • FieldIndex • SessionID **close** セッションを終了します。close アクションでは、[SessionId] 入力フィールドが必須です。

フィールド名 パラメータ	書式	説明
AlternativeContext	String	commit アクションにおいて、**[Alternatives.InContext]** フィールドと **[Alternatives.OutContext]** フィールドのどちらからの値を選択したかを表します。このフィールドは、他のアクションでは無視されます。次のいずれかです。 **in** **[Alternatives.InContext]** フィールドからの値を確定しています。つまり、**[AlternativeIndex]** 入力フィールドで指定した値は、**[Alternatives.InContext]** 出力フィールドの値に対応します。 **out** **[Alternatives.OutContext]** フィールドからの値を確定しています。つまり、**[AlternativeIndex]** 入力フィールドで指定した値は、**[Alternatives.OutContext]** 出力フィールドの値に対応します。
AlternativeIndex	文字列	commit アクションにおいて、作成中の住所で使用する値を指定します。例えば、都市を検索し、BuildGlobalAddress が 3 つの都市を返した場合に、インデックス値を指定することによって、どの都市を選択したかを表します。BuildGlobalAddress が提示する選択肢のインデックス値は 0 から開始します。つまり、最初の選択肢のインデックス値は 0、2 つめの選択肢のインデックス値は 1 で、以後同様に続きます。 この入力フィールドは、commit 以外のアクションでは無視されます。
Country	文字列	init アクションにおいて、作成する住所が所在する国を指定します。入力した国フォーマット (英語名、2 文字の ISO 3116-1 Alpha-2 コード、または 3 文字の ISO 3116-1 Alpha-3 コード) を使用して、国を指定します。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。 この入力フィールドは、init 以外のアクションでは無視されます。

フィールド名 パラメータ	書式	説明
FieldIndex	文字列	searchアクションに対しては、検索する住所要素を指定します。clearアクションに対しては、確定を取り消す住所要素を指定します。次のいずれかです。 **all** すべての住所要素に対して、"clear" アクションを実行します。このオプションは、"clear" アクションのみに適用されます。 **<インデックス番号>** 特定の住所要素に対して、アクションを実行します。住所要素のインデックスを調べるには、[Field.n.Name] フィールドを参照して、必要なフィールドを探します。値nは、フィールドのインデックスを表します。例えば、米国住所のZIP Codeのインデックスを調べる場合を考えます。init 呼び出しの後、[Field.0.Name] が "Zip" となるため、ZIP Code のフィールド インデックスが "0" であることがわかります。 この入力フィールドは、search と clear 以外のアクションでは無視されます。
SearchValue	文字列	search アクションにおいて、検索する値を指定します。この値は、[FieldIndex] で指定したフィールドに対して適切な値である必要があります。例えば、[FieldIndex] で ZIP Code フィールドを指定した場合は、このフィールドに ZIP Code または ZIP Code の一部を入力します。同様に、[FieldIndex] で City フィールドを選択した場合は、このフィールドに都市名または都市名の一部を指定します。このフィールドを空白のままにすると、searchはインコンテキストのすべての値を返します。インコンテキストおよびアウトオブコンテキストの値については、コンテキストとは（229ページ）を参照してください。 この入力フィールドは、search以外のアクションでは無視されます。
SessionId	文字列	この呼び出しで使用するセッション ID を指定します。セッション ID を取得するには、init アクションを使用します。動作のない状態が 5 分間続くと、セッションの有効期限が切れ、新しい init 呼び出しを実行して、新しいセッションを開始する必要が生じます。 このフィールドは、init 以外のすべてのアクションで必須です。

オプション

表 27 : BuildGlobalAddress のオプション

オプション名	説明
HomeCountry	デフォルト国を指定します。データ内の住所の多くが所在する国を指定する必要があります。例えば、住所の多くがカナダに所在する場合は、カナダを指定します。BuildGlobalAddress は、[StateProvince]、[PostalCode]、および [Country] の各住所フィールドから国を特定できなかった場合、指定された国を使用して、住所の検証を試みます。
OutputCountryFormat	出力で国の名前として使うフォーマットを指定します。次のいずれかです。 **E** 出力の国名には英語表記を使います (デフォルト)。 **I** 国を 2 文字の ISO コードで出力します。 **U** 国を 3 文字の UPU コードで出力します。
ShowExtraAddressLine	都市、州/省、および郵便番号を [AddressLine] 出力フィールドのいずれかに格納するかどうかを指定します。このオプションの設定とは関係なく、出力フィールド [都市]、[州/省]、および [郵便番号] には常に都市、州/省、および郵便番号が格納されます。 **Y** Y — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納します (デフォルト)。 **N** N — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納しません。
OutputPostalCodeSeparator	ZIP Code またはカナダの郵便番号において、区切り文字 (スペースまたはハイフン) を使用するかどうかを指定します。 例えば、区切り文字ありの ZIP + 4® Code は 20706-1844、区切り文字なしは 207061844 になります。区切り文字ありのカナダの郵便番号は P5E"1S7、区切り文字なしは P5E1S7 になります。 **Y** 区切り文字を使用します (デフォルト)。 **N** 区切り文字を使用しません。 **注 :** カナダの郵便番号ではスペースが、米国のZIP + 4® コードではハイフンが使用されます。

オプション名	説明
MaximumResults	このオプションのデフォルト値として、1 ～ 10000 の間の任意の値が設定できます。デフォルト値は、50 レコードです。Enterprise Designer で設定された値が、Management Console における設定値よりも優先されることに注意してください。

出力

BuildGlobalAddress は、各入力住所の住所データとリターン コードを返します。

住所データ

表 28 : BuildGlobalAddress の出力

フィールド名	書式	説明
Action	文字列	この呼び出しで Action 入力フィールドに指定された値を表示します。この入力フィールドの詳細については、入力（219ページ）を参照してください。
AddressLine1	文字列	フォーマット済みの最初の住所行。
AddressLine2	文字列	フォーマット済みの 2 行目の住所行。
AddressLine3	文字列	フォーマット済みの 3 行目の住所行。
AddressLine4	文字列	フォーマット済みの 4 行目の住所行。
AddressLine5	文字列	フォーマット済みの 5 行目の住所行。
AddressLine6	文字列	フォーマット済みの 6 行目の住所行。

フィールド名	書式	説明
AddressLine7	文字列	フォーマット済みの 7 行目の住所行。
AddressLine8	文字列	フォーマット済みの 8 行目の住所行。
AlternativeContext	文字列	この呼び出しで AlternativeContext 入力フィールドに指定された値を表示します。詳細については、入力（219ページ）を参照してください。
AlternativeIndex	文字列	この呼び出しで AlternativeIndex 入力フィールドに指定された値を表示します。この入力フィールドの詳細については、入力（219ページ）を参照してください。
Alternatives.InContext	文字列	検索したフィールドの候補値のうち、確定済みのフィールドのコンテキストに合致するものが、カンマで区切られたリスト。コンテキストの詳細については、コンテキストとは（229ページ）を参照してください。
Alternatives.InContext.Count	文字列	検索によって返された「インコンテキスト」結果の数。コンテキストの詳細については、コンテキストとは（229ページ）を参照してください。
Alternatives.OutContext	文字列	検索したフィールドの候補値のうち、確定済みのフィールドのコンテキストに合致しないものが、カンマで区切られたリスト。コンテキストの詳細については、コンテキストとは（229ページ）を参照してください。
Alternatives.OutContext.Count	文字列	検索によって返された「アウトオブコンテキスト」結果の数。コンテキストの詳細については、コンテキストとは（229ページ）を参照してください。
ApartmentLabel	文字列	アパート指定子 (STE や APT など)。例: 123 E Main St.APT 3

フィールド名	書式	説明
ApartmentNumber	文字列	アパート番号。例: 123 E Main St.APT **3**
Building	文字列	建物の名前。
City	文字列	都市名。
Country	文字列	この呼び出しの、[Country] 入力フィールドで指定された値。この入力フィールドの詳細については、入力（219ページ）を参照してください。
Country	文字列	2 文字または 3 文字の ISO コード、または、国の英語名。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
Department	文字列	複数の部門に整理された任意のものの個々の部分の名前。例えば、企業の中の財務部門など。
Field.n.CommitFlag	文字列	フィールド n の値を選択済みかどうか (つまり、値を "確定" したかどうか) を表します。次のいずれかです。 **Y** このフィールドの値は確定済みです。 **N** このフィールドの値は確定していません。
Field.n.Index	文字列	フィールド n (n は 0 ～ 10) を指すために使用するインデックス値。例えば、米国住所の場合、ZIP フィールドのインデックス値は "0" です。
Field.n.Name	文字列	フィールド n (n は 0 ～ 10) に含まれる住所要素の名前。例えば、米国住所の場合、Field.0.Name は ZIP です。

フィールド名	書式	説明
Field.n.Value	文字列	フィールド n (n は 0 ～ 10) に対して確定済みの値。 init 呼び出しでは、このフィールドは空白です。
FieldIndex	文字列	この呼び出しの、[FieldIndex] 入力フィールドで指定された値。この入力フィールドの詳細については、入力（219ページ）を参照してください。
FirmName	文字列	会社名。例: **Pitney Bowes** 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	文字列	家番号。例: 123 E Main **St**.Apt 3
POBox	文字列	郵便局の私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode	文字列	郵便番号。米国では、ZIP Code™になります。
PostalCode.AddOn	文字列	ZIP + 4® コードの 4 桁アドオン部分。例えば、60655-1844 という ZIP Code™ において、4 桁のアドオン部分は 1844 になります(米国住所のみ)。
PostalCode.Base	文字列	5 桁の ZIP Code™。例えば、20706 (米国住所のみ)。
Principality	文字列	国内の地域。例えば、イングランド、スコットランド、ウェールズは公国です。このフィールドは、通常は空白です。

フィールド名	書式	説明
SearchFieldIndex	文字列	前回の検索操作で検索されたフィールドのインデックス値。
SearchValue	文字列	この呼び出しで SearchValue 入力フィールドに指定された値を表示します。この入力フィールドの詳細については、入力（219ページ）を参照してください。
SessionId	文字列	この呼び出しで SessionId 入力フィールドに指定された値を表示します。この入力フィールドの詳細については、入力（219ページ）を参照してください。
StateProvince	文字列	州または省の省略形。
StreetName	文字列	ストリート名。例: 123 E Main **St**.Apt 3
StreetSuffix	文字列	ストリート接尾語。例: 123 E Main **St**.Apt 3
SubCity	文字列	地区または郊外。地区または郊外を住所に含めるのが一般的な国で使用します。例を次に示します。 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA この住所では "Bradley Stoke" が該当します。

フィールド名	書式	説明
SubStreet	文字列	住所の識別に使われる 2 番目のストリート名。2 つのストリート名を住所に含めるのが一般的な国で使用します。例を次に示します。 12 The Mews **High Street** この例では、"High Street" が 2 番目のストリート名です。このストリート名は、配達先を正確に特定するために使用できます。前の例の "The Mews" は短いストリートなので、住所を正確に示すために別のストリート名が必要とされることから、"High Street" が追記されています。このような場合、"High Street" がメインまたは既知のストリート名です。
USCountyName	文字列	米国住所に対しては、住所がある郡の名前です。

リターン コード

表 29 : BuildGlobalAddress のリターン コード

フィールド名	書式	説明
Status	文字列	マッチの成功または失敗。 **null** 成功 **F** 失敗
Status.Code	文字列	失敗の原因 (ある場合)。 • SessionError • SeverError • CountryNotFound

フィールド名	書式	説明
Status.Description	文字列	問題の説明 (ある場合)。 **Please initialize new session** Status.Code=SessionError の場合にこの値が表示されます。 **Null or empty action** Status.Code=SessionError の場合にこの値が表示されます。 **Unknown action** Status.Code=SessionError の場合にこの値が表示されます。 **Invalid session** Status.Code=SessionError の場合にこの値が表示されます。 **Invalid value for** Status.Code=SessionError の場合にこの値が表示されます。 **Cannot Search Committed Field** Status.Code=SessionError の場合にこの値が表示されます。 **Module not licensed** Status.Code=ServerError の場合にこの値が表示されます。 **Could Not Identify Country** Status.Code=CountryNotFound の場合にこの値が表示されます。

コンテキストとは

住所要素の検索を実行する際、BuildGlobalAddress は、確定済みの住所要素を参照し、返す値を、確定済みの住所要素のコンテキスト内にあるかどうかによって分割します。例えば、米国の場合、米国には次の都市が存在します。

イリノイ州の都市

- CHICAGO
- CHICAGO HTS
- CHICAGO RIDGE
- NORTH CHICAGO
- WEST CHICAGO

インディアナ州の都市

- EAST CHICAGO

ネバダ州の都市

- CHICAGO PARK

州として "IN" (インディアナ州) の値が確定済みの状態で、都市 "chicago" を検索した場合、BuildGlobalAddress は「インコンテキスト」結果として EAST CHICAGO を返します。それがインディアナ州に存在するためです。"chicago" に対するその他のマッチ結果はすべて、アウトオブコンテキスト結果として返されます。同様に、州として "IL" (イリノイ州) の値が確定済みである場合は、BuildGlobalAddress は、アウトオブコンテキスト結果として EAST CHICAGO と CHICAGO PARK を返し、「インコンテキスト」結果として CHICAGO、CHICAGO HTS、CHICAGO RIDGE、NORTH CHICAGO、および WEST CHICAGO を返します。

GetGlobalCandidateAddresses

GetGlobalCandidateAddresses は、与えられた入力住所にマッチすると思われる住所のリストを返します。入力住所が、Address Now データベースの複数の住所にマッチする場合は、可能性のある複数のマッチ結果が返されます。入力住所が、Address Now データベースの 1 つの住所のみにマッチする場合は、住所データは返されません。

GetGlobalCandidateAddresses は、Address Now モジュールに含まれています。

入力

GetGlobalCandidateAddresses は正規化済み住所を受け取ります。どの国の住所であるかにかかわらず、すべての住所がこのフォーマットを使用します。[AddressLine1] および [Country] は、必須の入力フィールドです。他のフィールドはすべてオプションです。

表 30 : GetGlobalCandidateAddresses の入力

columnName	書式	説明
AddressLine1	文字列	最初の住所行。これは必須のフィールドです。
AddressLine2	文字列	2 行目の住所行。
AddressLine3	文字列	3 行目の住所行。
AddressLine4	文字列	4 行目の住所行。

columnName	書式	説明
AddressLine5	文字列	5 行目の住所行。
AddressLine6	文字列	6 行目の住所行。
AddressLine7	文字列	7 行目の住所行。
AddressLine8	文字列	8 行目の住所行。
City	文字列	都市名
StateProvince	文字列	州または省。
PostalCode	文字列 [10]	住所の郵便番号は、次のフォーマットのいずれかで表されます。 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
Country	String	国。入力の国フォーマットとして選択したフォーマット (英語名または ISO コード) を使って国を指定します。ISO コードの一覧は、**ISO 国コードとモジュールサポート**（598ページ）を参照してください。
FirmName	文字列	会社名または企業名。

オプション

表 31 : GetGlobalCandidateAddresses のオプション

オプション名	説明/有効値
HomeCountry	デフォルト国を指定します。ほとんどの郵送物の宛先となる国を指定します。例えば、ほとんどの郵送先がカナダであれば、カナダを指定します。GetGlobalCandidateAddresses は、[StateProvince]、[PostalCode]、および [Country] の各住所フィールドから国を特定できなかった場合、指定された国を使用して、住所の検証を試みます。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力には、大文字と小文字が混在させます(デフォルト)。次に例を示します。123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。次に例を示します。123 MAIN ST MYTOWN FL 12345
OutputCountryFormat	出力で国の名前として使うフォーマットを指定します。次のいずれかです。 **E** 出力の国名には英語表記を使います (デフォルト)。 **I** 出力の国名には 2 文字の ISO コードを使います。 **U** 出力の国名には 2 文字の UPU コードを使います。
ShowExtraAddressLine	都市、州/省、および郵便番号を [AddressLine] 出力フィールドのいずれかに格納するかどうかを指定します。このオプションの設定とは関係なく、出力フィールド [都市]、[州/省]、および [郵便番号] には常に都市、州/省、および郵便番号が格納されます。 **Y** Y — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納します (デフォルト)。 **N** N — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納しません。

オプション名	説明/有効値
OutputPostalCodeSeparator	ZIP™ Code またはカナダの郵便番号において、区切り文字 (スペースまたはハイフン) を使用するかどうかを指定します。 例えば、区切り文字ありの ZIP + 4® Code は 20706-1844、区切り文字なしは 207061844 になります。区切り文字ありのカナダの郵便番号は P5E"1S7、区切り文字なしは P5E1S7 になります。 **Y** 区切り文字を使います (デフォルト)。 **N** 区切り文字を使いません。 **注 :** カナダの郵便番号ではスペースが、米国のZIP + 4® コードではハイフンが使用されます。
MaximumResults	出力する候補住所の最大数。デフォルトは 50 です。最大値は 100 です。
ReturnUserData	妥当性を確認できなかった入力住所を出力データに含めるかどうかを指定します。 **Y** 妥当性を確認できなかった入力データを含めます。 **N** 妥当性を確認できなかった入力データを含めません (デフォルト)。

出力

GetGlobalCandidateAddresses は、各住所の住所データとリターン コードを返します。

住所データ

表 32 : GetGlobalCandidateAddresses の住所データ出力

フィールド名	書式	説明
AddressLine1	文字列	フォーマット済みの最初の住所行。
AddressLine2	文字列	フォーマット済みの 2 行目の住所行。
AddressLine3	文字列	フォーマット済みの 3 行目の住所行。

フィールド名	書式	説明
AddressLine4	文字列	フォーマット済みの 4 行目の住所行。
AddressLine5	文字列	フォーマット済みの 5 行目の住所行。
AddressLine6	文字列	フォーマット済みの 6 行目の住所行。
AddressLine7	文字列	フォーマット済みの 7 行目の住所行。
AddressLine8	文字列	フォーマット済みの 8 行目の住所行。
ApartmentLabel	文字列	アパート指定子 (STE や APT など)。例: 123 E Main St.APT **3**
ApartmentNumber	文字列	アパート番号。例: 123 E Main St.APT **3**
Building	文字列	建物の名前。
City	文字列	都市名。
Country	文字列	国の ISO コードまたは英語名。ISO コードの一覧は、**ISO 国コードとモジュールサポート**（598ページ）を参照してください。
Department	文字列	複数の部門に整理された任意のものの個々の部分の名前。例えば、企業の中の財務部門など。

フィールド名	書式	説明
FirmName	文字列	会社名。例: **Pitney Bowes** 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	文字列	家番号。例: 123 E Main **St**.Apt 3
POBox	文字列	私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode	文字列	現地の郵便当局が記入を必須とする郵便番号。例えば、米国の場合、米国の郵便番号は ZIP Code です。
PostalCode.AddOn	文字列	米国住所に対しては、ZIP + 4® Code の末尾 4 桁。
PostalCode.Base	文字列	米国住所に対しては、5 桁の ZIP Code がこれに該当します。
Principality	文字列	国内の地域。例えば、イングランド、スコットランド、ウェールズは公国です。このフィールドは、通常は空白です。
StateProvince	文字列	州または省の省略形。
StreetName	文字列	ストリート名。例: 123 E Main **St**.Apt 3
StreetSuffix	文字列	ストリート接尾語。例: 123 E Main **St**.Apt 3

フィールド名	書式	説明
SubCity	文字列	地区または郊外。地区または郊外を住所に含めるのが一般的な国で使用します。例を次に示します。 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA この住所では "Bradley Stoke" が該当します。
SubStreet	文字列	住所の識別に使われる 2 番目のストリート名。2 つのストリート名を住所に含めるのが一般的な国で使用します。例を次に示します。 12 The Mews **High Street** この例では、"High Street" が 2 番目のストリート名です。このストリート名は、配達先を正確に特定するために使用できます。前の例の "The Mews" は短いストリートなので、住所を正確に示すために別のストリート名が必要とされることから、"High Street" が追記されています。このような場合、"High Street" がメインまたは既知のストリート名です。
USCountyName	文字列	米国住所に対しては、住所がある郡の名前です。

リターン コード

表 33 : GetGlobalCandidateAddresses のリターン コード

フィールド名	書式	説明
ACRCode	文字列	ACR (住所訂正結果) コードは、各レコードでどのデータが変更されたかを示します。このコードの意味については、**ACR コード**（267ページ）を参照してください。
Confidence	文字列	返された住所に割り当てられた確信レベル。範囲は 0 ～ 100 です。0 は失敗を表し、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表します。

フィールド名	書式	説明
Status	文字列	マッチの成功または失敗。 **Null** 成功 **F** 失敗
Status.Code	文字列	失敗の原因 (ある場合)。 • RequestFailed • ServerError • CountryNotFound
Status.Description	文字列	問題の説明 (ある場合)。 **Maximum records cannot be set to 0. Minimum value should be 1** Status.Code=RequestFailed の場合にこの値が表示されます。 **Address Not Found** Status.Code=RequestFailed の場合にこの値が表示されます。 **Module not licensed** Status.Code=ServerError の場合にこの値が表示されます。 **Could Not Identify Country** Status.Code=CountryNotFound の場合にこの値が表示されます。

ValidateGlobalAddress

ValidateGlobalAddress は、米国およびカナダ以外のアドレスのアドレス標準化と検証機能が強化されています。ValidateGlobalAddress 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の妥当性を確認する必要がある場合は、ValidateGlobalAddress の使用を検討してください。

ValidateGlobalAddress は、Address Now モジュールに含まれています。

入力

ValidateGlobalAddress は、正規化済みの住所を入力として受け取ります。どの国の住所であるかにかかわらず、すべての住所がこのフォーマットを使用します。

表 34 : ValidateGlobalAddress の入力

columnName	書式	説明
AddressLine1	文字列	最初の住所行。
AddressLine2	文字列	2 行目の住所行。
AddressLine3	文字列	3 行目の住所行。
AddressLine4	文字列	4 行目の住所行。
AddressLine5	文字列	5 行目の住所行。
AddressLine6	文字列	6 行目の住所行。
AddressLine7	文字列	7 行目の住所行。
AddressLine8	文字列	8 行目の住所行。
City	文字列	都市名
StateProvince	文字列	州または省。
PostalCode	文字列 99999 ~~99999-9999~~ A9A9A9 A9A 9A9 9999 999	住所の郵便番号。米国では、ZIP Code™になります。

columnName	書式	説明
Country	文字列	入力の国フォーマットとして選択したフォーマット (英語名または ISO コード) を使って国を指定します。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	文字列	会社名または企業名。

オプション

入力データ オプション

表 35 : ValidateGlobalAddress の入力データ オプション

オプション名	説明
HomeCountry	デフォルト国を指定します。住所の大半がある国を指定してください。例えば、処理する住所の大部分がカナダにある場合は、カナダを指定します。ValidateGlobalAddress では、[StateProvince]、[PostalCode]、または [Country] 住所フィールドで国を特定できない場合に自国を使って住所を確認しようとします。有効な値の一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。

出力データ オプション

表 36 : ValidateGlobalAddress の出力データ オプション

optionName	説明
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力に大文字と小文字を混在させます (デフォルト)。次に例を示します。123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。次に例を示します。123 MAIN ST MYTOWN FL 12345

optionName	説明
OutputCountryFormat	出力で国の名前として使うフォーマットを指定します。次のいずれかです。 **E** 出力の国名には英語表記を使います (デフォルト)。 **I** 出力の国名には 2 文字の ISO コードを使います。 **U** 出力の国名には 2 文字の UPU コードを使います。
StandardizeAddressOnFail	住所を検証できない場合に正規化された住所を返すかどうかを指定します。住所には、その国の標準住所書式が設定されます。このオプションを選択しない場合、住所のに失敗すると出力住所コンポーネント フィールド ([StreetName]、[HouseNumber] など) は空白になります。 **N** 失敗した住所の書式を整えません (デフォルト)。 **Y** 検証に失敗した住所を正規化します。
ShowExtraAddressLine	都市、州/省、および郵便番号を [AddressLine] 出力フィールドのいずれかに含めるかどうかを指定します。このオプションの設定とは関係なく、出力フィールド [都市]、[州/省]、および [郵便番号] には常に都市、州/省、および郵便番号が格納されます。 **Y** Y — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納します (デフォルト)。 **N** N — 都市、州/省、および郵便番号を [AddressLine] 出力フィールドに格納しません。
OutputPostalCodeSeparator	ZIP™ Code またはカナダ郵便番号で区切り文字 (スペースまたはハイフン) を使うかどうかを指定します。 例えば、区切り文字ありの ZIP + 4® Code は 20706-1844、区切り文字なしは 207061844 になります。区切り文字ありのカナダの郵便番号は P5E"1S7、区切り文字なしは P5E1S7 になります。 **Y** 区切り文字を使います (デフォルト)。 **N** 区切り文字を使いません。 **注：** カナダの郵便番号ではスペースが、米国のZIP + 4® コードではハイフンが使用されます。

optionName	説明
FormatOnFail	住所の妥当性を確認できない場合に書式を整えた住所を返すかどうかを指定します。住所には、その国の標準住所書式が設定されます。 **Y** 住所を検証できない場合に書式設定された住所を返します。 **N** 住所を検証できない場合に書式設定された住所を返しません (デフォルト)。
ValidateAddress	住所検証を有効にします。住所検証は以下の手順で進められます。 • 各コンポーネントを当該国の参照データと照合します。 • スペルの誤りを訂正します。 • 欠落しているコンポーネントを補います。 • 郵便番号を訂正するか補います。 **Y** 住所を検証します (デフォルト)。 **N** 住所を検証しません。
FormatAddress	住所のコンポーネントを法律で定められたフォーマットまたはカスタム フォーマットに書式設定します。 **Y** 住所に書式を設定します (デフォルト)。 **N** 住所に書式を設定しません。

正規化オプション

表 37 : ValidateGlobalAddress の正規化オプション

オプション名	説明
FlagVulgarWords	">VulgarWord<" 形式を使って、不適切な語を出力中にマークするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
DebugOutput	このオプションは、トラブルシューティング用の情報を出力フィールド [Email1]、[Email2]、[URL1]、および [URL2] に設定するかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReportVulgarWords	不適切な語を検出するかどうかを指定します。このオプションを有効にすると、ValidateGlobalAddress は、結果を示す値を [WCRCode] 出力フィールドに返します。 **Y** はい **N** いいえ (デフォルト)
StandardizeComponent.Department	住所を正規化するときに [Department] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.FirmName	住所を正規化するときに [FirmName] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.Building	住所を正規化するときに [Building] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.SubBuilding	住所を正規化するときに [SubBuilding] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
StandardizeComponent.HouseNumber	住所を正規化するときに [HouseNumber] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.SubStreet	住所を正規化するときに [SubStreet] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.StreetName	住所を正規化するときに [StreetName] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.POBox	住所を正規化するときに [POBox] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.SubCity	住所を正規化するときに [SubCity] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.City	住所を正規化するときに [City] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
StandardizeComponent.USCountyName	住所を正規化するときに [USCountyName] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.StateProvince	住所を正規化するときに [StateProvince] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.Principality	住所を正規化するときに [Principality] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.PostalCode	住所を正規化するときに [PostalCode] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.Plus4	住所を正規化するときに [+4] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
StandardizeComponent.Country	住所を正規化するときに [Country] フィールドに値を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

検証オプション

表 38 : ValidateGlobalAddress 検証オプション

オプション名	説明
ValidateComponent.Department	住所を検証するときに [Department] フィールドを対象とするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ValidateComponent.FirmName	住所を検証するときに [FirmName] フィールドを対象とするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ValidateComponent.Building Option.ValidateComponent.Building	住所を検証するときに [Building] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.SubBuilding	住所を検証するときに [SubBuilding] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.HouseNumber	住所を検証するときに [HouseNumber] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
ValidateComponent.SubStreet	住所を検証するときに [SubStreet] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.StreetName	住所を検証するときに [StreetName] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.POBox	住所を検証するときに [POBox] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.SubCity	住所を検証するときに [SubCity] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.City	住所を検証するときに [City] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.USCountyName	住所を検証するときに [USCountyName] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
ValidateComponent.StateProvince	住所を検証するときに [StateProvince] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.Principality	住所を検証するときに [Principality] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.PostalCode	住所を検証するときに [PostalCode] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.Plus4	住所を検証するときに [Plus4] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ValidateComponent.Country	住所を検証するときに [Country] フィールドを対象とするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.Department	住所を検証するときに [Country] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
ForceUpdate.FirmName	住所を検証するときに [FirmName] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.Building	住所を検証するときに [Building] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.SubBuilding	住所を検証するときに [SubBuilding] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.HouseNumber	住所を検証するときに [HouseNumber] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.SubStreet	住所を検証するときに [SubStreet] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.StreetName	住所を検証するときに [StreetName] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
ForceUpdate.POBox	住所を検証するときに [POBox] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.SubCity	住所を検証するときに [SubCity] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.City	住所を検証するときに [City] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.USCountyName	住所を検証するときに [USCountyName] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.StateProvince	住所を検証するときに [StateProvince] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.Principality	住所を検証するときに [Principality] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
ForceUpdate.PostalCode	住所を検証するときに [PostalCode] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.Plus4	住所を検証するときに [Plus4] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ForceUpdate.Country	住所を検証するときに [Country] フィールドを訂正するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ReplaceAlias.Department	Address Now データベースにエイリアスが見つかった場合に、[Department] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.FirmName	Address Now データベースにエイリアスが見つかった場合に、[FirmName] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.Building	Address Now データベースにエイリアスが見つかった場合に、[Building] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
ReplaceAlias.SubBuilding	Address Now データベースにエイリアスが見つかった場合に、[SubBuilding] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.HouseNumber	Address Now データベースにエイリアスが見つかった場合に、[HouseNumber] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.SubStreet	Address Now データベースにエイリアスが見つかった場合に、[SubStreet] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.StreetName	Address Now データベースにエイリアスが見つかった場合に、[StreetName] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.POBox	Address Now データベースにエイリアスが見つかった場合に、[POBox] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.SubCity	Address Now データベースにエイリアスが見つかった場合に、[Subcity] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
ReplaceAlias.City	Address Now データベースにエイリアスが見つかった場合に、[City] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.USCountyName	Address Now データベースにエイリアスが見つかった場合に、[USCountyName] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.StateProvince	Address Now データベースにエイリアスが見つかった場合に、[StateProvince] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.Principality	Address Now データベースにエイリアスが見つかった場合に、[Principality] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
ReplaceAlias.PostalCode	Address Now データベースにエイリアスが見つかった場合に、[PostalCode] フィールドを上書きするかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
ReplaceAlias.Plus4	Address Now データベースにエイリアスが見つかった場合に、[+4] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
ReplaceAlias.Country	Address Now データベースにエイリアスが見つかった場合に、[Country] フィールドを上書きするかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)
CautiousUpdate	このオプションは、[強制的に更新] オプションと併用することで、処理中にデータが大きく変更されないようにすることができます。 **Y** はい **N** いいえ (デフォルト)
CrossComponentMatch	住所正規化と検証の一般的なエラーを訂正するためにクロス コンポーネント マッチを実行するかどうかを指定します。クロス コンポーネント マッチを実行すると、入力データのフィールドにあるデータと Address Now データベースの別のフィールドにあるデータがマッチするかどうかがチェックされます。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
UseReferenceDiacritics	住所の違いが付加記号(アクセント記号、ウムラウト記号など)のみの場合にValidate Global Addressが住所を変更して郵便データベース内の住所の付加記号に一致させるかどうかを指定します。次のいずれかです。 **Y** はい (デフォルト) **N** いいえ 例えば、**[付加記号の参照を使用]** を有効にすると、データは以下のように処理されます。 入力される都市名: Chalon-Sur-Saône 郵便データベース内の都市名: CHALON SUR SAONE 出力される都市名: CHALON SUR SAONE 入力される都市名: ARTEMIVS'K 郵便データベース内の都市名: ARTEMIVSK 出力される都市名: ARTEMIVSK 一方、**[付加記号の参照を使用]**を有効にしない場合は、データは以下のように処理されます。 入力される都市名: Chalon-Sur-Saône 参照される都市名: CHALON SUR SAONE 出力される都市名: Chalon-Sur-Saône 入力される都市名: ARTEMIVS'K 参照される都市名: ARTEMIVSK 出力される都市名: ARTEMIVS'K このオプションの設定は、**[書き直し方法]** オプションに影響しないことに注意してください。
KeepStandardizationChanges	正規化による変更 ("ROAD" を "RD" に変更するなど) を ACR コードとして報告するかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

オプション名	説明
AcceptanceLevel	[許容レベル] 設定は、住所全体を検証されたと見なすために検証しなければならない住所コンポーネントの最小限の数を指定します。[許容レベル] に指定された値は、ACR コードの 2 番目の文字に対応します。詳細については、**ACR コード**（267ページ）を参照してください。 許容レベルは [内部マッチ スコア] オプションとは異なります。許容レベルは、Validate Global Address が検証するコンポーネントの数を指定するオプションであり、検証対象のコンポーネントが郵便データベースの住所コンポーネントにどの程度マッチするかは問いません。一方、[内部マッチ スコア] は、出力住所が検証済みの正しいバージョンの入力住所である可能性を示します。 次のいずれかです。 **-1** 許容レベルは、住所の国に基づいて適切なレベルに自動的に設定されます。例えば、米国の住所では、米国の住所は許容レベル 4 で処理されます。 **0** コンポーネントを 1 つも検証しません (デフォルト) **1** 国名のみを検証します **2** 都市名および国名を検証します **3** 都市名、郵便コード、および国名を検証します **4** ストリート名、都市名、郵便コード、および国名を検証します **5** 構内番号、建物名、従属する建物名、私書箱、企業名、ストリート名、都市名、郵便番号、および国名を検証します
InnerMatchScore	住所検証の最小確信レベルを指定します。[Confidence] 出力フィールドの値がこのレベル値と同じかそれを超える住所が検証の対象となります。値がレベル値よりも小さい住所は検証されません (出力フィールド [Status] に "F" が設定されます)。 0 ～ 100 の範囲の値を指定できます。値が大きいほど、住所検証の実行に必要な確信レベルが高くなります。デフォルト値は 60 です。
CompanyWeight	Address Now データベース内のデータと比較される [FirmName] フィールドの相対的な重要性を示す 0 ～ 10 の範囲の整数。このオプションは、確信値に影響するので、正しい更新と正しくない更新を区別するために確信の度合いを調節する目的に使えます。詳細については、**ACR コード**（267ページ）を参照してください。 デフォルト値は 1 です。

オプション名	説明
StreetWeight	Address Now データベース内のデータと比較される [StreetName] フィールドの相対的な重要性を示す 0 ～ 10 の範囲の整数。このフィールドが他のフィールドに対して持つ相対的な重要性を表す 0 ～ 10 の範囲の整数です。詳細については、**ACR コード**（267ページ）を参照してください。 デフォルト値は 10 です。
CityWeight	Address Now データベース内のデータと比較される [City] フィールドの相対的な重要性を示す 0 ～ 10 の範囲の整数。このフィールドが他のフィールドに対して持つ相対的な重要性を表す 0 ～ 10 の範囲の整数です。詳細については、**ACR コード**（267ページ）を参照してください。 デフォルト値は 8 です。
PostcodeWeight	Address Now データベース内のデータと比較される [PostalCode] フィールドの相対的な重要性を示す 0 ～ 10 の範囲の整数。このフィールドが他のフィールドに対して持つ相対的な重要性を表す 0 ～ 10 の範囲の整数です。詳細については、**ACR コード**（267ページ）を参照してください。 デフォルト値は 8 です。
OuterMatchScoreLines	外部マッチ スコア行を計算するときに使う住所の行数を示す 0 ～ 8 の範囲の値。デフォルト値は 8 です。外部マッチ スコア行の詳細については、**外部マッチ スコア**（266ページ）を参照してください。

出力フォーマット オプション

表 39 : ValidateGlobalAddress の出力フォーマット オプション

オプション名	説明
Transliteration	出力住所の付加記号をフォーマットする方法を指定します。次のいずれかです。 **0** 書き直しを実行しません。付加記号は、入力または郵便データベースで提供されたままで残されます。こちらがデフォルトです。 **1** 付加記号を除去し、それに相当する非装飾文字で置き換えます。 **2** 言語固有の書き直しルールに従って、付加記号をそれに相当する非装飾文字または文字シーケンスに書き直します。 例えば、スウェーデンの住所に 3 つの書き直しオプションを適用した場合のそれぞれの結果を以下に示します。"Västra Frölunda" の違いに注目してください。 **0** Gustaf Wernersgata 12 S-42132 Västra Frölunda **1** Gustaf Wernersgata 12 S-42132 Vastra Frolunda **2** Gustaf Wernersgata 12 S-42132 Vaestra Froelunda
FormatComponent.Department	[Department] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.FirmName	[FirmName] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
FormatComponent.Building	[Building] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.SubBuilding	[SubBuilding] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.HouseNumber	[HouseNumber] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.SubStreet	[SubStreet] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.StreetName	[StreetName] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.POBox	[POBox] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
FormatComponent.SubCity	[SubCity] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.City	[City] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.USCountyName	[USCountyName] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.StateProvince	[StateProvince] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.Principality	[Principality] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.PostalCode	[PostalCode] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ

オプション名	説明
FormatComponent.Plus4	[+4] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい (デフォルト) **N** いいえ
FormatComponent.Country	[Country] フィールドにフォーマット済み住所の出力を設定するかどうかを指定します。 **Y** はい **N** いいえ (デフォルト)

出力

住所データ出力

表 40 : ValidateGlobalAddress の住所データ出力

フィールド名	書式	説明
AddressLine1	文字列	フォーマット済みの最初の住所行。
AddressLine2	文字列	フォーマット済みの 2 行目の住所行。
AddressLine3	文字列	フォーマット済みの 3 行目の住所行。
AddressLine4	文字列	フォーマット済みの 4 行目の住所行。
AddressLine5	文字列	フォーマット済みの 5 行目の住所行。
AddressLine6	文字列	フォーマット済みの 6 行目の住所行。

フィールド名	書式	説明
AddressLine7	文字列	フォーマット済みの 7 行目の住所行。
AddressLine8	文字列	フォーマット済みの 8 行目の住所行。
ApartmentLabel	文字列	アパート指定子 (STE や APT など)。例: 123 E Main St.APT 3
ApartmentNumber	文字列	アパート番号。例: 123 E Main St.APT **3**
Building	文字列	建物の名前。
City	文字列	都市名。
Country	文字列	国の ISO コードまたは英語名。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
Department	文字列	フランス語圏やスペイン語圏で使われる国内の従属する地区。たとえば、フランスは 100 の department (県) に分割されます。
FirmName	文字列	会社名。例: **Pitney Bowes** 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	文字列	家番号。例: 123 E Main **St**.Apt 3

フィールド名	書式	説明
Latitude	文字列	住所から確認できる最も精度の高い緯度。ポイント レベルの場所またはセントロイドである場合があります。精度レベルは、[ECRCode] 出力フィールドをチェックすると確認できます。詳細については、**ECR コード**（265ページ）を参照してください。
Longitude	文字列	住所から確認できる最も精度の高い経度。ポイント レベルの場所またはセントロイドである場合があります。精度レベルは、[ECRCode] 出力フィールドをチェックすると確認できます。詳細については、**ECR コード**（265ページ）を参照してください。
POBox	文字列	郵便局の私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode	文字列	郵便番号。米国では、ZIP Code™になります。
PostalCode.AddOn	文字列	ZIP + 4® コードの 4 桁アドオン部分。例えば、60655-1844 という ZIP Code™ において、4 桁のアドオン部分は 1844 になります(米国住所のみ)。
PostalCode.Base	文字列	5 桁の ZIP Code™。例えば、20706 (米国住所のみ)。
Principality	文字列	国内の地域。例えば、イングランド、スコットランド、ウェールズは公国です。このフィールドは、通常は空白です。
StateProvince	文字列	州または省の省略形。
StreetName	文字列	ストリート名。例: 123 E Main **St**.Apt 3
StreetSuffix	文字列	ストリート接尾語。例: 123 E Main **St**.Apt 3

フィールド名	書式	説明
SubCity	文字列	地区または郊外。地区または郊外を住所に含めるのが一般的な国で使用します。例を次に示します。 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA この住所では "Bradley Stoke" が該当します。
SubStreet	文字列	住所の識別に使われる 2 番目のストリート名。2 つのストリート名を住所に含めるのが一般的な国で使用します。例を次に示します。 12 The Mews **High Street** この例では、"High Street" が 2 番目のストリート名です。このストリート名は、配達先を正確に特定するために使用できます。前の例の "The Mews" は短いストリートなので、住所を正確に示すために別のストリート名が必要とされることから、"High Street" が追記されています。このような場合、"High Street" がメインまたは既知のストリート名です。
USCountyName	文字列	米国住所に対しては、住所がある郡の名前です。

リターン コード

表 41 : ValidateGlobalAddress のリターン コード

columnName	書式	説明
ACRCode	文字列	ACR (住所訂正結果) コードは、各レコードでどのデータが変更されたかを示します。このコードの意味については、**ACR コード**（267ページ）を参照してください。

columnName	書式	説明
Confidence	文字列	返された住所に割り当てられた確信レベル。範囲は 0 ～ 100 です。0 は失敗を表し、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表します。この値は、ACR コードの最後の 3 文字と同じであり、検証マッチ スコアと呼ばれます。詳細については、**ACR コード**（267ページ）を参照してください。
ECRCode	文字列	住所訂正結果 (ECR) コードは、住所に対して返される緯度/経度の精度レベルを表します。詳細については、**ECR コード**（265ページ）を参照してください。
Email1	文字列	追加の正規化情報。
Email2	文字列	追加の正規化情報。
OuterMatchScore	文字列	各住所行への変更を測るスコア。詳細については、**外部マッチ スコア**（266ページ）を参照してください。
Status	文字列	マッチの成功または失敗。 • **null**—Success • **F**—Failure
Status.Code	文字列	失敗の原因 (ある場合)。 • UnableToValidate • ServerError • CountryNotFound
Status.Description	文字列	問題の説明 (ある場合)。 • **Address Not Found**— Status.Code=UnableToValidate の場合にこの値が表示されます。 • **Module not licensed**— Status.Code=ServerError の場合にこの値が表示されます。 • **Could Not Identify Country**— Status.Code=CountryNotFound の場合にこの値が表示されます。

columnName	書式	説明
URL1	文字列	追加の正規化情報。
URL2	文字列	追加の正規化情報。
WCRCode	文字列	語訂正結果 (WCR) コードは、入力住所に不適切な語が見つかったことを示します。コードは、2 つの要素に分かれています。 • **ロケーション コード** — 次のいずれかの値です。 • **AB** — 不適切な語が住所に見つかったことを示します。 • **NB** — 不適切な語が名前に見つかったことを示します。 • **個数** — ロケーション コードが示す場所に見つかった不適切な語の数。 例えば、"AB2" は 2 つの不適切な語が入力住所に見つかったことを意味します。

ECR コード

住所訂正結果 (ECR) コードは、住所に対して返される緯度/経度の精度レベルを表します。コードは、接頭文字とコード本体がダッシュで区切られるフォーマットで構成されます。

接頭文字は常に "EL" で始まり、これに全体の精度レベルを示す 1 ～ 5 の番号が続きます。

- 5 — ポイント ジオコード
- 4 — ストリートセントロイド
- 3 — 郵便番号セントロイド
- 2 — 都市セントロイド
- 1 — 地域セントロイド

コードの本体は、住所とジオコードのマッチングに使われたコンポーネントを示します。本体が英数字で構成されることに注意してください。コード本体の意味は次のとおりです。

- P — 敷地/家番号 (私書箱番号を含む)
- S — ストリート
- T — 都市
- R — 地域/州
- Z — 郵便番号
- C — 国

コードの本体で使用できる数字オプションは、4 または 0 です。

- 4 — コンポーネント データが、ジオコードと住所のマッチング検出に使用できました。
- 0 — コンポーネント データは使用できませんでした。

例: EL4-P0S4T4R4Z4C4

この例では、P の次にある 0 は、この住所マッチング検出に敷地/家番号を使用できなかったものの、それを除く、ストリート名から国名までのコンポーネントは使用できたことを意味します。

外部マッチ スコア

外部マッチ スコアは、住所を検証するために ValidateGlobalAddress が各住所行をどの程度変更したかを示します。このスコアは、正規化前の住所行を検証/フォーマット後の住所と比較して決定されます。`OuterMatchScoreLines` オプションを 0 より大きな値に設定した場合に、このスコアが生成されます。

外部マッチ スコアは、検証マッチ スコアと似ています。後者は ACR コードの一部です (**ACR コード**（267ページ）を参照)。両者の違いは、外部マッチ スコアが住所行への変更 (フォーマットなど) を数値化するのに対し、検証マッチ スコアはデータを検証できたかどうかのみを数値化することです。

例えば、次のような入力住所行を処理前に受け取ったとします。

住所行 1: 5 camden cres
住所行 2: bath
住所行 3: uk

この住所行は、処理後に以下のようになります。

住所行 1: 5 Camden Crescent
住所行 2: Bath
住所行 3: BA1 5HY
住所行 4: United Kingdom

このデータの検証マッチ スコアは 84% であり、外部マッチ スコアは 23% です。

検証マッチ スコアの値が高いのは、住所コンポーネントが検証前でもかなり正確だったからです。ストリート名は、大文字と小文字の違いと省略形が使われていたことを除き、実在の名称でした。都市と国の名前はどちらも有効でした。唯一正しくなかったのが郵便番号です (元の住所には欠落していました)。結果として、84% という比較的高い検証マッチ スコアが得られました。

外部マッチ スコアが低いのは、フォーマット後の住所行が入力住所とかなり異なるからです。前の例では、入力住所行 3 は "uk" でしたが、出力では "BA1 5HY" となっています。住所行 4 は入力では空でしたが、出力では値が設定されました。住所行 1 も変更されました。よって、外部スコアはかなり低い数値になります。

ACR コード

ACR (住所訂正結果) コードは、各レコードでどのデータが変更されたかを示します。ACR は、以下のような形式です。

L5-P0S0A5T1R0Z0C4-098

ACR コードは、以下の 3 つの部分で構成されています。

- バリデーション レベル
- コンポーネント ステータス
- 検証マッチ スコア

バリデーション レベル

住所訂正結果の最初の 2 文字は、バリデーションのタイプとレベルを表します。

最初の文字 (常に英字) はバリデーションのタイプを示します。

- **U** — 住所の正規化ができません。
- **C** — 住所はコンポーネント形式です。
- **L** — 住所は書式が設定され、住所行に変換されました。
- **R** — 住所は元に戻されました。許容レベルに達していません。

2 番目の文字 (常に数字) はバリデーションのレベルを示します。数字が大きいほど、バリデーションのレベルが高くなります。到達できるレベルは以下のとおりです。

- **0** — コンポーネントはいずれも検証されませんでした。
- **1** — 国名のみが検証されました。
- **2** — 都市名と国名が検証されました。
- **3** — 都市名、郵便番号、および国名が検証されました。
- **4** — ストリート名、都市名、郵便番号、および国名が検証されました。
- **5**—敷地番号、建物名、従属する建物名、私書箱、企業名、ストリート名、都市名、郵便番号、および国名が検証されました。

コンポーネント ステータス

ACR コードの 2 番目の部分は、住所の主要コンポーネントのステータスを表します。住所のコンポーネントは、以下のように識別されます。

- 3、4 文字目: P — 敷地/家番号
- 5、6 文字目: S — ストリート
- 7、8 文字目: A — 従属する都市 (都市エリア)

- 9、10 文字目: T — 都市
- 11、12 文字目: R — 地域/州
- 13、14 文字目: Z — 郵便番号/ZIP Code®
- 15、16 文字目: C — 国

コンポーネントの直後には数字が 1 文字配置され、以下のいずれかの値を示します。

- **0** — 見つかりません/空。
- **1** — 入力データ内での位置に基づいて推測しました。
- **2** — Address Now モジュール データベースに基づいて認識されました。
- **3** — Address Now モジュール データベースに基づいて認識され、正規の形式に変換されました。
- **4** — Address Now モジュール データベースを使って検証されました。
- **5** — Address Now モジュール データベースを使って更新/訂正されました。
- **6** — Address Now モジュール データベースを使って追加されました。
- **7** — 正常な空。
- **8** — Address Now モジュール データベースを使って部分的に認識されました。
- **9** — 訂正して Address Now モジュール データベースに一致させる必要があります。

検証マッチ スコア

検証マッチ スコアは、ACR コードの最後の 3 文字 (17 ～ 19 文字目) に設定されます。これは、正規化データ (コンポーネント形式のデータ) を、Address Now モジュール データベースから返されたマッチ候補と比較した結果です。

このスコアを計算するために、Address Now モジュール データベースから返されたすべてのフィールドが確認され、それらが個別に既存のコンポーネント データと比較されます。その後、全体のマッチ スコアを計算するために、これらの個別の値から平均スコアが求められます。この計算には、住所バリデーションのオプションを設定するダイアログ ボックスで指定できるマッチ スコア重み付けが加味されます。例を次に示します。

入力データ:

住所行 1: 11 High Street
都市: Anytown
国: UK

正規化データ:

敷地: 11
ストリート: High Street
都市: Anytown

このレコードを検証すると、Address Now モジュール データベースからは以下のデータが返されます。

敷地: 11
ストリート: High Street
都市: Anytown
郵便番号: ZZ9 9ZZ

Address Now モジュール データベースを正規化データと比較すると、以下の結果が得られます。

- 敷地番号: 100% マッチ
- ストリート: 100% match
- 都市: 100% match
- 郵便番号: 使用なし (入力時に空白)

これらの割合を結合すると、マッチ スコアは 100% となります。

別の例を以下に示します。

入力データ:

住所行 1: bergerstrasse 12
住所行 2: munich
住所行 3: 80124
国: Germany

正規化データ:

敷地: 12
ストリート: Bergerstr.
都市: München
郵便番号: 80124

Address Now モジュール データベースからの出力:

敷地: 12
ストリート: Burgerstr.
都市: München
郵便番号: 80142

Address Now モジュール データベースの出力を正規化データと比較すると、以下の結果が得られます。

- 敷地番号: 100% マッチ
- ストリート: 90% マッチ (実際の数値は、2 つの値をテキストとして照合して決定されます)
- 都市: 100% match
- 郵便番号: 80% マッチ (番号の位置が入れ替わっているため)

マッチ スコアの重み付けをすべて 1 に設定した場合、全体のマッチ スコアは 92% になります。郵便コードのマッチ スコア重み付けを増やすと、全体のマッチ スコアは減ります。これは、郵便番号コンポーネントのスコア (80%) が計算の際に重視されるからです。都市名のマッチ スコア重

み付けを増やすと、全体のマッチ スコアは増えます。これは、都市名コンポーネントのスコア (100%) がより重視されるからです。

例:

L5-P4S4A5T5R4Z4C4-098

- L は、住所行を作成するために書式が設定されたことを意味します。
- バリデーション レベルは 5 です。Address Now モジュール データベースとのマッチングが最高レベルで実行されたことを意味します。
- 従属する都市 (A) と都市 (T) を除き、すべてのコンポーネントが 4 に設定されています。これは、Address Now モジュール データベースを使って検証されたことを意味します。
- 従属する都市と都市のコードはどちらも 5 に設定されています。これらのコンポーネントが Address Now モジュール データベースに従って訂正されたことを意味します。

住所全体の Address Now モジュール データベースに対するマッチングは 98% です。

> **注:** また、検証マッチ スコアの代わりに "SDS" という値が返される場合があります。SDS が返されるのは住所が正規化されなかったことを意味し、その場合、住所が元に戻された可能性があります。

Enterprise Geocoding モジュール

Enterprise Geocoding モジュール

Enterprise Geocoding モジュールは、住所の正規化、住所ジオコーディング、および郵便番号セントロイド ジオコーディングを実行します。住所を入力して出力を取得できます。例えば、出力として取得する地図上のポイント（緯度と経度）は、詳細な空間分析やデモグラフィックスによる分類に使用できます。また、ジオコード (緯度と経度で表現される地図上のポイント) を入力し、そのジオコードに関する住所情報を取得することもできます。

コンポーネント

Enterprise Geocoding モジュールは、次のステージで構成されます。取得したライセンスによっては一部のステージが含まれないことがあります。

- **GeocodeAddressAUS —** オーストラリアの住所を受け取り、それに対応する緯度/経度座標などの情報を返します。

> **注：** Geocode Address AUS は非推奨になりました。Geocode Address AUS から使用されるステージは、GNAF PID Location Search のみです。その他のすべてのオーストラリアのジオコーディング機能には、Geocode Address Global コンポーネントを使用してください。

- **GeocodeAddressGBR** — 英国の住所を受け取り、それに対応する緯度/経度座標などの情報を返します。

  > **注：** Geocode Address GBR は、GBR AddressBase Plus データ ソースをサポートします。GBR Streets (TomTom) データ ソースには、Geocode Address Global を使用してください。

- **GeocodeAddressGlobal** — サポートされている任意の国の住所を受け取り、それに対応する緯度/経度座標などの情報を返します。Geocode Address Global は、ライセンスを取得した国の住所のみをジオコーディングします。オーストラリアと英国はサポート対象外です。
- **Geocode Address** — サポートされている国のいずれかに位置する住所を受け取り、都市センロイド (一部の国では郵便番号センロイド) を返します。Geocode Address World は、通り住所レベルでのジオコーディングを行うことができません。
- **Geocode Africa** — アフリカの多くの国々に対して通りレベルのジオコーディングを提供します。また、一部の国については、都市または地方のセントロイドや郵便番号セントロイドも決定できます。
- **Geocode Middle East** — 中東の多くの国々に対して通りレベルのジオコーディングを提供します。都市または地方のセントロイドも決定できます。Middle East は英語とアラビア語の両方の文字セットをサポートしています。
- **Geocode Latin America** — ラテン アメリカの多くの国々に対して通りレベルのジオコーディングを提供します。都市または地方のセントロイドも決定できます。一部の国には郵便番号の対象範囲があります。
- **GeocodeUSAddress** — 入力住所を受け取り、それに対応する緯度/経度座標などの情報を返します。
- **GNAFPIDLocationSearch** — Geocoded National Address File Persistent Identifier (G-NAF PID) の住所および緯度/経度座標を特定します。
- **ReverseAPNLookup** — Assessor's Parcel Number (APN)、FIPS (連邦情報処理標準) 郡コード、FIPS 州コードを受け取り、小区画の住所を返します。
- **ReverseGeocodeUSLocation** — ジオコード (緯度/経度座標) を入力として受け取り、その場所に対応する住所を返します。
- **ReversePBKeyLookup** — pbKey™ unique identifier を入力として受け取り、住所マッチングの一部として提供されるすべての標準の結果を返します。

Enterprise Geocoding データベース

以下の Enterprise Geocoding モジュール データベースが Spectrum™ Technology Platform サーバーにインストールされています。一部のデータベースは、Pitney Bowes が提供するサブスクリプションによって利用可能で、月に 1 回、または年に 4 回更新されます。その他のデータベースは、USPS® がライセンス提供しています。

米国のジオコーティング データベース (米国のみ)

これらのデータベースには、住所の正規化とジオコーディングに必要な空間データが格納されています。米国に対するジオコーディングを実行するには、これらのデータベースの少なくとも 1 つをインストールする必要があります。マッチングに使用するデータベースを処理オプションを使って設定します。Enterprise Geocoding は、指定したデータベースにマッチングを探します。目的のデータベースがマッチングに使われていることを確認するには、[StreetDataType] 出力フィールドに返される値を確認します。

これらのデータベースでは、GSD ファイルと呼ばれる独自形式のファイルが使われます。ZIP Code セントロイド マッチングについては、ファイル us.Z9 (拡張子は通常 z9) にすべての州のセントロイド情報が含まれています。

- **Centrus Enhanced Geocoding** — このデータベースは、米国地質調査所から提供された TIGER データと米国郵政公社から提供された住所データで構成されています。
- **TomTom Geocoding** — このデータベースは、Centrus Enhanced Geocoding データベースに未収録の最新データを格納しています。使用するには、ライセンスが別途必要です。このデータは、サードパーティの空間データ プロバイダである TomTom から提供され、郵便データは米国郵政公社から提供されています。
- **NAVTEQ Geocoding** — このデータベースは、Centrus Enhanced Geocoding データベースに未収録の最新データを格納しています。使用するには、ライセンスが別途必要です。NAVTEQ データは、サードパーティの空間データ プロバイダである NAVTEQ, から提供されます。これらのデータベースの詳細については、営業担当者にお問い合わせください。
- **ZIP + 4 Centroid** — このデータベースは、住所の正規化と ZIP + 4 セントロイド マッチングのみを提供します。通りレベルでのマッチングは提供しません。

各ジオコーディング データベースには、オプションの Statewide Intersections Index があります。Statewide Intersection Index は、州単位で交差点を迅速に識別するために設計されています。例えば、Statewide Intersection Index を使って、"1st and Main St, CO" をデータベース検索すると、ジオコーディング データベース全体から交差点の各インスタンスを検索する場合よりも迅速にコロラド州内の候補リストが返されます。

米国のポイント データベース (米国のみ)

Points データベースには、小区画の中心を特定できるデータが含まれています。これらのデータベースは、インターネットのマップ データ、損害保険、通信、ユーティリティなどの分野で高度なジオコーディング精度を提供します。

これらのデータベースはオプションですが、Reverse Assessor's Parcel Number (APN) Lookup には Centrus Enhanced Points または Centrus Premium Points が必須です。また、これらのデータベースは個別にライセンスされます。

- **Centrus Points** — このデータベースには、小区画または建物の中心を特定するために必要なデータが格納されています。Assessor's Parcel Number (APN) または標高データは含まれません。
- **Centrus Elevation** — このデータベースは、Centrus Points のデータに標高データが追加されたものです。
- **Centrus Enhanced Points** — このデータベースは、Centrus Points のデータに APN データを追加したものです。
- **Centrus Premium Points** — このデータベースは、Centrus Points のデータに APN データと標高データが追加されたものです。
- **Centrus TomTom Points Database** データベース — このデータベース内のデータはサードパーティの空間データ プロバイダである TomTom により提供されます。
- **Master Location Data** — このデータベースは、米国のすべての郵送可能および配達可能な住所について、取得できる最も適切な住所ポイントの場所を提供します。

Reverse Geocoding データベース (米国のみ)

このデータベースには、緯度/経度の場所を住所に変換するために必要なデータが含まれています。

このデータベースはオプションですが、ReverseGeocodeUSには必須です。また、このデータベースは個別にライセンスされます。

補助ファイル (米国のみ)

補助ファイルには、ユーザ定義レコードが含まれます。補助ファイルを使って、住所マッチングとジオコード マッチングに使うカスタム データを提供できます。

DPV® データベース (米国のみ)

Delivery Point Validation データベースは、米国の郵送先住所の妥当性をチェックするために使用できます。DPV データベースは、オプション機能として配布されており、ジオコーディング データベースの郵送先住所検証機能を強化するためにインストールできます。ジオコーディング データベースの新版がリリースされるたびに、それに対応するオプション DPV データベースの新版もリリースされます。DPV データベースの日付がジオコーディング データベースの日付に一致しな

ければ、DPV の処理は機能しません。DPV 検索は、DPV データベースの有効期限を過ぎると実行されなくなります。

このデータベースはオプションですが、CASS™ の処理には必須です。また、DPV データベースは、ZIP + 4 および ZIP + 4 関連出力 (DPBC、USPS レコード タイプなど) を決定する場合も必要です。また、このデータベースは個別にライセンスされます。

注 :

Postal Service ライセンスでは、DPV を住所または住所録の生成に使うことが禁じられています。また、DPV データベースを米国から輸出することも禁止されています。

EWS データベース (米国のみ)

Early Warning System (EWS) データベースには、郵便データが米国郵便データベースの最新版に掲載されていないことが必要です。

USPS® は、EWS ファイルを週に 1 回更新します。DPV または LACS^{Link} データベースとは異なり、EWS データベースはジオコーディング データベースと同じ日付である必要はありません。EWS.zip ファイルは、以下の USPS® RIBBS Web サイトの CASS セクションから無料でダウンロードできます。

https://ribbs.usps.gov//index.cfm?page=doclist

EWS データベースをダウンロードすると、ファイルは "OUT" という名前で保存されます。"OUT" ファイルを "EWS.txt" という名前に変更してから使用してください。

LACS^{Link} データベース (米国のみ)

LACS^{Link} データベースを使って、地方配送路の住所の通り名に沿った住所への変更、PO Box 番号の再割り当て、または通り名に沿った住所の変更に伴って変更された住所を訂正できます。

このデータベースはオプションですが、CASS™ の処理には必須です。ZIP + 4 または ZIP + 4 関連出力 (配達先バーコード、USPS レコード タイプなど) を受け取るために LACS^{Link} データベースを CASS モードで使用する必要もあります。

LACS^{Link} データベースの日付がジオコーディング データベースの日付に一致しなければ、LACS^{Link} の処理は機能しません。

注 :

USPS ライセンスでは、LACS^{Link} を住所または住所録の生成に使うことは禁じられています。また、LACS^{Link} データベースを米国から輸出することも禁止されています。

Enterprise Geocoding データベース

International Geocoding データベースには、米国外の場所について住所の正規化とジオコーディングを実行するのに必要な空間データが格納されています。各国には専用のデータベースがあり、一部の国にはジオコーディングを強化するためのオプションのデータベースが提供されています。

英国の AddressBase Premium データベース

AddressBase Premium は Ordnance Survey®、Royal Mail、および地方当局から提供されるポイント データベースです。

AddressBase Premium データベースは最高レベルの精度を提供しており、結果コード S8 で示されます。このデータベースには、さらに細かく分割されたプロパティ、教会、コミュニティ センターなど、郵便住所を持たないオブジェクトが含まれます。

AddressBase Premium データベースは UPRN (Unique Property Reference Number) を中心に構築されています。UPRN は、不動産の名前、ステータス、従属する地区、利用法 (単独居住から複数居住など) が変更されたり、不動産が解体されたりしても、一意の不動産を永続的に参照する一意の識別子です。歴史的住所、代替住所、暫定住所のすべてが同じ UPRN に対して記録されます。UPRN は、北アイルランドの住所を除く、すべての AddressBase Premium の候補と共に返されます。

Ordnance Survey データ ソースには北アイルランドの住所は含まれないので、AddressBase Premium には Royal Mail® の北アイルランドの郵便番号住所データが補足されています。この北アイルランド データは、郵便番号セントロイド (結果コード S3) の精度のみを備えています。

AddressBase Premium の詳細については、Ordnance Survey の **https://www.ordnancesurvey.co.uk/business-and-government/help-and-support/products/addressbase-premium.html** を参照してください。

United Kingdom CodePoint データベース

CodePoint Postal Address File (PAF) データベースは、郵便番号セントロイド ジオコーディングを提供します。この CodePoint データベースは、住所のマッチング、妥当性の確認など、ほとんどのアプリケーションに適しています。

CodePoint データベースは Royal Mail が提供するもので、英国 (グレート ブリテンおよび北アイルランド) の通り住所を網羅します。この CodePoint データベースのライセンスは、地域単位ではなくデータセット全体を対象とします。CodePoint データベースから提供される郵便番号セントロイド精度は、結果コード S3 として示されます。

Royal Mail データ ソースの詳細については、

http://www.royalmail.com

Australia Geocoded National Address File (G-NAF)

このデータベースを使って、オーストラリアの住所ジオコーディングを改善できます。これは、オーストラリア全土の地方、通り、および番号を表す唯一の公式インデックスであり、検証済みの地理的座標も含まれます。公式に認められた地方/都市部の住所と非公式の住所(エイリアス)が格納されています。郵便住所と私書箱番号は含まれません。ただし、適切な地方住所情報が存在しない地方があるため、G-NAF データセットには郵便箱 (RMB) 番号、ロット番号、ブロック番号、およびセクション番号が含まれます。

このデータベースをインストールすると、次の 2 つのサブフォルダが作成されます。

- **GNAF123** — ポイントレベル辞書が格納されます。これは、最高精度のジオコーディング (信頼レベル 1、2、または 3) です。
- **GNAF456** — 最高精度のジオコーディング条件を満たさない残りの G-NAF 住所 (信頼レベル 4、5、または 6) が格納されます。

これらはデータベース リソースとして Management Console で個別に指定する必要があります。

住所の有無の確認には両方のデータベースを使用し、小区画レベルのジオコーディングには GANF123 のみを使用することを推奨します。小区画レベルのジオコードが不要な場合は、ジオコーディングに GANF456 データベースを使用することができます。

New Zealand Point データベース

New Zealand Point データベース は、通り住所を建物レベルの精度で特定する郵便ポイント データに基づきます。このデータベースから候補として返されるロケーション X および Y は、建物レベルの精度を備えています。

このデータは、政府機関である Land Information New Zealand によって管理されます。データベースの内容は、各地区の下部組織からの情報に基づいて月に 1 回更新されます。

各国のその他のポイント データベース

各国のその他のポイント データベースも利用できます。Enterprise Geocoding モジュール ポイント データベースは、米国、英国、オーストラリア、ニュージーランドに加えて、以下の国に対して使用可能です。

- Andorra
- Austria
- Belgium
- Canada
- Czech Republic
- Denmark
- France
- French Guiana
- Germany

- Gibraltar
- Hong Kong
- India
- Ireland
- Japan
- Luxembourg
- Malaysia
- Martinique
- Mayotte
- Mexico
- Monaco
- Morocco
- Netherlands
- Portugal
- Reunion
- Singapore
- Slovakia
- Spain
- Sweden

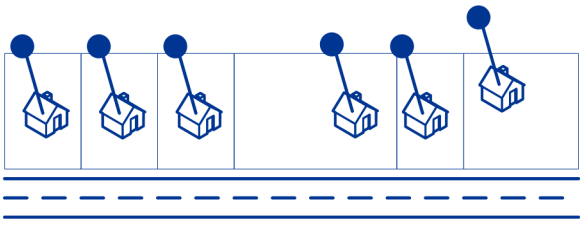
今後のリリースで、これ以外のポイント データベースも利用可能になる可能性があります。Enterprise Geocoding モジュール ポイント データベースのライセンス取得の詳細については、販売担当者にお問い合わせください。

ジオコーディングの概念

ジオコーディングは、住所の緯度/経度座標を決定するプロセスです。住所のジオコーディングを行う方法は複数あります。以下に、精度の高い方法から順に説明します。

ポイント レベル マッチング

ポイントレベル マッチングは、実際の建物の敷地または小区画の中心地点を特定します。最も精度の高いジオコーディング方法であり、インターネットのマップ データ、保険、通信、ユーティリティなどの分野で利用されます。



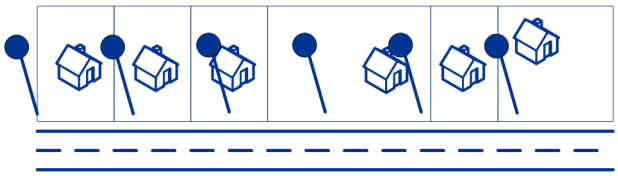
中央線マッチングは、ポイント レベルのジオコードを親のストリートセグメントにリンクするためにポイントレベル マッチングと併用されます。これにより、ポイントレベル マッチング単独で

は取得できない親のストリートセグメントに関する情報が手に入ります。この情報には、ポイント データ ジオコードから中央線マッチングまでの方位も含まれます。

通りマッチング

ストリートマッチングは、ストリートセグメント上の近似の住所位置を特定します。ストリートマッチングでは、住所のストリートにある家番号の範囲に基づいて特定の家番号の近似の位置を計算することで、場所を決定します。例えば、家番号 50 ～ 99 の範囲があるストリートセグメント上に住所がある場合、家番号 75 がこのストリートセグメントの真ん中に位置すると見なします。この方法では、住所がストリートセグメント上に均等な間隔で並んでいることを前提とします。実際には住所がストリートセグメントに均等に配置されていないこともあるので、この方法の精度はポイント マッチングに劣ります。

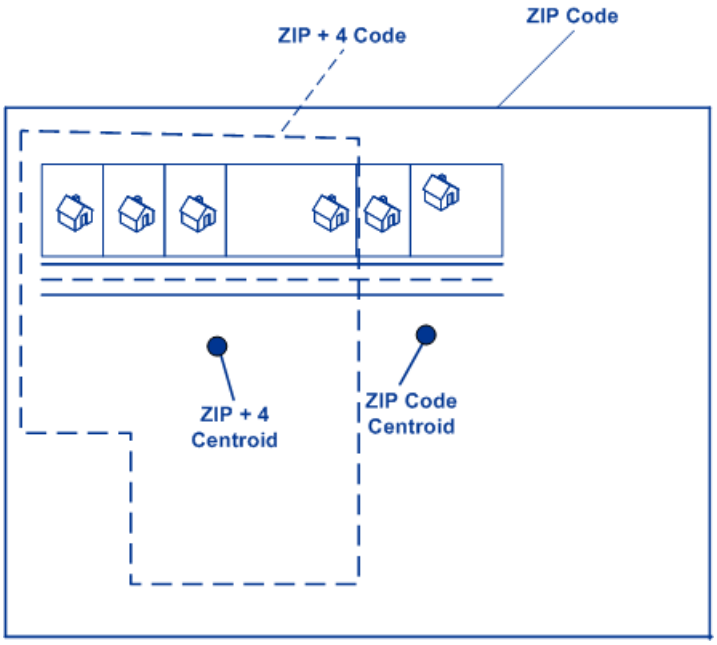
例えば、建物が均等に並んでいないセグメントでのストリートレベル マッチングの結果は、以下の図のようになります。最初の 3 棟の建物は均等な間隔で立地するので、この部分についてはかなり正確なジオコーディングが得られます。しかし、4 番目の建物は、このストリートに面する他の建物と比べて、やや広い敷地に建っています。ストリートレベルのマッチングは、建物が等間隔で並んでいることを前提とするため、4 番目、5 番目、6 番目の住居の精度は、最初の 3 棟の建物よりも劣る結果になります。ポイントレベルのジオコーディングを使っていれば、結果は正確なものとなります。



セントロイドのマッチング

ZIP Code セントロイドのマッチングは、ZIP Code または ZIP + 4 で定義されたエリアの中心点を取得するマッチングであり、最も精度の低いタイプのジオコードです。ZIP セントロイドは ZIP Code の中心点であり、ZIP + 4 セントロイドは ZIP + 4 の中心点です。ZIP + 4 は ZIP Code より小さいエリアを表すため、ZIP + 4 セントロイドは郵便番号セントロイドより正確です。

次の図は、セントロイド マッチングを表したものです。この例では、6 軒の家がすべて同じ ZIP + 4 コード内にあるため、同じジオコードになります。



米国以外のロケーションでのジオコーディング マッチ戦略

Enterprise Geocoding モジュールには、ジオコーディングの精度とマッチ率を制御するための各種オプションがあります。以下の情報は、米国以外のあらゆる国のジオコーダに適用できるさまざまなマッチング手法について説明したものです。異なるオプションを持つ米国のジオコーダ (GeocodeUSAddress) には適用できません。

マッチ率の最大化

マッチ率をできる限り高めるには、ExactMatch オプションを使用して家番号、ストリート、および都市/地方を指定しないようにします。

マッチ率を最大化するもう 1 つの方法は、FallbackToPostal=Y に設定することです。これは、ストリートレベルの近似一致が見つからなかった場合に、ジオコーダが 4 桁の郵便番号セントロイドで代替するということを意味します。このシナリオは誤検出を生むこともありますが、大規模データベースにジオコーディングを行う場合は最適なマッチング ソリューションです。

誤検出の割合が分析に影響を与えるかどうかを評価する必要があります。ヒット率を低下させずに誤検出数を減らすには、ジオコーディング セッション後に結果コードを分析し、それに応じて設定を調整してください。

精度の最大化

高精度でジオコード化された住所が分析に必要な場合は、ジオコーダが高精度のジオコードを返す割合を最大化し、不明確なマッチ (誤検出) を返す数を最小化する戦略を選択します。そのため

には、ExactMatch オプションを使用して、すべての住所要素に対するマッチングで近似一致を必須にします。また、FallbackToPostal=N に設定します。

この手法によって、マッチ率が低下することがありますが、最高の精度を得ることができます。

マッチ率と精度のバランスをとる

マッチ率と地理的精度のバランスをとる戦略を使用したいことがあります。つまり、できる限り多くのレコードに自動的にジオコーディングを行いたいが、同時に、不正確なマッチ (誤検出) の数は最小化したいという場合です。例えば、ジオコーダが次のものを検出した場合、誤検出が発生する可能性があります。

- 入力されたストリートと読みが似ているストリート
- 別の都市にある同じストリート (郵便番号の一致が必須でない場合)
- 異なる家番号のあるストリート (家番号が必須でない場合)

次の設定により、マッチ率と精度との適切なバランスを実現できる場合があります。

- **CloseMatchesOnly** — "Y" を指定します。
- **MustMatchHouseNumber** — "Y" を指定します。
- **MustMatchStreet** — "Y" を指定します。
- **FallbackToPostal** — "N" を指定します。

郵便番号の概念

以下のセクションでは、Enterprise Geocoding モジュールが使用する郵便番号の概念について説明します。

> **注 :** このセクションと、Locatable Address Conversion System、Delivery Point Validation、Early Warning System に関するトピックは、米国ジオコーディングのみに関連します。

二重住所

GeocodeUSAddress は、同一住所行に同一レコードの 2 つの住所を含む入力を処理することができます。例えば、GeocodeUSAddress は次の入力住所を処理できます。

3138 HWY 371
PO BOX 120
PRESCOTT AR 71857

GeocodeUS Address は、2 つの住所がどちらもストリート住所である二重住所を認識しません。例えば、GeocodeUSAddress は、135 Main St 4750 Walnut St Ste 200 を認識しません。GeocodeUSAddress は、2 つの住所が同じ住所タイプであるが、ストリート住所ではない二重住所を認識します。例えば、GeocodeUSAddress は、PO BOX 12 PO BOX 2000 を認識します。

GeocodeUSAddress は、二重住所をパースしたうえで一致するものを検索します。GeocodeUSAddress は、処理モードに基づいて、マッチに対する優先設定を持つ住所を判断します。CASS モードでの GeocodeUSAddress は、[PO Box を優先] オプションと [ストリートの住所を優先] オプションを無視し、PO Box、ストリート、地方配送路、局留め郵便の順序に基づいて一致するものを検索しようとします。緩和モードでは、GeocodeUSAddress は、[住所の優先設定] (AddressPreference) 入力オプションを認識します。

> **注：** GeocodeUSAddress は、完全一致モードと近似モードでは、二重住所処理を実行しません。GeocodeUSAddress は、複数行の住所に対して二重住所処理を実行しません。

Locatable Address Conversion System (LACS)

USPS® Locatable Address Conversion System (LACS) は、地方配送路の住所をストリート名に沿った住所に変換した場合、PO Box 番号の再割り当てがあった場合、またはストリート名に沿った住所が変更した場合に、それに伴って変更した住所を修正します。LACS^{Link} 変換の例を以下に示します。

- 地方配送路の住所のストリート名に沿った住所への変換: 旧住所: RR 3 Box 45 新住所: 1292 North Ridgeland Drive
- ストリート名またはストリート番号の変更: 旧住所: 23 Main Street 新住所: 45 West First Avenue
- PO Box 番号の再割り当て: 旧住所: PO Box 453 新住所: PO Box 10435

LACS^{Link} は、CASS 処理に必須です。

Delivery Point Validation (DPV)

Delivery Point Validation (DPV®) は、住所情報の正確さを個々の郵便住所まで照合する米国郵政公社® (USPS®) のテクノロジーです。この DPV® を使用して住所の照合をすることによって、住所不完全のため不達 (UAA) という事態を減少させ、郵便コストや不正な住所情報に関連する他のビジネス コストを削減することができます。

> **注：** DPV® は U.S. アドレスに対してのみ有効です。

DPV® なしでの個々の住所検証では、そのストリート上にある複数の有効な住所までしか照合できません。例えば、USPS データは Maple Lane 上の住所が 500 から 1000 まであることを示しています。住所 610 Maple Ln の検証を試みます。DPV® なしで検証すると、この住所は 500 ～ 1000 の範囲にあるため有効と見なされます。ただし、実際には 610 Maple Ln という住所は存在しません。このストリートの区画にある家の番号は、608、609、613、および 616 です。DPV® 処理を使うと、610 Maple Ln が存在しないことが警告され、住所を訂正する措置をとることができます。

DPV® は、ターゲットを絞り込んだ郵便リストの作成に役立つ独特の住所属性も備えています。例えば、DPV® は郵送先が空き家かどうかを確認したり、それが郵便受取代行業 (CMRA) や私設私書箱の住所であることを識別できます。

DPV® は既存の住所の正確性を検証できますが、DPV® を使って住所録を作成することはできません。例えば、123 Elm Street Apartment 6 という住所が存在することは確認できますが、同じストリートに Apartment 7 があるかどうかを調べることはできません。住所録の生成を阻止するために、DPV® データベースには誤検出レコードが含まれています。誤検出レコードは、誤検出テーブルに存在する人為的に作られた住所です。DPV® クエリでマッチしなかった場合は、誤検出テーブルに対してクエリが実行されます。このテーブルにマッチする場合、DPV® の処理は停止します。

Early Warning System (EWS)

Early Warning System (EWS) は、月に 1 回更新される USPS データベースにまだ反映されていない、新規の住所や最近変更された住所に関する最新住所情報を提供します。EWS は USPS® データベースの郵便データの更新遅れによる、住所レコードの誤った情報提供を防ぐことができます。

米国郵便データベースが古いほど、住所が誤って変換される可能性は高くなります。米国郵便データベースにあるマッチングする住所が不正確な場合、有効な住所が誤った住所に変換され、壊れた住所が生成されます。

EWS データは、ZIP Code™、ストリート名、前置/後置方位記号、および接尾語に限定される、部分住所情報から構成されます。住所が米国郵便データベースの最新版には存在しない場合に限り、住所レコードに EWS を適用できます。

USPS® は、EWS ファイルを週に 1 回更新します。USPS® の Web サイト **https://ribbs.usps.gov/cassmass/documents/tech_guides/** から EWS ファイルをダウンロードできます。

Geocode Address Global

Geocode Address Global へのアクセスに API を使用する方法については、ジオコーディング ガイドを参照してください。

Geocode Address World

GeocodeAddress World は、サポートされている国のいずれかに位置する住所を受け取り、都市セントロイド (一部の国では郵便番号セントロイド) を返します。GeocodeAddress World は、ストリート住所レベルでのジオコーディングを行うことができません。住所レベルでのジオコーディングが必要な場合は、GeocodeAddress Global を使用してください。

GeocodeAddress World は、通常、GeocodeAddress Global で得られない国をカバーするための代替ジオコーダとして使用されます。例えば、主にオーストラリアの住所のジオコーディングに

関心があるためにオーストラリア ジオコーダのライセンスを取得したとします。しかし、手持ちのデータの一部のレコードにオーストラリア以外のロケーションが含まれています。このような場合、GeocodeAddress World を使えばオーストラリア以外のセントロイド ジオコードを得ることができ、それと共にオーストラリア ジオコーダでオーストラリアの住所に関するより正確なジオコードを得ることができます。その他のデータフローでは、GeocodeAddress World を最初のパスのジオコーダとして使用したうえでその結果を国固有のジオコーダにルーティングすることができます。最善の方法はビジネス案件と住所データの特性によって異なります。

Geocode Address World は、Enterprise Geocoding モジュールのオプションのコンポーネントです。Enterprise Geocoding モジュールの詳細については、**Enterprise Geocoding モジュール**（270ページ）を参照してください。

Enterprise Geocoding モジュールの World Geocoder 用データベース リソースの追加

新しいデータベース リソースをインストール、または既存のデータベース リソースを変更するたびに、Management Console で定義して、システム上で使用できるようにする必要があります。この手順では、World Geocoder 用 Enterprise Geocoding モジュールのデータベース リソースを追加または変更する方法について説明します。

Geocode Address World データベース リソースを作成するには

1. データベースをまだインストールしていない場合は、データベース ファイルをシステムにインストールしてください。データベースのインストール手順については、*『Spectrum™ Technology Platform インストール ガイド』*を参照してください。
2. Management Console で、**[リソース]** の下の **[Spectrum データベース]** を選択します。
3. 追加ボタン  をクリックして新しいデータベースを作成するか、既存のデータベース リソースを選択して編集ボタン  をクリックしてそのデータベースを変更します。また、コピーボタン  を使って既存のデータベース リソースをコピーする方法でも新しいデータベースを作成できます。
4. 新しいデータベースを作成する場合は、**[名前]** フィールドにデータベース リソースの名前を入力します。任意の名前にすることができます。既存のデータベースをコピーして新しいデータベースを作成する場合は、必要に応じてデフォルト名を変更してください。既存のデータベース リソースの名前を変更することはできません。その名前でデータベースを参照しているサービスやジョブがあると、動作しなくなるからです。
5. **[プール サイズ]** フィールドで、このデータベースで処理する同時要求の最大数を指定します。

   最適なプール サイズはモジュールによって異なります。一般的には、サーバーが搭載する CPU の数の半分から 2 倍のプール サイズを設定すると、最適な結果が得られます。ほとんどのモジュールに最適なプール サイズは CPU 数と同数です。例えば、サーバーが 4 つの CPU を搭載している場合は、プール サイズを 2 (CPU 数の半分) ～ 8 (CPU 数の 2 倍) の間で試すことができ、多くの場合、最適なサイズは 4 (CPU 数と同数) です。

6. **[モジュール]** フィールドで、[InternationalGeocoder World] を選択します。
7. **[タイプ]** フィールドで、[Geocode Address Global] を選択します。
   .SPD ファイルを展開して `\server\app\dataimport` フォルダに配置した場合、Spectrum はこれらのファイルを自動的に `\repository\datastorage` フォルダに追加します。[データベースの追加] 画面にデータセットのリストが表示されます。
8. データベースにリソースとして追加するデータセットを選択します。長いリストからデータセットを検索するには、[フィルタ] テキスト ボックスを使用します。
9. データベースを保存します。
10. 起動中の Enterprise Designer セッションがある場合は、[更新] ボタンをクリックすると、新しいステージが表示されます。

ジオコードの精度

Geocode Address World は、入力に含まれるデータに基づいて、最善のジオコーディングを自動的に提供します。都市と有効な郵便番号が含まれる場合は、郵便番号セントロイドが返されます。都市名と無効な郵便番号を指定した場合、または郵便番号なしで都市名を指定した場合、GeocodeAddress World はその都市の地理的セントロイドを返します。**地理的ジオコーディング**（286ページ）、および**郵便番号ジオコーディング**（284ページ）を参照してください。

Management Console では、地理的ジオコーディングまたは郵便番号ジオコーディングを選択できます。最良一致を選択することもできます。地理的ジオコーディングと郵便番号ジオコーディングの両方が可能な状態で [最良一致] が選択されている場合、地理的結果の精度が都市レベルまたはそれ以上 (つまり、G3 または G4 結果コード) であれば、近似一致の地理的候補が返されます。地理的ジオコーディング結果の精度が都市レベルに満たない (つまり、G1 または G2 結果コード) の場合、[最良一致] が選択されていると郵便番号ジオコーディングの (Z1) 結果が返される可能性があります。郵便番号ジオコーディングで結果が得られない場合は、最も近い地理的候補が返されます。

地理的ジオコーディングの結果コード、および**郵便番号ジオコーディングの結果コード**を参照してください。

郵便番号ジオコーディング

Geocode Address World は、郵便番号情報がその国から提供されている場合、郵便セントロイドにジオコーディングできます。郵便番号情報は、任意のデータ ソース (TomTom、GeoNames、または Pitney Bowes) から取得できます。Geocode Address **各国の郵便データの対象範囲**（308ページ）の郵便データの対象範囲の概要については、World を参照してください。国によっては、郵便番号ジオコーディングの方が、地理的ジオコーディングよりも正確な結果を生成する場合があります。

郵便番号レベルのジオコーディングは、以下の条件が満たされる場合に可能です。

- 入力住所に、有効な郵便番号が含まれている。
- データ ソースに、その国の郵便番号情報が含まれている。すべての国に郵便番号データがあるとは限りません。

Geocode Address World は、郵便番号ジオコーディングに対する複数の近似一致を返す場合があります。例えば、郵便番号 12180 は米国ニューヨーク州トロイに一致しますが、同じ郵便番号は他の複数の国に存在します。入力が郵便番号のみである場合、これらすべての候補が近似一致として返されます。

入力に地理的な住所要素 (国、州、地域、都市名など) が含まれる場合、Geocode Address World は、その情報を利用して、より正確な単一の近似一致を返すことができる可能性があります。地理的な住所コンテンツを使用して、郵便番号ジオコーディング結果の精度を高めたい場合は、以下の点を考慮してください。

> **注：** 個々の国は、TomTom、GeoNames、または Pitney Bowes のソースのいずれかからその郵便番号データを取得します。そのため、郵便番号データ ソース内に存在する地理コンテンツは、国によって異なります。例えば、都市名 (City) は、GeoNames 郵便番号データ ソースを使用する国においては近似一致の重み付け係数ですが、TomTom 郵便番号データ ソースを使用する国では、都市名は無視されます。TomTom、GeoNames、Pitney Bowes のデータ ソースの地理コンテンツについては、**データ ソースと対象範囲**（288ページ）を参照してください。

地理情報による郵便番号ジオコーディング

この郵便番号ジオコーディングの例では、入力住所に 41012 という有効な郵便番号と、Emilia Romagna (エミリア ロマーニャ) という州 (StateProvince) が含まれています。ストリートの住所が提供されていますが、これは郵便番号ジオコーディングでは無視されます。

Fornaci 40
Emilia Romagna
41012

イタリアの TomTom 郵便番号データ ソースにはStateProvinceが含まれているため、近似一致の評価時には Emilia Romagna という州が考慮されます。そのため、郵便番号 41012 に一致するイタリアのエミリア ロマーニャが、結果コード Z1 とともに単一の近似一致として返されます。郵便番号が 41012 である他の国の候補は、非近似一致として返されます。StateProvinceまたは国の情報が入力で指定されなかった場合、Geocode Address World は複数の近似一致を返します。5 桁の郵便番号 41012 は複数の国に存在するからです。

> **注：** 郵便番号ジオコーディング結果の精度を高めるには、郵便番号データ ソースに地理コンテンツが存在する必要があります。例えば、イタリアの TomTom 郵便番号データ ソースには、都市/町 (City) が含まれません。そのため、郵便番号 41012 とともに Carpi という都

市を入力すると、Geocode Address World は都市名を無視し、郵便番号 41012 に一致する複数の近似一致を返します (ITA という国名も指定した場合を除きます)。TomTom、GeoNames、Pitney Bowes のデータ ソースの地理コンテンツについては、**データ ソースと対象範囲**（288ページ）を参照してください。

地理的ジオコーディング

World は、行政区分 (町や村など) のセントロイドにジオコーディングできます。

World は、以下の条件が満たされる場合に、地理的レベルのジオコーディングが可能です。

- 入力住所に、正確な地理情報が含まれており、住所コンテンツとして有効な郵便番号は含まれていない。問題の住所に有効な郵便番号の入力情報が含まれている場合、World は、郵便番号ジオコーディングを試みます。
- データ ソースに、その国の地理的レベルの情報が含まれている。地理情報は、任意のデータ ソース (TomTom、GeoNames、または Pitney Bowes) から取得できます。
- 国名や ISO 国コードは必要ありませんが、含まれている場合にはそれらが一致する必要があります。国名が含まれている場合には、より適切な近似一致が得られる可能性があります。

Cityへの地理的ジオコーディング

この例では、入力住所に Vaihingen an der Enz という都市 (City) が含まれています。この例では国は指定されていません。ストリートの住所情報 (ストリート名と番号) は、地理的ジオコーディングの際には無視されます。

Muldenweg 2
Vaihingen an der Enz

World は、G3 の近似一致候補を返します。国が指定されなかった場合でも、World は、ドイツ (DEU) における 1 つの近似一致を特定します。

StateProvince: Baden-Württemberg
County: Ludwigsburg
City: Vaihingen an der Enz
Country: DEU
Result Code: G3
X: 8.95948
Y: 48.930059

よくある都市名に対する地理的ジオコーディング

この例では、入力住所に Venice という都市 (City) が含まれています。この都市名は、複数の国に存在しますが、入力には国が指定されていません。

St Marks Plaza
Venice

World は、近似一致候補としてイタリアのベニス(ベネチア)を選択します。イタリアのベニスは人口が多く (約 27 万人)、また、イタリアのヴェネト州の州都であるためです。他の国の Venice という名の複数の都市も、非近似一致として返される場合があります。Venice, ITA に対する近似一致候補は以下のとおりです。

StateProvince: Veneto
County: Venezia
City: Venice
Country: ITA
Result Code: G3
X: 12.33878
Y: 45.43434

州/省の略語に対する地理的ジオコーディング

この例では、入力住所に Rome という都市名と、米国ジョージア州の略語である GA が含まれています。州/省の略語が認められている国については、**州または省の略語**（325ページ）を参照してください。州の略語が使用されているため、国名を指定する必要はありません。

Rome, GA

World は、StateProvince を考慮して、Rome, Georgia USA (米国ジョージア州ローマ) に対する近似一致を返します。Rome, Italy (イタリアのローマ) の方がずっと大きな都市であり、イタリアの首都でもありますが、StateProvince(GA) が入力で指定されたため、これは非近似候補として返されます。

StateProvince: Georgia
County: Floyd
City: Rome
Country: USA
Result Code: G3
X: -85.16467
Y: 34.25704

Localityへの地理的ジオコーディング

この例では、入力住所に Altamira (アルタミラ) という地方と GRO という州の略語が含まれています。World は、GRO という州の略語を認識するため、国名は必要ありません。

City: Altamira
StateProvince: GRO

この例では、Altamira が Worldとして入力された場合でも、Locality は Altamira の (City) の近似一致を返します。GRO の (StateProvince) も返されます。StateProvince として Guerrero が入力された場合は、Guerrero (ゲレロ) が返されます。

StateProvince: GRO
City: ACAPULCO DE JUÁREZ
Locality: ALTAMIRA
Country: MEX
Result Code: G4
X: 99.87984
Y: 16.87637

住所は、個々の入力フィールドにフォーマットして入力するか、またはフォーマットせずに入力することができます (単一行入力)。フォーマットされていない入力のジオコーディングについては、単一行入力（326ページ）を参照してください。

地理的エリア

すべての国に行政区分があり、これらの行政エリアの多くは住所で使用されています。World は、それぞれが行政区分に対応する 4 つの AreaName を識別します。行政区分の命名方法と階層は国ごとに異なります。

- 地方
- 都市
- 郡
- 州/省

。

データ ソースと対象範囲

Geocode Address World は、複数のデータ ソースを参照して、その包括的な世界中の住所データベースを構築します。入力住所が、データ ソースの 1 つを使用して特定できない場合、Geocode Address World は、他のデータ ソースの 1 つを使用します。最も適切な候補が返されます。

データ ソース (地理データと郵便番号データの両方) は、以下の順序で使用されます。

- TomTom データ
- GeoNames データ
- Pitney Bowes World データ

Geocode Address World は、大陸に基づき 6 つのデータベースに分割されています。地理データと郵便番号データは、各住所辞書に統合され、地理的ジオコーディングと郵便番号ジオコーディングの両方をサポートします。

- Africa
- Asia
- Europe
- NorthAmerica
- Oceana
- SouthAmerica

地理的ジオコーディングの説明と例については、**地理的ジオコーディング**（286ページ）を参照してください。郵便番号ジオコーディングの説明と例については、**郵便番号ジオコーディング**（284ページ）を参照してください。

郵便番号ソース データは地理コンテンツにアクセス可能で、それによって郵便番号解析結果の精度を高めることができます。つまり、複数の国に同じ郵便番号が見つかった場合に、地理情報 (国名および行政区分) を使用して、近似一致を評価することができます。

郵便番号データのソースによって、以下の地理情報を使用して、郵便番号解析結果の精度を高めることができます。

- TomTom ソース: 国、StateProvince
- GeoNames ソース: 国、StateProvince および City
- Pitney Bowes World ソース: 国、StateProvince、County、City、および Locality

> **注：** Geocode Address World データ セットには、**http://www.geonames.org** 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 (**http://creativecommons.org/licenses/by/3.0/legalcode**) 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 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 PBSI 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。

国の対象範囲

Geocode Address World は、世界中のほとんどすべての国を対象範囲として含みます。対象範囲の精度とスコープは、提供されているデータ ソースの品質に依存します。郵便番号データを含む国もあれば、地理的対象範囲のみを含む国もあります。

TomTom、GeoNames、Pitney Bowes の地理データ ソースおよび郵便データ ソースの詳細については、**データ ソースと対象範囲**（288ページ）を参照してください。

国別の地理的対象範囲の全一覧については、**各国の地理的データの対象範囲**（290ページ）を参照してください。国別の郵便の対象範囲については、**各国の郵便データの対象範囲**（308ページ）を参照してください。

各国の地理的データの対象範囲

表 42 : 国名と地理的データの対象範囲

国名	ISO 3166 国コード	データソース	ヴィンテージ
アフガニスタン	AFG	GeoNames	2011.07
オーランド諸島	ALA	GeoNames	2011.07
アルバニア	ALB	TomTom	2011.06
アルジェリア	DZA	GeoNames	2011.07
アメリカ領サモア	ASM	GeoNames	2011.07
ANDORRA	AND	TomTom	2011.06
アンゴラ	AGO	TomTom	2011.06
アンギラ	AIA	GeoNames	2011.07
南極	ATA	GeoNames	2011.07
アンティグア・バーブーダ	ATG	GeoNames	2011.07
アルゼンチン	ARG	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
アルメニア	ARM	GeoNames	2011.07
アルバ	ABW	GeoNames	2011.07
オーストラリア	AUS	GeoNames	2011.07
オーストリア	AUT	TomTom	2011.06
アゼルバイジャン	AZE	GeoNames	2011.07
バハマ	BHS	GeoNames	2011.07
バーレーン	BHR	TomTom	2011.06
バングラデシュ	BGD	GeoNames	2011.07
バルバドス	BRB	GeoNames	2011.07
ベラルーシ	BLR	TomTom	2011.06
ベルギー	BEL	TomTom	2011.06
ベリーズ	BLZ	GeoNames	2011.07
ベナン	BEN	TomTom	2011.06
バミューダ	BMU	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
ブータン	BTN	GeoNames	2011.07
ボリビア	BOL	GeoNames	2011.07
ボネール島、シント・ユースタティウス島、およびサバ島	BES	GeoNames	2011.07
ボスニア・ヘルツェゴビナ	BIH	TomTom	2011.06
ボツワナ	BWA	TomTom	2011.06
ブーベ島	BVT	GeoNames	2011.07
ブラジル	BRA	TomTom	2011.06
イギリス領インド洋地域	IOT	GeoNames	2011.07
ブルネイ・ダルサラーム	BRN	TomTom	2011.06
ブルガリア	BGR	TomTom	2011.06
ブルキナファソ	BFA	TomTom	2011.06
ブルンジ	BDI	GeoNames	2011.07
カンボジア	KHM	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
カメルーン	CMR	TomTom	2011.06
カナダ	CAN	TomTom	2011.06
カーボベルデ	CPV	GeoNames	2011.07
ケイマン諸島	CYM	GeoNames	2011.07
中央アフリカ共和国	CAF	GeoNames	2011.07
チャド	TCD	GeoNames	2011.07
チリ	CHL	TomTom	2011.06
中国	CHN	GeoNames	2011.07
クリスマス島	CXR	GeoNames	2011.07
ココス (キーリング) 諸島	CCK	GeoNames	2011.07
コロンビア	COL	GeoNames	2011.07
コモロ	COM	GeoNames	2011.07
コンゴ	COG	TomTom	2011.06
コンゴ民主共和国	COD	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
クック諸島	COK	GeoNames	2011.07
COSTA RICA	CRI	GeoNames	2011.07
コートジボワール	CIV	GeoNames	2011.07
クロアチア (現地名: HRVATSKA)	HRV	TomTom	2011.06
キューバ	CUB	GeoNames	2011.07
キュラソー	CUW	GeoNames	2011.07
キプロス	CYP	GeoNames	2011.07
チェコ共和国	CZE	TomTom	2011.06
デンマーク	DNK	GeoNames	2011.07
ジブチ	DJI	GeoNames	2011.07
ドミニカ	DMA	GeoNames	2011.07
ドミニカ共和国	DOM	GeoNames	2011.07
エクアドル	ECU	GeoNames	2011.07
エジプト	EGY	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
エルサルバドル	SLV	GeoNames	2011.07
赤道ギニア	GNQ	GeoNames	2011.07
エリトリア	ERI	GeoNames	2011.07
エストニア	EST	TomTom	2011.06
エチオピア	ETH	GeoNames	2011.07
フォークランド諸島 (マルビナス)	FLK	GeoNames	2011.07
フェロー諸島	FRO	GeoNames	2011.07
フィジー	FJI	GeoNames	2011.07
フィンランド	FIN	TomTom	2011.06
フランス	FRA	TomTom	2011.06
FRENCH GUIANA	GUF	TomTom	2011.06
フランス領ポリネシア	PYF	GeoNames	2011.07
フランス領南方・南極地域	ATF	GeoNames	2011.07
ガボン	GAB	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
ガンビア	GMB	GeoNames	2011.07
グルジア	GEO	GeoNames	2011.07
ドイツ	DEU	TomTom	2011.06
ガーナ	GHA	TomTom	2011.06
ジブラルタル	GIB	GeoNames	2011.07
ギリシャ	GRC	TomTom	2011.06
グリーンランド	GRL	GeoNames	2011.07
グレナダ	GRD	GeoNames	2011.07
GUADELOUPE	GLP	TomTom	2011.06
グアム	GUM	GeoNames	2011.07
グアテマラ	GTM	GeoNames	2011.07
ガーンジー	GGY	GeoNames	2011.07
ギニア	GIN	GeoNames	2011.07
ギニアビサウ	GNB	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
ガイアナ	GUY	GeoNames	2011.07
ハイチ	HTI	GeoNames	2011.07
ハード島とマクドナルド諸島	HMD	GeoNames	2011.07
ホンジュラス	HND	GeoNames	2011.07
香港	HKG	TomTom	2011.06
ハンガリー	HUN	TomTom	2011.06
アイスランド	ISL	GeoNames	2011.07
インド	IND	GeoNames	2011.07
インドネシア	IDN	TomTom	2011.06
イラン・イスラム共和国	IRN	GeoNames	2011.07
イラク	IRQ	GeoNames	2011.07
アイルランド	IRL	TomTom	2011.06
マン島	IMN	GeoNames	2011.07
イスラエル	ISR	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
イタリア	ITA	TomTom	2011.06
ジャマイカ	JAM	GeoNames	2011.07
日本	JPN	GeoNames	2011.07
ジャージー	JEY	GeoNames	2011.07
ヨルダン	JOR	GeoNames	2011.07
カザフスタン	KAZ	GeoNames	2011.07
ケニア	KEN	TomTom	2011.06
キリバス	KIR	GeoNames	2011.07
朝鮮民主主義人民共和国	PRK	GeoNames	2011.07
大韓民国	KOR	GeoNames	2011.07
クウェート	KWT	TomTom	2011.06
キルギス	KGZ	GeoNames	2011.07
ラオス人民民主共和国	LAO	GeoNames	2011.07
ラトビア	LVA	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
レバノン	LBN	GeoNames	2011.07
レソト	LSO	TomTom	2011.06
リベリア	LBR	GeoNames	2011.07
大リビア・アラブ社会主義人民ジャマーヒリーヤ国	LBY	GeoNames	2011.07
LIECHTENSTEIN	LIE	GeoNames	2011.07
リトアニア	LTU	TomTom	2011.06
LUXEMBOURG	LUX	TomTom	2011.06
マカオ	MAC	TomTom	2011.06
マケドニア旧ユーゴスラビア共和国	MKD	TomTom	2011.06
マダガスカル	MDG	GeoNames	2011.07
マラウィ	MWI	TomTom	2011.06
マレーシア	MYS	TomTom	2011.06
モルディブ	MDV	GeoNames	2011.07
マリ	MLI	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
マルタ	MLT	TomTom	2011.06
マーシャル諸島	MHL	GeoNames	2011.07
MARTINIQUE	MTQ	GeoNames	2011.07
モーリタニア	MRT	TomTom	2011.06
モーリシャス	MUS	TomTom	2011.06
MAYOTTE	MYT	GeoNames	2011.07
メキシコ	MEX	TomTom	2011.06
ミクロネシア連邦	FSM	GeoNames	2011.07
モルドバ共和国	MDA	TomTom	2011.06
MONACO	MCO	GeoNames	2011.07
モンゴル	MNG	GeoNames	2011.07
モンテネグロ	MNE	TomTom	2011.06
モントセラト	MSR	GeoNames	2011.07
モロッコ	MAR	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
モザンビーク	MOZ	TomTom	2011.06
ミャンマー	MMR	GeoNames	2011.07
ナミビア	NAM	GeoNames	2011.07
ナウル	NRU	GeoNames	2011.07
ネパール	NPL	GeoNames	2011.07
オランダ	NLD	TomTom	2011.06
オランダ領アンティル	ANT	Pitney Bowes	C.2006
ニューカレドニア	NCL	GeoNames	2011.07
ニュージーランド	NZL	GeoNames	2011.07
ニカラグア	NIC	GeoNames	2011.07
ニジェール	NER	TomTom	2011.06
ナイジェリア	NGA	TomTom	2011.06
ニウエ	NIU	GeoNames	2011.07
ノーフォーク島	NFK	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
北マリアナ諸島	MNP	GeoNames	2011.07
ノルウェー	NOR	TomTom	2011.06
オマーン	OMN	TomTom	2011.06
パキスタン	PAK	GeoNames	2011.07
パラオ	PLW	GeoNames	2011.07
被占領パレスチナ地域	PSE	GeoNames	2011.07
パナマ	PAN	GeoNames	2011.07
パプアニューギニア	PNG	GeoNames	2011.07
パラグアイ	PRY	GeoNames	2011.07
ペルー	PER	GeoNames	2011.07
フィリピン	PHL	TomTom	2011.06
ピトケアン	PCN	GeoNames	2011.07
ポーランド	POL	TomTom	2011.06
ポルトガル	PRT	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
プエルトリコ	PRI	GeoNames	2011.07
カタール	QAT	TomTom	2011.06
レユニオン	REU	TomTom	2011.06
ルーマニア	ROU	TomTom	2011.06
ロシア連邦	RUS	TomTom	2011.06
ルワンダ	RWA	GeoNames	2011.07
サン・バルテルミー島	BLM	GeoNames	2011.07
セントヘレナ・アセンションおよびトリスタン・ダ・クーニャ	SHN	GeoNames	2011.07
セントクリストファー・ネイビス	KNA	GeoNames	2011.07
セントルシア	LCA	GeoNames	2011.07
フランス領サン・マルタン島	MAF	GeoNames	2011.07
サンピエール島とミクロン島	SPM	GeoNames	2011.07
セントビンセントおよびグレナディーン諸島	VCT	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
サモア	WSM	GeoNames	2011.07
サンマリノ	SMR	TomTom	2011.06
サントメ・プリンシペ	STP	GeoNames	2011.07
サウジアラビア	SAU	TomTom	2011.06
セネガル	SEN	TomTom	2011.06
セルビア	SRB	TomTom	2011.06
セーシェル	SYC	GeoNames	2011.07
シエラレオネ	SLE	GeoNames	2011.07
シンガポール	SGP	TomTom	2011.06
オランダ領シント・マールテン島	SXM	GeoNames	2011.07
スロバキア (スロバキア共和国)	SVK	TomTom	2011.06
スロベニア	SVN	TomTom	2011.06
ソロモン諸島	SLB	GeoNames	2011.07
ソマリア	SOM	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
南アフリカ	ZAF	GeoNames	2011.07
サウスジョージア・サウスサンドウィッチ諸島	SGS	GeoNames	2011.07
スペイン	ESP	TomTom	2011.06
スリランカ	LKA	GeoNames	2011.07
スーダン	SDN	GeoNames	2011.07
スリナム	SUR	GeoNames	2011.07
スヴァールバル諸島およびヤンマイエン島	SJM	GeoNames	2011.07
スワジランド	SWZ	TomTom	2011.06
スウェーデン	SWE	TomTom	2011.06
スイス	CHE	TomTom	2011.06
シリア・アラブ共和国	SYR	GeoNames	2011.07
台湾	TWN	TomTom	2011.06
タジキスタン	TJK	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
タンザニア連合共和国	TZA	TomTom	2011.06
タイ	THA	TomTom	2011.06
東ティモール	TLS	GeoNames	2011.07
トーゴ	TGO	TomTom	2011.06
トケラウ	TKL	GeoNames	2011.07
トンガ	TON	GeoNames	2011.07
トリニダード・トバゴ	TTO	GeoNames	2011.07
チュニジア	TUN	GeoNames	2011.07
トルコ	TUR	TomTom	2011.06
トルクメニスタン	TKM	GeoNames	2011.07
タークス・カイコス諸島	TCA	GeoNames	2011.07
ツバル	TUV	GeoNames	2011.07
ウガンダ	UGA	TomTom	2011.06
ウクライナ	UKR	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
アラブ首長国連邦	ARE	TomTom	2011.06
英国	GBR	TomTom	2011.06
米国	USA	GeoNames	2011.07
合衆国領有小離島	UMI	GeoNames	2011.07
ウルグアイ	URY	TomTom	2011.06
ウズベキスタン	UZB	GeoNames	2011.07
バヌアツ	VUT	GeoNames	2011.07
バチカン市国 (法王聖座)	VAT	GeoNames	2011.07
ベネズエラ	VEN	GeoNames	2011.07
ベトナム	VNM	GeoNames	2011.07
イギリス領ヴァージン諸島	VGB	GeoNames	2011.07
アメリカ領ヴァージン諸島	VIR	GeoNames	2011.07
ウォリス・フツナ	WLF	GeoNames	2011.07
西サハラ	ESH	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
イエメン	YEM	GeoNames	2011.07
ザンビア	ZMB	TomTom	2011.06
ジンバブエ	ZWE	GeoNames	2011.07

各国の郵便データの対象範囲

表 43 : 国名と郵便データの対象範囲

国名	ISO 3166 国コード	データソース	ヴィンテージ
アルジェリア	DZA	Pitney Bowes	C.2006
アメリカ領サモア	ASM	GeoNames	2011.07
ANDORRA	AND	TomTom	2011.06
アルゼンチン	ARG	GeoNames	2011.07
アルメニア	ARM	Pitney Bowes	C.2006
オーストラリア	AUS	GeoNames	2011.07
オーストリア	AUT	TomTom	2011.06
アゼルバイジャン	AZE	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
バーレーン	BHR	Pitney Bowes	C.2006
バングラデシュ	BGD	GeoNames	2011.07
ベラルーシ	BLR	Pitney Bowes	C.2006
ベルギー	BEL	TomTom	2011.06
バミューダ	BMU	Pitney Bowes	C.2006
ボスニア・ヘルツェゴビナ	BIH	Pitney Bowes	C.2006
ブラジル	BRA	TomTom	2011.09
イギリス領インド洋地域	IOT	Pitney Bowes	C.2006
ブルネイ・ダルサラーム	BRN	Pitney Bowes	C.2006
ブルガリア	BGR	GeoNames	2011.07
カンボジア	KHM	Pitney Bowes	C.2006
カナダ	CAN	TomTom	2011.09
カーボベルデ	CPV	Pitney Bowes	C.2006
チリ	CHL	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
中国	CHN	Pitney Bowes	C.2006
クリスマス島	CXR	Pitney Bowes	C.2006
ココス (キーリング) 諸島	CCK	Pitney Bowes	C.2006
COSTA RICA	CRI	Pitney Bowes	C.2006
クロアチア (現地名: HRVATSKA)	HRV	GeoNames	2011.07
キューバ	CUB	Pitney Bowes	C.2006
キプロス	CYP	Pitney Bowes	C.2006
チェコ共和国	CZE	TomTom	2011.06
デンマーク	DNK	GeoNames	2011.07
ドミニカ共和国	DOM	GeoNames	2011.07
エクアドル	ECU	Pitney Bowes	C.2006
エジプト	EGY	Pitney Bowes	C.2006
エルサルバドル	SLV	Pitney Bowes	C.2006
エストニア	EST	TomTom	2011.06

国名	ISO 3166 国コード	データソース	ヴィンテージ
エチオピア	ETH	Pitney Bowes	C.2006
フォークランド諸島 (マルビナス)	FLK	Pitney Bowes	C.2006
フェロー諸島	FRO	GeoNames	2011.07
フィンランド	FIN	TomTom	2011.06
フランス	FRA	TomTom	2011.06
FRENCH GUIANA	GUF	GeoNames	2011.07
フランス領ポリネシア	PYF	Pitney Bowes	C.2006
グルジア	GEO	Pitney Bowes	C.2006
ドイツ	DEU	TomTom	2011.06
ギリシャ	GRC	TomTom	2011.06
グリーンランド	GRL	GeoNames	2011.07
GUADELOUPE	GLP	GeoNames	2011.07
グアム	GUM	GeoNames	2011.07
グアテマラ	GTM	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
ガーンジー	GGY	GeoNames	2011.07
ギニア	GIN	Pitney Bowes	C.2006
ギニアビサウ	GNB	Pitney Bowes	C.2006
ハイチ	HTI	Pitney Bowes	C.2006
ホンジュラス	HND	Pitney Bowes	C.2006
ハンガリー	HUN	GeoNames	2011.07
アイスランド	ISL	GeoNames	2011.07
インド	IND	GeoNames	2011.07
インドネシア	IDN	TomTom	2011.06
イラン・イスラム共和国	IRN	Pitney Bowes	C.2006
イラク	IRQ	Pitney Bowes	C.2006
アイルランド	IRL	Pitney Bowes	C.2006
マン島	IMN	GeoNames	2011.07
イスラエル	ISR	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
イタリア	ITA	TomTom	2011.06
ジャマイカ	JAM	Pitney Bowes	C.2006
日本	JPN	GeoNames	2011.07
ジャージー	JEY	GeoNames	2011.07
ヨルダン	JOR	Pitney Bowes	C.2006
カザフスタン	KAZ	Pitney Bowes	C.2006
ケニア	KEN	Pitney Bowes	C.2006
大韓民国	KOR	Pitney Bowes	C.2006
クウェート	KWT	Pitney Bowes	C.2006
キルギス	KGZ	Pitney Bowes	C.2006
ラオス人民民主共和国	LAO	Pitney Bowes	C.2006
ラトビア	LVA	TomTom	2011.06
レバノン	LBN	Pitney Bowes	C.2006
レソト	LSO	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
リベリア	LBR	Pitney Bowes	C.2006
LIECHTENSTEIN	LIE	GeoNames	2011.07
リトアニア	LTU	TomTom	2011.06
LUXEMBOURG	LUX	GeoNames	2011.07
マケドニア旧ユーゴスラビア共和国	MKD	GeoNames	2011.07
マダガスカル	MDG	Pitney Bowes	C.2006
マレーシア	MYS	GeoNames	2011.07
モルディブ	MDV	Pitney Bowes	C.2006
マルタ	MLT	Pitney Bowes	C.2006
マーシャル諸島	MHL	GeoNames	2011.07
MARTINIQUE	MTQ	GeoNames	2011.07
MAYOTTE	MYT	GeoNames	2011.07
メキシコ	MEX	TomTom	2011.06
ミクロネシア連邦	FSM	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
モルドバ共和国	MDA	GeoNames	2011.07
MONACO	MCO	GeoNames	2011.07
モンゴル	MNG	Pitney Bowes	C.2006
モロッコ	MAR	TomTom	2011.06
モザンビーク	MOZ	Pitney Bowes	C.2006
ミャンマー	MMR	Pitney Bowes	C.2006
ネパール	NPL	Pitney Bowes	C.2006
オランダ	NLD	TomTom	2011.06
ニューカレドニア	NCL	Pitney Bowes	C.2006
ニュージーランド	NZL	GeoNames	2011.07
ニカラグア	NIC	Pitney Bowes	C.2006
ニジェール	NER	Pitney Bowes	C.2006
ナイジェリア	NGA	Pitney Bowes	C.2006
ノーフォーク島	NFK	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
北マリアナ諸島	MNP	GeoNames	2011.07
ノルウェー	NOR	TomTom	2011.06
オマーン	OMN	Pitney Bowes	C.2006
パキスタン	PAK	GeoNames	2011.07
パラオ	PLW	Pitney Bowes	C.2006
パプアニューギニア	PNG	Pitney Bowes	C.2006
パラグアイ	PRY	Pitney Bowes	C.2006
フィリピン	PHL	GeoNames	2011.07
ピトケアン	PCN	Pitney Bowes	C.2006
ポーランド	POL	TomTom	2011.06
ポルトガル	PRT	TomTom	2011.06
プエルトリコ	PRI	GeoNames	2011.07
レユニオン	REU	GeoNames	2011.07
ルーマニア	ROU	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
ロシア連邦	RUS	TomTom	2011.06
セントヘレナ・アセンションおよびトリスタン・ダ・クーニャ	SHN	Pitney Bowes	C.2006
サンピエール島とミクロン島	SPM	GeoNames	2011.07
サンマリノ	SMR	TomTom	2011.06
サウジアラビア	SAU	Pitney Bowes	C.2006
セネガル	SEN	Pitney Bowes	C.2006
シンガポール	SGP	TomTom	2011.06
スロバキア (スロバキア共和国)	SVK	TomTom	2011.06
スロベニア	SVN	TomTom	2011.06
南アフリカ	ZAF	GeoNames	2011.07
サウスジョージア・サウスサンドウィッチ諸島	SGS	Pitney Bowes	C.2006
スペイン	ESP	TomTom	2011.06
スリランカ	LKA	GeoNames	2011.07

国名	ISO 3166 国コード	データソース	ヴィンテージ
スーダン	SDN	Pitney Bowes	C.2006
スワジランド	SWZ	Pitney Bowes	C.2006
スウェーデン	SWE	GeoNames	2011.07
スイス	CHE	TomTom	2011.06
台湾	TWN	TomTom	2011.06
タジキスタン	TJK	Pitney Bowes	C.2006
タイ	THA	TomTom	2011.06
東ティモール	TLS	Pitney Bowes	C.2006
チュニジア	TUN	Pitney Bowes	C.2006
トルコ	TUR	TomTom	2011.06
トルクメニスタン	TKM	Pitney Bowes	C.2006
タークス・カイコス諸島	TCA	Pitney Bowes	C.2006
ウクライナ	UKR	Pitney Bowes	C.2006
アラブ首長国連邦	ARE	Pitney Bowes	C.2006

国名	ISO 3166 国コード	データソース	ヴィンテージ
英国	GBR	TomTom	2011.06
米国	USA	TomTom	2011.06
ウルグアイ	URY	Pitney Bowes	C.2006
ウズベキスタン	UZB	Pitney Bowes	C.2006
バチカン市国 (法王聖座)	VAT	TomTom	2011.06
ベネズエラ	VEN	Pitney Bowes	C.2006
ベトナム	VNM	Pitney Bowes	C.2006
アメリカ領ヴァージン諸島	VIR	GeoNames	2011.07
ウォリス・フツナ	WLF	Pitney Bowes	C.2006
西サハラ	ESH	Pitney Bowes	C.2006
ザンビア	ZMB	Pitney Bowes	C.2006

ジオコーディングのシナリオ

Enterprise Manager を使用することによって、ビジネス要件や、データの性質と品質に適したデータフローを作成できます。

Geocode Address World を最終ジオコーディング パスとして使用した複数国ステージ

入力を複数のパスでジオコーディングすることによって、結果を最適化できる場合があります。一般的に、最初のパスにはより厳しいマッチング条件を適用できます。続くジオコーディング パスでは、前のパスで近似一致候補を返すことのできなかった住所に対して、より緩和されたマッチング条件を適用できます。この方法によって、品質の高い住所に対しては正確な一致を返し、正確さに欠ける住所や、対象範囲のレベルが包括的ではない国の住所に対しては、できる限り最良の結果を返すことができます。

以下のシナリオを例として考えます。

- 入力ファイルに、6 カ国の住所が含まれているとします。6 カ国とは、アルゼンチン (ARG)、ブラジル (BRA)、メキシコ (MEX)、チリ (CHL)、ベネズエラ (VEN)、パナマ (PAN) です。
- これらの国のうちの 3 カ国 (ARG、BRA、MEX) のジオコーダは、複数国ステージで展開されています。
- Geocode Address World は、国固有のジオコーダによって特定できなかった住所をジオコーディングするための別個のステージに展開されています。
- ステージでは、Conditional Router (と、場合によっては Stream Combiner) を使用して、ジオコーディング フローを管理します。

1. 入力を複数国ステージに読み込みます。ジオコーディングされた住所は、ファイルに書き出すか、Stream Combiner に送信することもできます。
2. 一部の住所は、ステップ 1 ではジオコーディングできません。こうした状況が起こり得る理由は、住所が CHL、VEN、または PAN のものであり、これらの国のジオコーダが最初のステージにないためです。あるいは、住所の入力に誤りやあいまいな部分があるために、最初のステージでは近似一致候補を返すことができなかったという場合もあり得ます。ジオコーディングされなかったこれらの住所は、Geocode Address World ステージへと送られます。
3. 住所は、Geocode Address World によって郵便番号の精度または地理的な精度にジオコーディングできます。正しくジオコーディングされた住所は、ファイルに書き出すか、Stream Combiner に送信することもできます。

   郵便ジオコーディングされた候補は、結果コードが Z1 になります。郵便ジオコーディングの結果は、郵便番号システムが確実に定められている国においては非常に正確である可能性があります。**郵便番号ジオコーディング**（284ページ）を参照してください。地理的候補には、G 結果コードが付与されます (例えば、町/都市が一致した場合は G3)。**地理的ジオコーディング**（286ページ）を参照してください。

4. Stream Combiner (データフローにおいて使用した場合) は、ジオコーディングされたすべての住所を結合し、ファイルに書き出すか、または、さらなる処理に向けて送ることができます。

これは、シナリオの 1 つです。Enterprise Manager を使用することによって、ニーズに適した、より複雑なデータフローを設計することができます。

最初のジオコーディング パスとしての Geocode Address World の使用

Geocode Address World を最初のジオコーディング パスとする方法を利用することもできます。

次のような状況を想定します。

- 通常、住所に国が指定されていません (ただし、なかには指定されているものもあります)。
- 一部の住所にはストリートと都市の住所情報のみが含まれています。
- 一部の国については国固有のジオコーダがありますが、すべての国のものが揃っているわけではありません。
- ジオコーディング プロセスを管理するためにメイン データフローにサブフローを組み合わせて使用します。

以下の動作を実行するデータフロー (場合によってはサブフローが付属するもの) を使用します。以下の手順は、サンプル データフローを簡単に説明したものです。

1. 入力を複数国ステージ内に読み込みます。このステージには Geocode Address World も含まれています。都市名 (および場合によっては米国住所の州名) に基づき、各住所について、可能性のある 1 つ以上の近似一致候補を複数の異なる国から生成できます。ここで、入力住所に国が含まれていない場合であっても、各候補に国が関連付けられます。
2. 国固有のジオコーダが利用できる場合、候補はそのジオコーダに送られます。この処理には、Conditional Router、Stream Combiner をはじめとする他の Spectrum™ Technology Platform制御ステージが必要です。入力住所の完全性と国固有のジオコーダの機能に応じて、各候補はストリート (S 結果コード)、地理的 (G 結果コード)、または郵便番号 (Z 結果コード) レベルにジオコーディングされます。
3. 国固有のジオコーダが利用できない場合、候補は Geocode Address World にルーティングされ、そこで地理的レベルまたは郵便番号レベルにジオコーディングできます。
4. すべてのサブフローからの候補が組み合わされ、数々の基準を使用してランク付けされます。ランク付けは、都市の人口 (都市ランク)、一致の精度 (ストリート、地理的、郵便番号)、ユーザの地域からの近さなどの基準に従って行うことができます。

入力

Geocode Address World は、入力として住所を受け取ります。最大のパフォーマンスと最良のマッチ結果を得るには、入力住所リストが可能な限り完全で、綴りの誤りや不完全な住所がなく、できる限り郵便当局の規格に従っている必要があります。多くの郵便当局が、その国の住所規格に関する情報を掲載した Web サイトを提供しています。

> **注：** 国名または 2 文字か 3 文字の ISO 国コードは省略可能です。国名を省略すると、Geocode Address World は、他の入力情報に基づいて最も適切な候補を返します。

入力フィールド

以下の表に、Geocode Address World の入力のフォーマットとレイアウトに関する情報を示します。

> **注：** `DataTable`クラスを使用して入力を指定します。詳細については、『Spectrum™ Technology PlatformAPI ガイド』を参照してください。

表 44 : Geocode Address World の入力データ

columnName	書式	説明
AddressLine1	String	最初の住所行。次の例では 4360 DUKES RD です。 4360 DUKES RD **KALGOORLIE WA 6430**
AddressLine2	String	2 行からなる住所の 2 行目の住所行。次の例では Level 6 51 Jacobson St です。 26 WELLINGTON ST E **SUITE 500** TORONTO ON M5E 1S2 このフィールドは、オーストラリア、オーストリア、ベルギー、ブラジル、デンマーク、フィンランド、フランス、ドイツ、アイルランド、イタリア、リヒテンシュタイン、ルクセンブルク、マレーシア、オランダ、ポーランド、ポルトガル、スペイン、スウェーデン、スイス、およびタイでは使用されません。
City	String	都市または町の名前。入力住所には正式な都市名を使用してください。最適なジオコーディング結果が得られます。 タイの住所では、このフィールドに従属する地区 (タムボン) が格納されます。

columnName	書式	説明
County	String	国により、次のいずれかの名前。 • **使用しない** — AUT、BRA、CAN、FIN、GBR、MYS、PRT、SGP • **郡 (Department)** — FRA • **地区 (District) (amphoe)** — THA • **郡 (District) (fylke/counties)** — NOR • **地区 (District) (poviat)** — POL • **コミューン (Kommun)** — SWE • **郡 (Kreis)** — DEU • **地方自治体 (Local Government Authority: LGA)** — AUS • **州 (Province)** — BEL、CHE、DNK、ESP、IRL、ITA、LIE、LUX、NLD • **地方行政区画 (Region)** — NZL
FirmName	String	会社名または場所の名前。次の例では PITNEY BOWES です。 **PITNEY BOWES** 4360 DUKES RD KALGOORLIE WA 6430
LastLine	String	住所の最終行。次の例では KALGOORLIE WA 6430 です。 4360 DUKES RD **KALGOORLIE WA 6430**
Locality	String	国により、次のいずれかの名前。 • **使用しない** — AUS、AUT、BEL、CHE、DEU、DNK、FIN、FRA、IRL、LIE、LUX、MYS、NLD、NOR、POL、SGP、SWE、THA • **散布エリア (DA) および列挙エリア (EA)** — CAN • **地方 (Locality)** — BRA、GBR、ITA、PRT • **郊外 (Suburb)** — NZL
PostalCode	String	各国の標準フォーマットで表記された郵便番号。

columnName	書式	説明
StateProvince	String	国により、次のいずれかの名前。 • **使用しない** — BEL、CHE、DNK、IRL、LIE、LUX、NLD、NOR、SGP • **連邦州 (Bundesland)** — DEU • **州 (Province)** — CAN • **県 (Province) (changwat)** — THA • **県 (Province) (voivodship)** — POL • **地域 (Region)** — AUT、ESP、FRA、GBR、NZL、PRT • **地域 (Region) (län)** — FIN • **地域 (Region) (lan)** — SWE • **州 (State)** — AUS、BRA • **州 (State) (negeri)** — MYS
Country	String	2 文字または 3 文字の ISO 国コード。このフォールドはオプションです。国名を省略すると、Geocode Address World は、取得できる最も適切な候補を他の入力情報に基づいて返します。 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。

住所のエイリアス

一部の国には、行政上の名称が複数存在します。例えば、都市や町には正式名称がありますが、その他に、同じ都市や町に対して、一般的に使用されているものの正式ではない別名が存在する場合があります。ソース データにエイリアス情報が存在する場合、World では、データベースにこのエイリアスを含めます。World は、入力住所に別名が使用されている場合に正しくジオコーディングすることができます。

言語のエイリアス（324ページ） もサポートされています。

言語のエイリアス

一部の国では、複数の言語が正式に、または一般的に使用されています。例えば、同じ町に対し、ドイツ語の名前とイタリア語の名前がどちらも一般的に知られている場合があります。ソース データに言語のエイリアス情報が存在する場合、World では、データベースにこのエイリアスを含めます。World は、入力住所に別の言語による名前が使用されている場合に正しくジオコーディングすることができます。

エイリアスは、StateProvince 州/省から Locality 地方までのすべての行政レベルに対して存在します。地理データに関連付けられた行政レベルについては、**行政区分と郵便番号**（325ページ）を参照してください。

住所のエイリアス（324ページ）も、一般的に使用される、別の行政区域に対してサポートされています。

州または省の略語

一部の国において、州または省は住所の重要な部分であり、この住所要素は省略形で示される場合がよくあります。一部の国に対し、World では、州/省の略語が認められています。例えば、米国には各州を表す 2 文字の略語が存在します (カリフォルニア州に対する CA など)。同様に、オランダでは、州の略語 (ヘルデルラント州に対する GLD など) が認められています。

World は、以下の国における州/省の略語を認識します。

表 45 : 国に対する州/省の略語のサポート

国名	州または省の区分	例
オーストラリア (AUS)	StateProvince (州)	NSW (ニュー サウス ウェールズ州の略語)
カナダ (CAN)	StateProvince (県)	AB (アルバータ州の略語)
イタリア (ITA)	County (県)	MO (モデナ県の略語)
メキシコ (MEX)	StateProvince (州)	JA (ハリスコ州の略語)
オランダ (NLD)	County (州)	FR (フリースラント州の略語)
米国 (USA)	StateProvince (州)	CA (カリフォルニア州の略語)

World は、これらの州または省の略語を評価することにより、さらに適切な近似一致を特定します。この機能を説明する例については、州/省の略語に対する地理的ジオコーディング（287ページ）を参照してください。

行政区分と郵便番号

一般的な入力住所は、ストリートの住所、行政区分、および郵便番号の情報で構成されます。World は地理的または郵便ジオコーディングの際に、行政区分と郵便番号を使用します。

- StateProvince(州または省)
- County(郡、地域、または地区)
- City(町または市)
- Locality(地方、郊外、または村)
- 郵便番号

行政区分の指定は国によって異なります。例えば、Localityには、国に応じて地方 (locality)、郊外 (suburb)、または区域 (barrio) が含まれます。StateProvinceには州 (state)、省 (province)、地域 (region) などの名前を、その国に合わせて使用します。州や省の略語が Worldでどのように変換されるかについては、**州または省の略語**（325ページ）を参照してください。

どの国でもすべての行政区分が住所規約に使用されているわけではありません。例えば、米国では通常County(郡) を住所に使用しませんが、いくつかの国では住所の重要な一部として Countyを使用します。

入力データに郵便番号が含まれる場合、World はソース データにその国の郵便番号データが含まれるという前提で、これを郵便番号ジオコーディングに使用できます。

入力に関する推奨事項

入力レコードを適切に用意し、理解することによって、World の結果を最適化することができます。以下のガイドラインに従ってください。

- できる限り完全で正確な住所を入力します。入力住所に誤りがあっても、World はその住所をジオコーディングできる場合がありますが、複数の一致候補が得られたり、非近似一致が得られたりする可能性が生じます。入力住所が不完全であったり不正確であったりした場合に、それを確認して修正することができれば、より良い結果を得ることができます。
- 郵便番号がわかる場合は、それを入力住所に含めます。これは必須ではありませんが、郵便番号があれば、World は郵便番号ジオコーディングを実行できます。これによって、国や、他の住所要素の完全性と精度にも依存しますが、一部の住所に対してより正確な結果が得られる場合があります。
- 国名または正式な3 文字または 2 文字の ISO 国コードを入力住所に含めます。これは必須ではありませんが、これがあれば World は、異なる国に存在する類似の住所や都市名を区別できる場合があります。
- 入力住所を一貫した形式でフォーマットします。World は、多様なフォーマットの入力住所を処理でき、また、フォーマットされていない (単一行の) 入力も処理できます。しかし、入力住所が一貫した形式でフォーマットされており、国固有の住所規約に従っているならば、より正確かつ高速に結果を得ることができます。住所が単一行に入力されている (フォーマットされていない) 場合でも、住所要素が一貫した順序で並んでいれば、より良い結果とパフォーマンスが得られる可能性があります。フォーマットされていない住所の入力には、AddressLine1 の入力エリアを使用します。このサービスの詳細については、**単一行入力**（326ページ）を参照してください。

単一行入力

住所入力は、個別の入力フィールドに合わせて書式設定することも、単一行として行うこともできます。単一行入力には、AddressLine1 を使用します。

単一行の地理的ジオコーディング

この例では、フォーマットされていない (単一行の) 入力が使用されています。World は、単一行入力を分析して、地理的な住所要素 (この例では Graz) を特定し、続いて地理的セントロイドにジオコーディングします。MainAddress (ストリート情報) は使用しません。

Sackstraße 10 Graz

World は、City の一致に基づいて地理的な近似一致候補を返します。国が指定されなかった場合でも、World は、オーストリア (AUT) における 1 つの近似一致を特定します。

: SteirmarkStateProvince
County: Graz (Stadt)
City: Graz
Country: AUT
Result Code: G3
X: 15.44172
Y: 47.06792

入力住所が正確である場合は、フォーマットされていない入力に対しても、フォーマットされている入力に匹敵するマッチ率を得ることができます。ただし、フォーマットされていない住所のジオコーディングは通常、フォーマットされている住所のジオコーディングよりもパフォーマンスは低くなります。

国が指定されている場合の単一行郵便番号ジオコーディング

この例では、単一行に住所が入力されており、郵便番号が提供されています。国としてオーストリア (AUT) も指定されています。ストリートの住所も入力されていますが、これはオーストリアの指定によって無視されます。

Alpenstraße 117 5020 AUT

オーストリアでは、郵便番号セントロイド近似一致候補が返されます (結果コードは Z1)。入力において国 (AUT) が指定されているため、国が必ず一致する必要があり、オーストリアにおいてその郵便番号に対応する単一の近似一致が返されます。他の国における郵便番号が 5020 である非近似一致も返されます。

StateProvince: Salzburg
Country: AUT
Postcode: 5020
Result Code: Z1
X: 13.04685
Y: 47.80262

オプション

ジオコーディング オプション

以下の表に、特定の場所の座標を決定する方法を制御するためのオプションを示します。

表 46 : ジオコーディング オプション

オプション名	説明
CoordinateSystem	座標系は、空間におけるポイントの位置を一意に表すリファレンス システムです。カルテシアン (二次元) 座標、測地 (地理) 座標などが、ユークリッド幾何学に基づくリファレンス システムとして挙げられます。Spectrum™ Technology Platform は、European Petroleum Survey Group (EPSG) によって認識されるシステムをサポートしています。 次のいずれかを選択します。 **EPSG:4283** GDA94 座標系とも呼ばれます。 **EPSG:4326** WGS84 座標系とも呼ばれます。こちらがデフォルトです。

マッチング オプション

表 47 : マッチング オプション

オプション名	説明
KeepMultimatch	住所がデータベース内の複数の候補に一致する場合に結果を返すかどうかを指定します。このオプションを選択しない場合、複数の候補に一致する住所のジオコーディングは失敗します。 このオプションを選択する場合は、[最大候補数] オプション (下記を参照) を使用して返す候補の最大数を指定します。 **Y** 複数の候補が見つかった場合に候補を返します。こちらがデフォルトです。 **N** 候補を返しません。複数の候補が見つかる住所のジオコーディングは失敗します。

オプション名	説明
MaxCandidates	KeepMultimatch=Y を指定した場合、返す候補の最大数をこのオプションで指定します。 デフォルト値は 1 です。
CloseMatchesOnly	近似一致候補であるジオコード結果のみを返すかどうかを指定します。例えば、10 個の候補があり、そのうちの 2 個が近似一致である場合、このオプションを有効にすると、10 個全部ではなく 2 個の近似一致のみが候補として返されます。 **Y** 近似一致のみを返します。 **N** 近似一致のみを返しません。こちらがデフォルトです。

データ オプション

[データ] タブを使って、ジオコーディングに使うデータベースを指定できます。データベースには、指定の住所のジオコードを決定するために必要な住所とジオコード データが格納されています。データは、郵便当局や地理データ サプライヤから取得された住所とジオコーディング データに基づきます。

> **注：** EGM モジュールにより管理タスクが Web ベースの Management Console に移行されると、オプションのレベルで、Enterprise Designer と異なる表現が使用される場合があります。動作の変更はありません。

表 48 : データ オプション

optionName	説明
DatabaseSearchOrder	検索プロセスで使う 1 つ以上のデータベース リソースの名前。Management Console の Spectrum のデータベース ページで指定したデータベース名を使用します。ツール。詳細については、『*Spectrum™ Technology Platform管理ガイド*』を参照してください。 複数のデータベース リソースを指定できます。複数のデータベースを指定する場合は、優先度の高いデータベースからリストに追加します。データベースの順序は、複数のデータベースに近似一致が見つかったときに意味を持ちます。返される近似一致は、検索リストの先頭にあるデータベースから取得されます。それより下位のデータベースに見つかった近似一致は、非近似一致に格下げされます。

出力

Geocode Address World は、緯度/経度、都市、郡、および結果インジケータを返します。結果インジケータは、入力がどの程度まで既知の場所や割り当てられた緯度/経度に一致したかを表し、マッチング試行全体のステータスも示します。情報は大文字を使用して返されます。

API を使用する場合は、出力は`DataTable`クラスで返されます。詳細については、『Spectrum™ Technology PlatformAPI ガイド』を参照してください。

住所の出力

表 49 : 住所の出力

columnName	説明
City	地方自治体名。
CityRank	CityRank は、総合的および相対的な人口、重要度、その他の基準に基づいて決まる 1 (最高) から 10 (最低) までの数値です。
Country	3 文字の ISO 3166-1 Alpha 3 国コード。2 文字のコードも使用できます。地理的ジオコーディングの国とデータ ソースのリストについては、**各国の地理的データの対象範囲**（290ページ）を参照してください。郵便番号ジオコーディングの国とデータ ソースのリストについては、**各国の郵便データの対象範囲**（308ページ）を参照してください。

columnName	説明
County	このフィールドには、州/省より小さく都市より大きいエリアが含まれます。特定のエリアは国によって異なります。 • AUS — 地方自治体 (Local Government Authority: LGA) • AUT — 州 (Province) • BEL — 州 (Province) • BHS — 使用せず • BRA — 使用せず • CAN — 使用せず • CHE — 州 (Province) • DEU — 郡 (Kreis) • DNK — 州 (Province) • FIN — 州 (Province) (kommune) • FRA — 郡 (Department) • GBR — 郡 (County) • ITA — 州 (Province) • LIE — 州 (Province) • LUX — 州 (Province) • MYS — 地区 (District) (daerah) • NLD — 州 (Province) • NZL — 使用せず • POL — 地区 (District) (poviat) • PRT — 使用せず • SGP — 地区 (District) • SWE — 地域 (Region) (kommun) • THA — 地区 (District) (amphoe)
PostalCode	住所の郵便番号。郵便番号のフォーマットは国によって異なります。

columnName	説明
StateProvince	StateProvince の意味は国によって異なります。 • AUS — 州 (State) • AUT — 地域 (Region) • BEL — 使用せず • BRA — 州 (State) • CAD — 州 (Province) • CHE — 州 (State) • DEU — 連邦州 (Bundesland) • DNK — 使用せず • ESP — 地域 (Region) • FIN — 地域 (Region) (län) • FRA — 地域 (Region) • GBR — 地域 (Region) • IRL — 使用せず • ITA — 地域 (Region) • LIE — 州 (State) • LUX — 使用せず • MYS — 州 (State) (negeri) • NLD — 使用せず • NOR — 使用せず • NZL — 地域 (Region) • POL — 県 (Province) (voivodship) • PRT — 地域 (Region) • SGP — 使用せず • SWE — 地域 (Region) (lan) • THA — 県 (Province) (changwat)

ジオコード出力

表 50 : ジオコード出力

columnName	説明
CoordinateSystem	緯度/経度座標を決定するために使われる座標系。座標系は地図投影法、座標単位などを指定します。例は EPSG:4326 です。EPSG は European Petroleum Survey Group の略語です。

columnName	説明
Latitude	小数点以下 4 桁までが計算される 7 桁の度数 (指定したフォーマットで表記されます)。
Longitude	小数点以下 4 桁までが計算される 7 桁の度数 (指定したフォーマットで表記されます)。

結果コード

結果コードは、ジオコーディングの成功または失敗に関する情報やジオコードの精度に関する情報を示します。

表 51 : 結果コード出力

columnName	説明
Geocoder.MatchCode	入力住所が候補住所にどの程度近いかを示します。
IsCloseMatch	住所が近似一致と見なされるかどうかを示します。住所は、[マッチング] タブの [近似検索条件] オプションで設定した基準に基づいて近似かどうかが決定されます。 **Y** 住所は近似一致です。 **N** 住所は近似一致ではありません。
MultiMatchCount	ストリート住所のジオコーディングの場合は、指定された住所に見つかったマッチングする住所の数。 交差点のジオコーディングの場合は、指定された住所に見つかったマッチングする交差点の数。
Status	マッチの成功または失敗を報告します。 **NULL** 成功 **F** 失敗

columnName	説明
Status.Code	ジオコーダが住所を処理できない場合、このフィールドにその理由が設定されます。 • Internal System Error • No Geocode Found • Insufficient Input Data • Multiple Matches Found • Exception occurred • Unable to initialize Geocoder • No Match Found
Status.Description	ジオコーダが住所を処理できない場合、このフィールドに失敗に関する説明が設定されます。 **Problem + explanation** Status.Code = Internal System Error の場合にこれが返されます。 **Geocoding Failed** Status.code = No Geocode Found の場合にこれが返されます。 **No location returned** Status.code = No Geocode Found の場合にこれが返されます。 **No Candidates Returned** ジオコーダは住所に一致する候補を識別できませんでした。 **Multiple Candidates Returned and Keep Multiple Matches not selected** 住所に一致する候補が複数見つかりました。候補の住所が返されるためには、`KeepMultimatch=Y` を指定する必要があります。

columnName	説明	
LocationPrecision	ジオコードの精度を表すコード。次のいずれかです。	
	0	この候補住所の座標情報はありません。
	1	補間されたストリート住所。
	2	ストリートセグメントの中間点。
	3	郵便番号 1 セントロイド。
	4	部分郵便番号 2 セントロイド。
	5	郵便番号 2 セントロイド。
	6	交差点。
	7	POI (point-of-interest)。プレースホルダ値です。Spectrum のデータベースには POI データがないので、この値を返すことはできません。
	8	州/省セントロイド。
	9	郡セントロイド。
	10	都市セントロイド。
	11	地方セントロイド。
	12 ～ 15 (LocationPrecision コード)	ほとんどの国では、LocationPrecision コード 12 ～ 15 が未指定のカスタム項目用に予約されています。
	13	未指定のカスタム項目に使う追加のポイント精度。
	14	未指定のカスタム項目に使う追加のポイント精度。
	15	未指定のカスタム項目に使う追加のポイント精度。
	16	結果は住所ポイント。
	17	住所ポイント データを使って候補セグメント データを修正し、結果を生成しました。
	18	結果は、中央線オフセット機能を使用して投影された住所ポイント。中央線オフセット機能を使用し、それによって LocationPrecision 18 を返すには、ポイントとストリート範囲の両方のデータベースを使用する必要があります。

columnName	説明
StreetDataType	住所のジオコーディングに使うデータベースのデフォルトの検索順序ランク。値 "1" はそのデータベースがデフォルト検索順序の先頭のデータベースであり、値 "2" はデフォルト検索順序の 2 番目のデータベースであることを意味します。以降も同様です。 デフォルトのデータベース検索順序は、Management Console で指定します。

地理的候補のランク付け

多くの国において、名前が同一の地理的エリアが見つかる場合があります。このような場合、World はランク付けシステムを使用して、最も可能性の高い近似一致候補を決定します。

この重みランク付けは、データ ソース (TomTom、GeoNames、または Pitney Bowes のソース) によって具体的な詳細部分は一部異なりますが、近似一致である可能性が最も高い候補の決定には、以下の基準が重みとして適用されます。

- 国の首都
- 行政区分 (州/省、地域、郡) の首都
- 人口規模

国の首都である場合は、他のどの地理的ランク付け基準よりも優先されます。例えば、都市として San Juan が入力された場合、プエルトリコ (PRI) の San Juan (サンファン) が近似一致として返されます。プエルトリコの首都であるためです。他の国の都市である San Juan (スペイン、コスタリカ、ドミニカ共和国、フィリピンなど) は、その人口に関係なく、非近似一致として返されます。複数のマッチを返すには、Management Console のマッチング オプションにおける [複数の一致を保持] チェック ボックスをオンにして、返すマッチ数を指定する必要があります。

同様に、World は、Roma, ITA (イタリアのローマ) を近似一致として返します。これがイタリアの首都であるためです。ルーマニア、ホンジュラス、パナマの Roma は、非近似一致として返されます。

州/省の行政上の首都である場合は、人口があまり多くない場合でも高く加重されます。例えば、Springfield に対しては、Springfield, Illinois USA (米国イリノイ州スプリングフィールド) が地理的な近似一致として返されます。イリノイ州の州都であるためです。Springfield, Massachusetts (マサチューセッツ州スプリングフィールド) の方が人口はやや多いですが、イリノイ州のスプリングフィールドは州都であることから、こちらが優先されます。米国やその他の国の人口がそれよりも少ない Springfield という名前の地域も近似一致として返されますが、イリノイ州スプリングフィールドよりも下に表示されます。大都市が、名前が同じでそれよりも小さい州/省の首都と、同等の近似一致としてランク付けされることはあり得ます。しかし、州/省の首都は、人口が比較的少ない場合でも、そのランクを引き下げられることはありません。

同様に、都市として Albany が入力され、国が指定されなかった場合には、World は、Albany, NY, USA (米国ニューヨーク州アルバニー) を近似一致候補として返します。これは、アルバニーがニューヨーク州の州都であることから、行政区域の首都として高く加重されるためです。人口も、ランク付けにおいて考慮される要素です。New Zealand (ニュージーランド) など、別の国とともに Albany という都市を指定すると、その国が使用され、近似一致候補として Albany, NZL (ニュージーランドのアルバニー) が返されます。

候補に都市が含まれていて CityRank 値が利用できる場合は、その値も返されます。CityRank は、都市の相対的な重要度を表す 1 から 10 までの数値です。重要度は 1 が最高で、10 が最低です。このランク付けは、相対的な人口、行政上の位置づけなどの基準に基づいています。複数の地理的ジオコーディングが返されるときは、都市ランクの順序でソートされます。

マッチ コード

G カテゴリでの一致は、その候補が次のいずれかの精度レベルで地理的セントロイドに存在することを示します。すべての精度レベルがすべての国で使用できるわけではありません。

- **G0** — 国セントロイド。GeocodeAddressWorld ではこの精度は返されません。
- **G1** — 州/省セントロイド。日本では、これは都道府県の一致を示します。
- **G2** — 郡セントロイド。日本では、これは市の一致を示します。
- **G3** — 都市セントロイド。日本では、地方自治体の下位区分である subcity (大字) の一致を示します。オーストラリアでは、Local Government Authority (LGA) 情報は G-NAF データベースを使わずに、ストリート範囲住所データベースのみで生成できます。
- **G4** — 地方セントロイド。日本では、これは市内の地区 (丁目) の一致を示します。

Z カテゴリの一致は、ストリートのマッチングが以下のいずれかの理由で成立しなかったことを意味します。

- 郵便番号セントロイドへのマッチングを指定した。結果のポイントは、4 段階の精度で郵便番号セントロイドに位置付けられます。
- 近似一致が見つからなかった。なおかつ、郵便番号セントロイドへの代替を指定した。

Z カテゴリには、次の 4 段階の精度があります。

- **Z0** — 使用できる座標がない郵便番号一致 (めったにないケース)。
- **Z1** — 郵便番号セントロイド一致。
- **Z3** — 完全な郵便番号セントロイド一致。カナダでは、これは FSALDU セントロイドです。
- **Z6** — ポイント ZIP の郵便番号セントロイド一致。

GNAFPIDLocationSearch

GNAFPIDLocationSearch は、Geocoded National Address File Persistent Identifier (G-NAF PID) の住所および緯度/経度座標を特定します。G-NAF PID は、G-NAF データベース (オーストラリア

の住所のデータベース) 内の G-NAF の住所を一意に定義する 14 文字の英数字からなる文字列です。PID は G-NAF データベースの主要な住所フィールドの組み合わせから構成されています。G-NAF PID は、以下のような形式です。

GAVIC411711441

> **注：** GNAFPIDLocationSearch を使用するには、G-NAF データベースがインストールされている必要があります。

GNAFPIDLocationSearch は、Geocoding Address AUS コンポーネントの一部です。Geocode Address AUS から使用されるステージは、GNAF PID Location Search のみです。このコンポーネントはそれ以外では非推奨になっています。その他のすべてのオーストラリアのジオコーディング機能には、Geocode Address Global コンポーネントを使用してください。

Enterprise Geocoding モジュールの詳細については、**Enterprise Geocoding モジュール**（270ページ）を参照してください。

G-NAF PID 入力

GNAFPIDLocationSearch は、入力として G-NAF PID を受け取ります。そして、Geocoded National Address File Persistent Identifier (G-NAF PID) の住所および緯度/経度座標を返します。

> **注：** GNAF PID Location Search 機能は、Geocode Address Global コンポーネントによってサポートされていません。この機能には、Geocode Address AUS コンポーネントを使用する必要があります。Geocode Address AUS から使用されるステージは、GNAF PID Location Search のみです。このコンポーネントはそれ以外では非推奨になっています。

表 52 : GNAFPIDLocationSearch の入力

columnName	書式	説明
GNAFPID	String	検索する 14 文字の G-NAF Persistent Identifier。例: GAVIC411711441

> **注：** `DataTable`クラスを使用して入力を指定します。詳細については、『Spectrum™ Technology PlatformAPI ガイド』を参照してください。

G-NAF PID Location Search のオプション

GNAFPIDLocationSearch には、PID 検索用の G-NAF データベースを選択するオプションがあります。

G-NAF ジオコーディング オプション

表 53 : GNAFPIDLocationSearch のジオコーディング オプション

オプション名	説明
GNAFPointType	小区画の緯度/経度を返すか、ストリートの入口の緯度/経度を返すかを指定します。このオプションは、G-NAF データベースがインストールされている場合にのみ使用できます。このオプションは、G-NAF データベースのデータに一致する住所のみに適用されます。 次のいずれかです。 **P** ストリート住所マッチングで小区画の正確な場所を返します。これは標準 G-NAF ポイントであり、G-NAF データベースから返される正規のポイントです。こちらがデフォルトです。 **S** ストリート住所マッチングで小区画のストリートの入口ポイントを返します。ストリートの入口ポイントは、小区画の入口の境界から 12.5 m 離れています。ストリートの入口ポイントは、ルーティング アプリケーションでの使用に適しています。
Return8DecimalPlaceLatLong	元の緯度/経度を、小数点以下 8 桁までの精度で返すかどうかを指定します。これは、G-NAF データベースのデータに一致する候補の緯度/経度です。これらは G-NAF データから直接取得された元の座標であり、切り捨てや四捨五入は行われていません。このオプションは、G-NAF データベースがインストールされている場合にのみ使用できます。このオプションは、G-NAF データベースのデータに一致する住所のみに適用されます。 **Y** 元の緯度/経度を、小数点以下 8 桁までの精度で返します。 **N** 元の緯度/経度を、小数点以下 8 桁までの精度で返しません。

G-NAF PID のデータ オプション

表 54 : GNAFPIDLocationSearch のデータ オプション

オプション名	説明
Database	小区画の検索に使用するデータベースを指定します。Management Console で指定したデータベース名を使用します。詳細については、『*Spectrum™ Technology Platform 管理ガイド*』を参照してください。 **注 :** この一覧には、G-NAF データベースを含むデータベース リソースのみが表示されます。

出力

住所の出力

表 55 : 住所の出力

columnName	説明
AddressLine1	住所の最初の行。
AddressLine2	住所の 2 番目の行。
ApartmentLabel	ユニット タイプ。アパート、スイート、号など。
ApartmentNumber	ユニット番号。
City	地方自治体名。
Country	3 文字の ISO 3166-1 Alpha 3 国コード。

columnName	説明
County	地方自治体 (LGA) の名前。
FirmName	会社名または場所の名前。
HouseNumber	一致したロケーションの建物番号。
HouseNumberHigh	住所がある範囲の最も大きな家番号。
HouseNumberLow	住所がある範囲の最も小さな家番号。
HouseNumberParity	家番号の範囲に奇数または偶数、またはその両方の番号が含まれるかどうかを示します。 **E** 偶数 **O** 奇数 **B** 両方
LastLine	完成された最終の住所行 (都市、州/省、および郵便番号)。
LeadingDirectional	ストリート名の前に付けてストリートの方向を表します。例えば、138 N Main Street の N がこれに該当します。
Locality	一般的には、農村部の地方または都市部の郊外です。
NumberOfCandidateRanges	住所に家番号があるかどうかを示します。次のいずれかです。 **0** 住所に家番号はありません。家番号がない住所の例として、私書箱の住所および局留めの住所があります。 **1** 住所に家番号があります。家番号が含まれる範囲については、[HouseNumberHigh]、[HouseNumberLow]、および [HouseNumberParity] の各フィールドから確認できます。

columnName	説明
NumberOfRangeUnits	住所にスイート番号やアパート番号などのユニット番号が含まれるかどうかを示します。次のいずれかです。 **0** 住所にユニット番号が含まれません。 **1** 住所にユニット番号が含まれます。ユニット番号が含まれる範囲については、[UnitNumberHigh] フィールドと [UnitNumberLow] フィールドから確認できます。
PostalCode	住所の郵便番号。郵便番号のフォーマットは国によって異なります。
PostalCode.Addon	郵便番号の 2 番目の部分。例えば、カナダの住所ではこれは LDU です。ほとんどの国ではこのフィールドを使用しません。
PreAddress	ストリート名の前に記述されるその他の情報。
PrivateMailbox	現在、このフィールドは使用されていません。
SegmentParity	ストリートのどちら側に奇数番号が振られているかを示します。 **L** ストリートの左側 **R** ストリートの右側 **B** ストリートの両側 **U** 未確認
StateProvince	州の名前。
StreetDataType	住所のジオコーディングに使うデータベースのデフォルトの検索順序ランク。値 "1" はそのデータベースがデフォルト検索順序の先頭のデータベースであり、値 "2" はデフォルト検索順序の 2 番目のデータベースであることを意味します。以降も同様です。 デフォルト検索順序は、Management Console の Spectrum のデータベース ページで指定します。

columnName	説明
StreetName	ストリート名。
StreetPrefix	基本のストリート名の前にストリートタイプを明記する場合に、そのストリートタイプ。例えば、以下の場合の AVENUE。 12 **AVENUE** B KALGOORLIE WA 6430
StreetSuffix	一致した場所のストリートタイプ。例えば、Avenue の AVE など。
TrailingDirectional	ストリート名の後に記述するストリートの方位記号。例えば、456 Washington N の N。
UnitNumberHigh	ユニットが含まれる範囲における最も大きなユニット番号。
UnitNumberLow	ユニットが含まれる範囲における最も小さなユニット番号。

ジオコード出力

表 56 : ジオコード出力

フィールド名	説明
CoordinateSystem	緯度/経度座標を決定するために使われる座標系。地図投影法、座標単位などを指定する座標系 (例えば、EPSG:4326)。EPSG は European Petroleum Survey Group の略語です。
Latitude	小数点以下 4 桁までが計算される 7 桁の度数 (指定したフォーマットで表記されます)。
Longitude	小数点以下 4 桁までが計算される 7 桁の度数 (指定したフォーマットで表記されます)。

結果コード

結果コードは、ジオコーディングの成功または失敗に関する情報やジオコードの精度に関する情報を示します。

> **注：** EGM モジュールにより管理タスクが Web ベースの Management Console に移行されると、オプションのレベルで、Enterprise Designer と異なる表現が使用される場合があります。動作の変更はありません。

表 57：結果コード出力

columnName	説明
Geocoder.MatchCode	入力住所が候補住所にどの程度近いかを示します。
IsCloseMatch	住所が近似一致と見なされるかどうかを示します。住所は、[マッチング] タブの [近似検索条件] オプションで設定した基準に基づいて近似かどうかが決定されます。 **Y** 住所は近似一致です。 **N** 住所は近似一致ではありません。
MultiMatchCount	ストリート住所のジオコーディングの場合は、指定された住所に見つかったマッチングする住所の数。 交差点のジオコーディングの場合は、指定された住所に見つかったマッチングする交差点の数。
Status	マッチの成功または失敗を報告します。 **NULL** 成功 **F** 失敗

columnName	説明
Status.Code	ジオコーダが住所を処理できない場合、このフィールドにその理由が設定されます。 • Internal System Error • No Geocode Found • Insufficient Input Data • Multiple Matches Found • Exception occurred • Unable to initialize Geocoder • No Match Found
Status.Description	ジオコーダが住所を処理できない場合、このフィールドに失敗に関する説明が設定されます。 **Problem + explanation** Status.Code = Internal System Error の場合にこれが返されます。 **Geocoding Failed** Status.code = No Geocode Found の場合にこれが返されます。 **No location returned** Status.code = No Geocode Found の場合にこれが返されます。 **No Candidates Returned** ジオコーダは住所に一致する候補を識別できませんでした。 **Multiple Candidates Returned and Keep Multiple Matches not selected** 住所に一致する候補が複数見つかりました。候補の住所が返されるためには、`KeepMultimatch=Y` を指定する必要があります。

columnName	説明	
LocationPrecision	ジオコードの精度を表すコード。次のいずれかです。	
	0	この候補住所の座標情報はありません。
	1	補間されたストリート住所。
	2	ストリートセグメントの中間点。
	3	郵便番号 1 セントロイド。
	4	部分郵便番号 2 セントロイド。
	5	郵便番号 2 セントロイド。
	6	交差点。
	7	POI (point-of-interest)。プレースホルダ値です。Spectrum のデータベースには POI データがないので、この値を返すことはできません。
	8	州/省セントロイド。
	9	郡セントロイド。
	10	都市セントロイド。
	11	地方セントロイド。
	12 ～ 15 (LocationPrecision コード)	ほとんどの国では、LocationPrecision コード 12 ～ 15 が未指定のカスタム項目用に予約されています。
	13	未指定のカスタム項目に使う追加のポイント精度。
	14	未指定のカスタム項目に使う追加のポイント精度。
	15	未指定のカスタム項目に使う追加のポイント精度。
	16	結果は住所ポイント。
	17	住所ポイント データを使って候補セグメント データを修正し、結果を生成しました。
	18	結果は、中央線オフセット機能を使用して投影された住所ポイント。中央線オフセット機能を使用し、それによって LocationPrecision 18 を返すには、ポイントとストリート範囲の両方のデータベースを使用する必要があります。

columnName	説明
StreetDataType	住所のジオコーディングに使うデータベースのデフォルトの検索順序ランク。値 "1" はそのデータベースがデフォルト検索順序の先頭のデータベースであり、値 "2" はデフォルト検索順序の 2 番目のデータベースであることを意味します。以降も同様です。 デフォルトのデータベース検索順序は、Management Console で指定します。

G-NAF 出力

以下の表に、Australian Geocoded National Address File (G-NAF®) データベース固有の出力フィールドを示します。G-NAF は、全 6 州と 2 つの特別地域に対応したオプションのデータベースです。G-NAF は、オーストラリア全土の地方、ストリート、および番号を表す唯一の公式インデックスであり、検証済みの地理的座標も含まれます。

表 58 : Australia G-NAF 出力

columnName	説明
AUS.GNAF_ADDRESS_CLASS	Address_Class は、G-NAF Data Dictionary ソース テーブルの要素を組み合わせて作成されます。Address_Class フィールドの構成要素は次のとおりです。 **A** エイリアス住所レコード **P** 主要住所レコード **PP** 主要プライマリ住所レコード **PS** 主要セカンダリ住所レコード **AP** エイリアス プライマリ住所レコード **AS** エイリアス セカンダリ住所レコード

columnName	説明
AUS.GNAF_CONFIDENCE	住所が含まれる G-NAF データセットの数を示します。同じ住所が多くのデータ供給ソースで見つかるほど、確信レベルは高くなります。次のいずれかです。 **<数値>** 住所が含まれることが確認されたデータセットの総数から 1 を引いた値です。例えば、値が 0 のときは、住所が 1 つの供給元のデータセットに見つかったことを意味し、値 1 は 2 つの供給元のデータセットに見つかったことを意味します。値 2 は 3 つの供給元のデータセットに見つかったことを意味し、以降もこれと同様です。 **-1** 住所はどの G-NAF データセットにも見つかりませんでした。
AUS.GNAF_EIGHT_DECIMAL_PLACE_LATITUDE	小区画の緯度。小数点以下 8 桁までの精度で表されます。これは、G-NAF データベースのデータに一致した候補の緯度です。これらは G-NAF データから直接取得された元の座標であり、切り捨てや四捨五入は行われていません。 このフィールドは、**Return8DecimalPlaceLatLong=Y** を指定した場合にのみ返されます。
AUS.GNAF_EIGHT_DECIMAL_PLACE_LONGITUDE	小区画の経度。小数点以下 8 桁までの精度で表されます。これは、G-NAF データベースのデータに一致した候補の経度です。これらは G-NAF データから直接取得された元の座標であり、切り捨てや四捨五入は行われていません。 このフィールドは、**Return8DecimalPlaceLatLong=Y** を指定した場合にのみ返されます。

columnName	説明
AUS.GNAF_GEOCODE_LEVEL	住所のジオコードのレベルを示す番号。G-NAF データベース内の主要エリアの住所には最低1つの地方レベルのジオコードがあります。さらに、ストリートレベルやポイントレベルのジオコードがある場合もあります。 次のいずれかです。 **0** ジオコードはありません。 **1** 小区画レベルのジオコードのみ (地方レベルまたはストリートレベルのジオコードはありません)。 **2** ストリートレベルのジオコードのみ (地方レベルまたは小区画レベルのジオコードはありません)。 **3** ストリートレベルと小区画レベルのジオコードのみ (地方レベルのジオコードはありません)。 **4** 地方レベルのジオコードのみ (ストリートレベルまたは小区画レベルのジオコードはありません)。 **5** 地方レベルと小区画レベルのジオコード (ストリートレベルのジオコードはありません)。 **6** 地方レベルとストリートレベルのジオコード (小区画レベルのジオコードはありません)。 **7** 地方レベル、ストリートレベル、および小区画レベルのジオコード。
AUS_GNAF_PARCEL_ID	Parcel ID フィールドは、管理データによって提供される汎用小区画 ID フィールドで、政府機関にとって便利な地番の説明を表します。正確なフォーマットはさまざまです。G-NAF ソース データには、Parcel_ID を持つレコードが 700 万以上あります。オーストラリアのジオコーダは、これを補足して、12,730,000 を越える G-NAF レコードに Parcel_ID フィールドを設定します。
AUS.GNAF_PID	G-NAF Persistent Identifier (G-NAF PID) は、G-NAF の住所を一意に定義する 14 文字の英数字からなる文字列です。PID は G-NAF データベースの主要な住所フィールドの組み合わせから構成されています。G-NAF PID は、以下のような形式です。 GAVIC411711441

columnName	説明
AUS.GNAF_RELIABILITY	ジオコードの精度を示す番号。信頼度は、ジオコードを決定するために使われる辞書の品質に左右されます。ジオコード信頼レベル 1、2、および 3 のデータは、GNAF123 Dictionary に格納されています。これはポイント (小区画) レベルのジオコード データです。ジオコード信頼レベル 4、5、および 6 のデータは、GNAF456 Dictionary に格納されています。この辞書には、小区画以外のセントロイド ジオコード データが格納されています。 **1** ジオコードの精度は、適切な測定基準を満たしました。例えば、手動で実行された住所レベルのジオコードは、このレベルになります。ジオコードの解像度は、GPS を使ってセントロイドを住所サイト境界内に配置できるレベルです。 **2** ジオコードの精度は、セントロイドを住所サイト境界内に配置できるレベルです。例えば、対応する地籍上の小区画のセントロイドとして自動的に計算された住所レベルのジオコードは、このレベルになります。 **3** ジオコードの精度は、セントロイドを住所サイト境界の近く (場合によっては内部) に配置できるレベルです。例えば、他の境界ジオコード住所に基づいて対象の住所が位置すると思われる道路を計算するという方法で自動的に計算された住所レベルのジオコードは、このレベルになります。 **4** ジオコードの精度は、住所サイトを道路の固有の特性に関連付けることができるレベルです。例えば、道路の中央線リファレンス データを使って自動的に計算されたストリートレベルのジオコードは、このレベルになります。 **5** ジオコードの精度は、住所サイトを特定の地方または地区に関連付けることができるレベルです。例えば、地方のセントロイドとして自動的に計算された地方レベルのジオコードは、このレベルになります。 **6** ジオコードの精度は、住所サイトを特定の地域に関連付けることができるレベルです。例えば、地形特性から導き出された地方レベルのジオコードは、このレベルになります。

columnName	説明
AUS.GNAF_SA1	Statistical Area Level 1 (SA1) フィールドは、Australian Statistical Geography Standard (ASGS) に定義されている 2 番目に小さい地理的地域です 。Mesh Block が最小単位です。SA1 は、Census データの処理およびリリースの最小単位としてCensus of Population and Housing で使用するためのものです。SA1 は、一意の 7 桁のコードで表されます。
AUS.LEVEL_NUMBER	多層建築の階またはレベルの番号。例を次に示します。 **Floor 2**, 17 Jones Street G-NAF データベースには、オーストラリアの一部の州のレベル情報が含まれています。レベル情報がユニット情報に関連付けられている場合もありますが、常にこの関連付けがあるわけではありません。G-NAF データベースでは、複数のレコードが同じレベルに含まれます。入力住所に個有のコンテンツ（ユニット番号など）がある場合にのみ、レベル情報が返されます。G-NAF データベースに住所のレベル情報がある場合、ジオコーダは一致した候補と共にその情報を返します。 入力住所にレベル情報がない場合、あるいは入力のレベル情報が不正確な場合でも、正しいレベル情報が返されます。入力住所にレベル情報があるが、G-NAF データベースにはマッチング住所にレベル情報がない場合、G-NAF データによって検証されない情報であるとして入力レベル情報が破棄されます。

columnName	説明
AUS.LEVEL_TYPE	多層建築の階に使われるラベル。例えば、"Level" または "Floor"。この例で、レベル タイプは "Level" です。 Suite 3 **Level 7**, 17 Jones Street この例で、Suite 3 はユニットです。 G-NAF データベースには、オーストラリアの一部の州のレベル情報が含まれています。レベル情報がユニット情報に関連付けられている場合もありますが、常にこの関連付けがあるわけではありません。G-NAF データベースでは、複数のレコードが同じレベルに含まれます。入力住所に個有のコンテンツ（ユニット番号など）がある場合にのみ、レベル情報が返されます。G-NAF データベースに住所のレベル情報がある場合、ジオコーダは一致した候補と共にその情報を返します。 入力住所にレベル情報がない場合、あるいは入力のレベル情報が不正確な場合でも、正しいレベル情報が返されます。入力住所にレベル情報があるが、G-NAF データベースにはマッチング住所にレベル情報がない場合、G-NAF データによって検証されない情報であるとして入力レベル情報が破棄されます。
AUS.MESH_BLOCK_ID	Meshblock は、オーストラリア統計局 (ABS) が統計データを収集するために作成した最も小さい地理的な単位です。通常、Meshblock には最低 20 から 50 の世帯が含まれています。これは、収集区 (CD: Collection District) の約 5 分の 1 の大きさです。Meshblock ID を使って、独自のデータに属性を追加できます。
AUS.LOT_NUMBER	適切な物理的または家番号情報を持たない地方住所があるため、敷地番号が G-NAF 候補として返されます。
AUS.STREET_TYPE_ABB	ストリートタイプの略語です。例えば、EX は Extension の略語で、FTRL は Firetrail の略語です。

Reverse Geocode Address Global

Reverse Geocode Address Global へのアクセスに API を使用する方法については、ジオコーディング ガイドを参照してください。

国際ジオコーディングの結果コード

Spectrum のジオコーダによって返される候補は、国際ジオコーディング結果コードと呼ばれる別のクラスのリターン コードを返します。マッチング試行ごとに結果コードが Geocoder.MatchCode 出力フィールドに返されます。

国際ストリートジオコーディングの結果コード (S コード)

ストリートレベルでジオコーディングされた候補は、文字 S で始まる結果コードを返します。コードの 2 番目の位置は、ジオコーディングされたレコードの結果ポイントの位置的な精度を示します。

表 59 : ストリート (S) 結果コード

S 結果コード	説明
S1	郵便番号セントロイドにポイントが位置付けられた単一近似一致。
S3	郵便番号セントロイドにポイントが位置付けられた単一近似一致。
S4	ストリートセントロイドにポイントが位置付けられた単一近似一致。データベース ヴィンテージ 2014 Q4 以降では、入力家番号が見つからなかった場合でも、その家番号が候補とともに返されます。S4 コードの後に、マッチングの精度を示す文字とダッシュが設定されます。**結果コード S の意味**（354ページ）を参照してください。
S5	ストリート住所の位置にポイントが位置付けられた単一近似一致。S5 コードの後にはマッチ精度を表す文字とダッシュが続きます。これらの文字の詳細については、**結果コード S の意味**（354ページ）を参照してください。

S 結果コード	説明
S7	候補のストリートセグメント沿いの補間ポイントに位置付けられた単一一致。潜在的な候補が住所ポイント候補ではなく、他の住所ポイント候補には家番号が正確に一致するものがない場合、S7 の結果コードが住所ポイント補間を使って返されます。このポイント補間は、セグメントが交差し、家番号が元の候補の家範囲に含まれる 2 番目に高いか低い住所ポイント候補に従って行われます。ストリートセグメント上の既知の住所リファレンス ポイントを使って、S7 ポイントをより正確な位置に調整できます。
S8	住所ポイント候補に関連付けられた単一ポイント、または家番号が同一の住所ポイント候補にポイントが位置付けられた単一近似一致。補間は必要ありません。S8 を返すことが可能なのはポイント データベースを使用する場合のみです。
SX	交差点にポイントが位置付けられた単一近似一致。

結果コード S の意味

国際結果コード S (ストリートジオコーディング) では、追加の 8 文字により、住所がデータベース内の住所にどの程度一致するかが示されます。これらの文字は、以下の表に示す順序で並びます。一致しない住所要素はダッシュで表わされます。

例えば、S5--N-SCZA という結果コードは、ストリート名、後置方位記号、都市名、および郵便番号が一致する単一近似一致を意味します。ダッシュは、家番号、前置方位記号、および大ストリートタイプにマッチングがないことを示します。一致する候補は、ストリート範囲住所データベースに見つかりました。このレコードは、見つかった候補のストリート住所の位置にジオコーディングされます。

Category	説明	例
H	家番号	18
P	ストリートの前置方位記号 P は、次の条件が 1 つでも満たされた場合に示されます。 • 候補の前置方位記号が、入力の前置方位記号と一致する。 • 前置方位記号と後置方位記号を入れ替えると、候補の後置方位記号と入力の前置方位記号が一致する。 • 入力に前置方位記号が含まれない。	North

Category	説明	例
N	ストリート名	Merivale
T	ストリートタイプ	St
S	ストリートの後置方位記号 結果コードの S は、次の条件が 1 つでも満たされた場合に示されます。 • 候補の後置方位記号が、入力の後置方位記号に一致する。 • 前置方位記号と後置方位記号を入れ替えると、候補の前置方位記号と入力の後置方位記号が一致する。 • 入力に後置方位記号が含まれない。	w
C	都市名	South Brisbane
Z	郵便番号	4101
A、G、または U	一致する候補の取得に使われるデータベースのタイプ。 • A — ストリート範囲住所データベース。 • U — 顧客 (ユーザ定義) データベース。	A

国際郵便番号ジオコーディングの結果コード (Z コード)

Z カテゴリの一致は、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。郵便番号一致が返されるのは、次のどちらかの場合です。

- 郵便番号セントロイドへのマッチングを指定した。結果のポイントは、以下の精度レベルをとり得る郵便番号セントロイドに位置付けられます。
- ストリートレベルの近似一致が見つからなかった。なおかつ、郵便番号セントロイドへの代替を指定した。

表 60 : 郵便 (Z) 結果コード

Z 結果コード	説明
Z1	郵便番号セントロイド一致。
Z3	完全な郵便番号セントロイド一致。カナダでは、これは FSALDU セントロイドです。

郵便番号レベルでジオコーディングされた候補は、Z という文字で始まる結果コードを返します。World は、Z1 結果コードを生成できます。国固有のジオコーダは、より正確な郵便番号ジオコーディング結果 (結果コード Z2 または Z3) を生成できることがあります。

郵便番号候補がユーザ辞書から得られた場合は、結果に U の文字が付加されます。例えば、Z1U は、カスタム ユーザ辞書から得られた郵便番号セントロイド一致を示します。

国際地理的ジオコーディングの結果コード (G コード)

地理的レベルでジオコーディングされた候補は、G という文字で始まる結果コードを返します。G の後に続く結果コード内の数値は、その候補の精度に関するより詳細な情報を提供します。

表 61 : 地理的 (G) 結果コード

G 結果コード	説明
G1	州または省セントロイドの一致に基づいて、地理的な近似一致候補を返します。
G2	郡 (地区または地域) セントロイド一致です。
G3	都市または町 (地方自治体) セントロイド一致です。
G4	地方 (村、郊外、または地区) セントロイド一致です。

地理的候補がユーザ辞書から得られた場合は、結果コードに U の文字が付加されます。例えば、G4U は、カスタム ユーザ辞書から得られた地方セントロイド一致を示します。

リバース ジオコーディング コード (R コード)

R カテゴリの一致は、レコードがリバース (逆順序) のジオコーディングで一致したことを意味します。R 結果コードの 2 番目の文字は、見つかったマッチングのタイプを示します。R のジオコード結果には、マッチングが見つかった辞書を示す追加の文字が含まれます。

リバース ジオコーディング コードの例を以下に示します。

表 62 : リバース ジオコーディング (R) の結果コード

リバース ジオコーディング コード	説明
RS8A	リバース ジオコーディングのポイント/小区画レベルの精度。住所辞書から返された候補です。
RS5A	リバース ジオコーディングの補間後のストリート候補。住所辞書から返された候補です。
RS4A	リバース ジオコーディングのストリートセントロイド候補。住所辞書から返された候補です。

リバース ジオコーディングされた候補が、ユーザ辞書から得られた場合は、結果に U の文字が付加されます。例えば、RS8U は、カスタム ユーザ辞書から得られたポイント/小区画レベルのリバース ジオコード一致を示します。

一致なしコード

次の結果コードは、マッチングがなかったことを示します。

- **N** — 近似一致はありません。
- **NX** — 交差点の近似一致はありません。
- **ND** — Spectrum™ Technology Platform は、入力された郵便番号または地方自治体/州/省のジオコーディング データベースを見つけられませんでした。

GeoConfidence モジュール

GeoConfidence モジュール

GeoConfidence モジュールは、指定された領域に住所または交差点が含まれる可能性を判定するために使われます。このモジュールは住所または交差点の場所 (Geocode US Address で確認された場所) を受け取り、それをポイント、ライン、またはポリゴン (どれになるかはマッチング精度による) に変換した後で、その形状を既知の形状のデータベースと照合して、重なり合う形状があるかどうかを調べ、あった場合は重なる部分の割合を確認します。例えば、GeoConfidence モジュールを使って洪水危険地域の格付けを判定できます。この場合、住所の場所と洪水発生地域データの重なり合う面積が判定基準となります。過去 100 年間の洪水発生地域と 95% 以上が重なるなら、その住所は洪水危険地域と判定できます。逆に、重なる面積が 95% 未満であれば、ビジネス プロセスに従って、その住所を除外プロセスに引き渡して手動で検証を行うことができます。

住所または交差点は、ポイント、ストリートセグメント沿いの住所 (ストリートセグメント ポイントの配列)、ZIP + 4 セントロイド、ZIP + 2 セントロイド、または ZIP Code セントロイド (ポリゴン) にジオコードできます。これらの形状 (ポイント、ライン、またはポリゴン) を他の形状と比較して重なり合う部分があるかどうかを調べ、その結果を根拠としてリスクや可能性を判定できます。

モジュールから返される GeoConfidence 結果によって生成されるポリゴンは異なります。Enterprise Geocoding モジュールによって返される GeoConfidence 情報の詳細については、Enterprise Geocoding モジュールのドキュメントを参照してください。

GeoConfidence モジュールは、米国の住所のみに対応します。

> **注：** GeoConfidence では、Enterprise Geocoding モジュールおよび Location Intelligence モジュールから提供されるサービスが利用されます。

コンポーネント

GeoConfidence から展開される 3 つのデータフローは、Enterprise Designer を使って変更できます。各データフローはさまざまなコンポーネントによって構成されますが、それらのコンポーネントは Enterprise Geocoding モジュールや Location Intelligence モジュールに付属する形でインストール済みです。

インストール済みのデータフローの各コンポーネントの詳細については、『*Spectrum™ Technology Platformユーザ ガイド*』の関連コンポーネントの章を参照してください。

以下のデータフローが使用可能です。

- **GeoConfidenceSurface** ― 詳細な分析に利用できる GeoConfidence Surface を作成するデータフローです。入力は、Enterprise Geocoding モジュールから返された GeoConfidence 情報です。現在、Geocode US Address ステージのみがこの情報を返すことができます。
- **CreatePointsConvexHull** ― GeoConfidenceSurface テンプレートで使われるサブフローです。通常、このサブフローを変更する必要はありません。
- **FloodRiskAnalysis** ― サンプルのデータフローです。

GeoConfidence データベース

GeoConfidence が使うデータベースは、Enterprise Geocoding モジュールおよび Location Intelligence モジュールと同じです。

これらのデータベースを追加する方法については、『*Spectrum™ Technology Platform 管理ガイド*』を参照してください。

これらのデータベースのほかに、GeoConfidence モジュールには ZIP Code ポリゴンのデータベースが含まれます。これは GeoConfidence Surface で使われます。

GeoConfidence Surface

GeoConfidence Surface は、Enterprise Geocoding モジュールによって生成されるジオコード情報の品質に基づいて地理信頼性ポリゴン（等高線とも呼ばれます）を返します。地理信頼性ポリゴンが生成されたら、そのポリゴンに他の空間データを重ねてリスクや確率を明らかにできます。

このサービスは、GeoConfidence モジュールの FloodZoneAnalysis データフロー テンプレートによって使用されます。

> **注：** GeoConfidence では、Enterprise Geocoding モジュールおよび Location Intelligence モジュールから提供されるサービスが利用されます。

入力

GeoConfidence Surface の入力フィールドは、Enterprise Geocoding モジュールの GeoConfidence 出力カテゴリから返された出力フィールドです。これらのフィールドについて、以下に詳しく説明します。

columnName	最大nullターミネータによる長さフィールド	説明
GeoConfidenceCode	13	このフィールドに返される値は、どのタイプの Geoconfidence Surface が返されたかを示します。 有効な値を次に示します。 **INTERSECTION** 2 つの通りが交差する位置のジオコード ポイント。 **ADDRESS** 住所が位置付けられた通りセグメントを表す、通りセグメント ポイントの配列。 **POINT** ジオコーダがポイント データを使って住所のマッチングに成功した場合、その住所が位置付けられたポイント ジオメトリ。 **POSTAL1** ZIP セントロイドのジオコード ポイント。 **POSTAL2** 住所が位置付けられた ZIP + 2 に含まれるすべての通りセグメントのポイント配列。 **POSTAL3** 住所が位置付けられた ZIP + 4 に含まれる通りセグメントのポイント配列。 **ERROR** エラーが発生しました。
StreetSegmentPoints	1024	通りセグメント ポイントを表す緯度/経度値の配列。 **注:** このフィールドには、`GeoConfidenceCode` フィールドが `ADDRESS`,`POSTAL2`、または `POSTAL3`。
GeoConfidenceCentroidLatitude	11	Geoconfidence ポリゴンのセントロイドの緯度。
GeoConfidenceCentroidLongitude	12	Geoconfidence ポリゴンのセントロイドの経度。

出力

[GeoConfidenceSurface] 出力フィールドには、Geoconfidence ポリゴンが格納されます。

フィールド名	説明
Geometry	返されたジオメトリを表す Geoconfidence ポリゴン。

GeoConfidence モジュールのカスタマイズ

GeoConfidence モジュールから展開される 3 つのデータフロー テンプレートは、Enterprise Designer を使って変更できます。各データフローはさまざまなコンポーネントによって構成されますが、それらのコンポーネントは Enterprise Geocoding モジュールや Location Intelligence モジュールに付属する形でインストール済みです。

次のデータフロー テンプレートを使用できます。

- **GeoConfidenceSurface** — 詳細な分析に利用できる GeoConfidence Surface を作成するテンプレートです。入力は、Enterprise Geocoding モジュールから返された GeoConfidence 情報です。現在、Geocode US Address ステージのみがこの情報を返すことができます。このテンプレートをカスタマイズするには、少なくとも 5 桁の ZIP Code 空間ソースを ZIP ステージ (Query Spatial) で指定する必要があります。
- **CreatePointsConvexHull** — GeoConfidenceSurface テンプレートで使われるサブフローです。通常、このサブフローを変更する必要はありません。
- **FloodRiskAnalysis** — サンプルのテンプレートです。このテンプレートをカスタマイズするには、少なくとも Flood 空間ソースを Find Nearest ステージで指定する必要があります。

Universal Addressing モジュール

Universal Addressing モジュール

Universal Addressing モジュールは、住所品質モジュールで、住所の正規化とバリデーションを実行して、郵便物の配達品質を高めることができます。Universal Addressing モジュールを使用すると、住所データに対して郵便当局が定める品質規格への準拠を徹底できます。住所がこれら

の規格に準拠していれば、郵便物を規定の配達日数でより確実に配達できます。また、差出人も、これらの規格に準拠すれば、郵便料金の大幅な割引を受けることができます。米国における郵便料金の割引については、**www.usps.com** にある USPS *Domestic Mail Manual* (*DMM*) を参照してください。カナダにおける郵便料金の割引については、カナダ郵便公社の Web サイト **www.canadapost.ca** を参照してください。オーストラリアにおける郵便料金の割引については、オーストラリア郵便公社の Web サイト **www.auspost.com.au** を参照してください。

Universal Addressing モジュールは、供与されているライセンスに応じて、バッチ モード、リアルタイム モード、またはホステッド サービスとして使用できます。バッチ バージョンの Universal Addressing モジュールは USPS™ による CASS 認定®です。また、オーストラリア郵便当局による AMAS 認定でもあります。

Spectrum™ Technology Platformでは、2 つの住所品質モジュールを使用でき、Universal Addressing モジュールはその 1 つです。もう 1 つは Address Now モジュールです。このモジュールは、米国とカナダ以外の住所のサポートが強化されており、より多くの国のバリデーションが可能で、2 バイトにも対応しています。国際住所データが大量にある場合は、住所の正規化とバリデーションの両面で利点があることから、Address Now モジュールの使用を検討してください。

コンポーネント

Universal Addressing モジュールは、次のコンポーネントで構成されます。適切なデータベース (独自の環境で Universal Addressing を稼働している場合)、またはホステッド サービス (Pitney Bowesホステッド サービスを介して Universal Addressing を利用している場合) のライセンスを取得していれば、これらのコンポーネントを、米国、カナダ、オーストラリア、および国際住所に対して適用できます。

- **AutoCompleteLoqate** — フォームに入力された各文字に基づいて瞬時に結果が返され、正確なデータのみがデータベースに入力されることを保証します。
- **GetCandidateAddresses** — 指定された住所に一致する可能性のあるもののリストを返します。
- **GetCandidateAddressesLoqate** — Loqate エンジンとデータベースを使用して、指定された住所に一致する可能性のあるもののリストを返します。
- **GetCityStateProvince** — 指定された郵便番号に対する都市および州または省を返します。
- **GetCityStateProvinceLoqate** — Loqate エンジンとデータベースを使用して、指定された郵便番号に対する都市および州または省を返します。
- **GetPostalCodes** — 指定された都市の郵便番号を返します。
- **GetPostalCodesLoqate** — Loqate エンジンとデータベースを使用して、指定された都市の郵便番号を返します。
- **ValidateAddress** — 米国、カナダ、および国際郵便データを使用して、住所を正規化し、妥当性を確認します。
- **ValidateAddressAUS** — オーストラリアの郵便データを使用して、住所を正規化し、妥当性を確認します。

- **ValidateAddressGlobal** — ValidateAddressGlobal 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。ValidateAddressGlobal 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、ValidateAddressGlobal の使用を検討してください。
- **ValidateAddressLoqate** — ValidateAddressLoqate 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。ValidateAddress 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、州または省名など、欠落している郵便情報を追加します。

Universal Addressing データベース

Universal Addressing モジュールは、いくつかの必須データベースとオプション データベースを使用します。これらのデータベースは Spectrum™ Technology Platform サーバーにインストールされます。一部のデータベースは、Pitney Bowes が提供するサブスクリプションによって利用可能で、月に 1 回、または年に 4 回更新されます。その他のデータベースは、USPS® がライセンス提供しています。次の表に、Universal Addressing データベースの一覧を示します。

表 63 : Universal Addressing モジュールのデータベース

データベース名とその説明	必須またはオプションの区別	提供元
米国郵便データベース 米国郵便データベースは、Pitney Bowes 独自のフォーマットで提供されています。米国内のすべての家番号範囲が含まれており、月に 1 回更新されます。このデータベース ファイルには、次の情報が含まれています。 • ZIP + 4® Code • 正規化済みの住所要素 • 都市および州の情報 米国郵便データベースには、Enhanced Street Matching (ESM) および All Street Matching (ASM) の実行に必要なデータも含まれています。ESM および ASM は、通常の住所検証プロセスでマッチしなかった任意の入力住所に対して、追加のマッチング ロジックを適用します。	米国の処理を有効にする	Pitney Bowes サブスクリプション (月 1 回更新)

データベース名とその説明	必須またはオプションの区別	提供元
カナダ郵便データベース カナダ郵便データベースは、Pitney Bowes 独自のフォーマットです。このデータベース ファイルには、次の情報が含まれています。 • 郵便番号 • 正規化済みの住所要素 • 自治体および州の情報	カナダ住所処理に必須	Pitney Bowes サブスクリプション (月 1 回更新)
オーストラリア郵便公社の郵便住所ファイル データベース 郵便住所ファイルは、オーストラリア郵便公社の Address Matching Approval System (AMAS) プログラムに含まれています。このデータベース ファイルには、次の情報が含まれています。 • 郵便番号 • 正規化済みの住所要素	オーストラリア住所処理に必須	Pitney Bowes サブスクリプション (月 1 回更新)
国際郵便データベース 国際郵便データベースは、世界各地の郵便住所データの集まりです。各国のデータは、提供されているデータのレベルに応じて分類されています。カテゴリは次のとおりです。 • **カテゴリ A —** 住所の郵便番号、都市名、州/郡名、ストリートの住所要素、および国名の検証と修正が可能です。 • **カテゴリ B —** 住所の郵便番号、都市名、州/郡名、および国名の検証と修正が可能です。ストリートの住所要素の検証または修正はサポートしません。 • **カテゴリ C —** 国名の検証および修正と、郵便番号の書式の検証が可能です。	国際住所処理に必須	Pitney Bowesサブスクリプション (年 4 回更新)

データベース名とその説明	必須またはオプションの区別	提供元
DPV® データベース Delivery Point Validation Database は、米国の郵送先住所の妥当性をチェックするために使用できます。DPV データベースにより、米国郵便データベースによる郵便住所の検証能力を高めることができます。 **注：** DPV データベースには、Commercial Mail Receiving Agency (CMRA: 民間私書箱) の処理に必要なデータも含まれています。 米国郵便データベースの新しいエディションが提供される度に、DPV データベースの対応するエディションが提供されます。USPS ライセンスでは、有効期限を過ぎても米国郵便データベースを使用することを許可しますが (一部制約があります)、DPV データベースの有効期限後に、DPV 検索を実行することはできません。 USPS ライセンスでは、DPV データを、住所または住所一覧の作成に使用することを禁じています。住所一覧の作成を防止するために、DPV Database には「誤検出レコード」が含まれています。誤検出レコードとは、人為的に作成された住所のことです。DPV クエリでマッチしなかった場合は、DPV データベース内の誤検出テーブルに対するクエリが実行されます。このテーブルにマッチする場合、DPV の処理は停止します。 USPS ライセンスでは、米国外に DPV データを輸出することも禁じています。	オプション。ただし、CASS 認定™の処理には必須。米国住所のみに対応	Pitney Bowes サブスクリプション (月 1 回更新)
eLOT® データベース Enhanced Line of Travel (eLOT) データベースは、Enhanced Carrier Route の郵送が実際の配達順序にできる限り近くなることを保証する米国住所データベースです。eLOT データベースは、一部の種類の郵便料金割引を受けるために必須です。 eLOT データベースに対する毎月の更新情報は、米国郵便データベースと同じメディアで提供されます郵便データベースの最新版に掲載されていないことが必要です。 同一月の米国郵便データベースと eLOT データベースをインストールする必要があります (つまり、9 月の eLOT データは、9 月の米国郵便データベースで処理しなければなりません)。米国郵便データベースと eLOT データベースが同一月のものでない場合、eLOT 番号を割り当てられない ZIP + 4® Code が存在する恐れがあります。eLOT コードを割り当てるには、住所の ZIP Code™、ZIP + 4 Code、配達ルート コード、および配達ポイントが提供される必要があります。	オプション。米国住所のみに対応	Pitney Bowes サブスクリプション (月 1 回更新)

データベース名とその説明	必須またはオプションの区別	提供元
EWS データベース Early Warning System (EWS) データベースは、米国郵便データベースの郵便データの更新遅れに起因する住所検証の誤りを防ぎます郵便データベースの最新版に掲載されていないことが必要です。 EWS データベースは、ZIP Code™、ストリート名、接頭および接尾方向指示、接尾語という一部の住所情報のみで構成されています。住所が米国郵便データベースの最新版には存在しない場合に限り、住所レコードに EWS を適用できます郵便データベースの最新版に掲載されていないことが必要です。 USPS® は、EWS ファイルを週に 1 回 (木曜日) 更新します。USPS® Web サイト **ribbs.usps.gov** から EWS ファイルをダウンロードできます。	オプション。米国住所のみに対応	USPS® Web サイトから無償でダウンロード可能
LACS^{Link}® データベース LACS^{Link} データベースを使って、地方配送路の住所のストリート名に沿った住所への変更、PO Box 番号の再割り当て、またはストリート名に沿った住所の変更に伴って変更された住所を訂正できます。 USPS ライセンスでは、LACS^{Link} データを、住所または住所一覧の作成に使用することを禁じています。住所一覧の作成を防止するために、LACS^{Link} データベースには「誤検出レコード」が含まれています。誤検出レコードとは、人為的に作成された住所のことです。LACS^{Link} クエリでマッチしなかった場合は、LACS^{Link} データベース内の誤検出テーブルに対するクエリが実行されます。このテーブルにマッチすると、LACS^{Link} 処理は停止します。 USPS ライセンスでは、米国外に LACS^{Link} データベースを輸出することも禁じています。	オプション。ただし、CASS 認定™の処理には必須。米国住所のみに対応	Pitney Bowes サブスクリプション (月 1 回更新)
RDI™ データベース Residential Delivery Indicator (RDI™) データベースには、郵送物に対する最良の配送料を調べることができるデータが含まれます。 RDI は、RDI データがハッシュ テーブルとして提供される点で、DPV に似ています。しかし、住所全体ではなく 9 桁および 11 桁の ZIP Code™ に対してのみ標準ハッシュ アルゴリズムが決定されるため、RDI は DPV よりもはるかにシンプルな処理です。	オプション。米国住所のみに対応	USPS® から直接ライセンス

データベース名とその説明	必須またはオプションの区別	提供元
Suite[Link™] データベース Suite[Link] は、補助的な住所情報の妥当性が確認できなかった米国の企業住所に対し、その補助的な住所情報を修正します。Suite[Link] 処理が有効な場合、ValidateAddress は FirmName フィールドの値を既知の会社名のデータベースに照合します。その後、ValidateAddress は正しい補助的な住所情報を提供します。	オプション。米国住所のみに対応	Pitney Bowes サブスクリプション (月 1 回更新)

AutoCompleteLoqate

AutoCompleteLoqate は、住所データのリアルタイム入力に対して高速で正確な結果を返します。フォームに入力された各文字に基づいて瞬時に結果が返され、正確なデータのみがデータベースに入力されることを保証します。AutoCompleteLoqate には [高度な検索を優先] オプションも用意されており、インデックス ファイル形式のデータを使用することによって、238 カ国に対して入力時間を最大 80% 短縮します。

入力

AutoCompleteLoqate の入力の一覧を以下の表に示します。

表 64 : 入力フォーマット

フィールド名	説明
AddressLine1	最初の住所行。
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。
AddressLine4	4 行目の住所行。

フィールド名	説明
City	都市名。
Country	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 桁の ISO 国コード • 3 桁の UPU 国コード • 英語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	会社名または企業名。
PostalCode	住所の郵便番号。
StateProvince	州または省。

オプション

表 65 : AutoCompleteLoqate のオプション

optionName	説明
Database.Loqate	住所処理に使用するデータベースを指定します。Management Console の **[データベース リソース]** パネルで定義されたデータベースのみが使用可能です。

optionName	説明
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力に大文字と小文字を混在させます (デフォルト)。例: 123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。例: 123 MAIN ST MYTOWN FL 12345
HomeCountry	デフォルト国を指定します。大部分の住所が存在する国を指定してください。例えば、処理する住所の大部分がドイツにある場合は、ドイツを指定します。有効な国名には次のものがあります。 Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe

optionName	説明
OutputCountryFormat	**Country** 出力フィールドに返される国名に使用するフォーマットを指定します。例えば、英語を選択した場合、"Deutschland" という国名は "Germany" として返されます。 **E** 英語の国名を使用します (デフォルト)。 **I** 国名の代わりにその国の 2 文字の ISO の略語を使用します。 **U** 国名の代わりにその国の万国郵便連合 (Universal Postal Union: UPU) の略語を使用します。
OutputScript	出力がどのアルファベットまたはスクリプトで返されるかを指定します。このオプションは双方向で、通常はネイティブからラテン文字へ、およびラテン文字からネイティブへ実行されます。 **Input** 書き直しを実行せず、入力と同じスクリプトで出力します(デフォルト)。 **Native** 使用可能な場合は、選択した国のネイティブ スクリプトで出力します。 **Latn** 英語の値を使用します。
MaximumResults	AutoCompleteLoqate が返す住所の最大数。デフォルトは 10 です。

optionName	説明
isPowersearchEnable	インデックス ファイル形式のデータを使用することによって、240 カ国に対して入力時間を最大 80% 短縮します。検索を実行すると、Loqate エンジンはまず、該当するインデックスを検索します。インデックスが存在する場合は、候補住所のリストを直ちに返そうと試みます。インデックスが存在しないか、インデックスによって結果が1つも返されない場合は、オリジナルの検索処理を開始します。 **注：** 高度な検索は、入力ファイルにフィールドが 2つだけ存在する場合に実行可能です。1 つは Country フィールドで、もう 1 つはいずれかの AddressLine フィールドです。このオプションを選択し、入力ファイルにそれ以外のフィールドが含まれる場合は、オリジナルの検索処理が自動的に開始されます。 検索を行うために、Auto Complete インデックスは米国内の検索に対しては最初の 10 文字まで、その他すべての対象国内の検索に対しては最初の 15 文字までを使用します。空白と句読文字は、この文字数にカウントされません。 高度な検索は、ボツワナ、エチオピア、インド、カザフスタン、マレーシア、モンゴル、セントクリストファー・ネイビス、およびサンマリノでは使用できません。 **注：** 高度な検索を使用するには、有効なライセンスが必要です。高度な検索のライセンスを取得していないか、ライセンスの期限が切れている状態でこのオプションを選択すると、エラーとなります。
IsDuplicateHandlingMaskEnable	重複処理マスクを有効にし、重複レコードの処理および削除の方法を指定します。次のオプションから 1 つ以上を選択します。 **S** デフォルトで選択されています。入力の前処理により、単一フィールドで発生している重複を削除します。 **C** デフォルトで選択されています。入力の前処理により、すべてのフィールドわたって重複を削除します。 **T** 入力の前処理により、標準住所フィールドでないフィールド内の重複を削除します。 **F** デフォルトで選択されています。検証の出力の後処理により、検証されていないフィールドから重複を削除します。

optionName	説明
FailJobOnDataLicenseError	データ ライセンス エラーの発生時に Spectrum Technology Platform がどのように応答するかを指定します。 **ジョブのエラー** データライセンスエラーが発生した場合、ジョブ全体をエラーにします。 **レコードのエラー** データライセンスエラーの発生原因となったレコードをエラーにし、処理を続行します。

出力

AutoCompleteLoqate の出力はオプションであり、[AutoCompleteLoqate オプション] ダイアログボックスの [出力フィールド] セクションで選択したフィールドに直接対応します。

表 66 : AutoCompleteLoqate の出力

フィールド名	説明
AddressLine1	最初の住所行。
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。
AddressLine4	4 行目の住所行。
City	都市名。
Country	3 文字の ISO 3116-1 Alpha-3 国コード。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	企業名。

フィールド名	説明
HouseNumber	候補住所の家番号が含まれる範囲の終了家番号。
PostalCode	郵便番号。
PostalCode.AddOn	ZIP + 4® Code の末尾 4 桁。
ProcessedBy	住所を処理した住所コーダーを示します。 **LOQATE** Loqate コーダーが住所を処理しました。
StateProvince	州または省の省略形。
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。 • DisabledCoder • RequestFailed • NoLookupAddressFound
Status.Description	問題の説明 (ある場合)。 **Did not return multiples** 入力住所はデータベース内の 1 つの住所とのみマッチしました。AutoCompleteLoqate は、一致する可能性のある住所が複数見つかった場合のみデータを返します。 **Not able to look up the address pattern** AutoCompleteLoqate は、部分的な住所を処理できません。

GetCandidateAddresses

GetCandidateAddresses は、与えられた入力住所にマッチするとみなされる住所のリストを返します。GetCandidateAddresses は、入力住所が郵便データベースの複数の住所にマッチする場合のみ、候補の住所を返します。入力住所が、郵便データベースの 1 つの住所のみにマッチする場合は、住所データを返しません。

米国およびカナダ以外の住所については、ValidateAddress が返す複数のマッチ結果と、GetCandidateAddresses が同じ住所に対して返す結果の間に、矛盾が存在する場合があります。矛盾した結果が得られるのはおそらく、ValidateAddress のパフォーマンス チューニング設定で 100 以外の値を設定しているためです。GetCandidateAddresses と ValidateAddress で矛盾のない結果を得るには、パフォーマンス チューニング オプションを 100 に設定します。

> **注：** デフォルトでは、GetCandidateAddresses は個々の家番号との一致は確認しません。各ストリートの家番号の範囲とのマッチングを行います。GetCandidateAddresses はストリート名、都市名、州/省名、および郵便番号を特定した後、入力された家番号が、マッチしたストリート名の家番号の範囲に含まれるかどうかを確認します。ユニット番号についても同様の処理が行われます。個々の家番号が有効であることを確認するには、ValidateAddress Delivery Point Validation (DPV) 処理オプションを使用する必要があります。DPV 処理は、米国住所に対してのみ使用可能です。

カナダのコーダーには、特定の郵便番号を入力として受け取り、その郵便番号のデータベースに格納されたストリート情報を返す逆検索ルーチンが含まれています。この機能を使用するには、PostalCode フィールドにカナダの郵便番号を入力します。カナダの郵便番号を入力した場合の結果については、以下の 2 つめの例を参照してください。

GetCandidateAddresses は、Universal Addressing モジュールに含まれています。

入力

GetCandidateAddresses の入力の一覧を以下の表に示します。

表 67：入力フォーマット

フィールド名	説明
AddressLine1	最初の住所行。

フィールド名	説明
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。 米国およびカナダのアドレスには適用されません。
AddressLine4	4 行目の住所行。 米国およびカナダのアドレスには適用されません。
AddressLine5	5 行目の住所行。 英国住所のみに適用できます。ストリート名、ユニット番号、ビルディング番号などを含めることができます。
City	都市名。
StateProvince	州または省。 米国住所に対しては、州は、StateProvince フィールドではなく、City フィールドに入力することもできます。
PostalCode	住所の郵便番号。米国住所の場合は、次のいずれかの形式の ZIP Code™ になります。 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 **注：** カナダ住所の場合は、このフィールドのみを入力すると、候補となる住所データが返されます。その他の国の場合は、AddressLine1 と AddressLine2 も入力する必要があります。

フィールド名	説明
Country	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 桁の ISO 国コード • 3 桁の UPU 国コード • 英語の国名 • フランス語の国名 • ドイツ語の国名 • スペイン語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	会社名または企業名。
USUrbanName	米国住所都市化名。主にプエルトリコの住所に使用します。

オプション

表 68 : GetCandidateAddresses オプション

optionName	説明
PerformUSProcessing	米国住所を処理するかどうかを住所をサポートしていません。米国住所処理を有効にすると、GetCandidateAddresses は、米国住所の候補となる住所の取得を試みます。米国住所処理を無効にすると、米国住所は失敗します。つまり、Status 出力フィールドに "F" が設定されて返されます。出力フィールド Status.Code は、"DisabledCoder" となります。米国住所処理のライセンスを取得していない場合は、ジョブに米国住所が含まれるか否かにかかわらず、米国住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注 :** 米国住所を正常に処理するには、米国住所処理の有効なライセンスを取得する必要があります。米国住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、米国住所処理を有効にすると、エラーとなります。 **Y** はい、米国住所を処理します (デフォルト)。 **N** いいえ、米国住所をサポートしていません。
Database.US	米国住所処理に使用するデータベースを指定します。Management Console の **[米国データベース リソース]** パネルで定義されたデータベースのみが使用可能です。

optionName	説明
PerformCanadianProcessing	カナダ住所を処理するかどうかを指定します。カナダ住所処理を有効にすると、GetCandidateAddresses は、カナダ住所の候補となる住所の取得を試みます。カナダ住所処理を無効にした場合、Status フィールドに "F" が設定されカナダ住所は失敗します。出力フィールド Status.Code は、"DisabledCoder" となります。カナダ住所処理のライセンスを取得していない場合は、ジョブにカナダ住所が含まれるか否かにかかわらず、カナダ住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注：** カナダ住所を正常に処理するには、カナダ住所処理の有効なライセンスを取得する必要があります。カナダ住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、カナダ住所処理を有効にすると、エラーとなります。 **Y** カナダ住所を処理します (デフォルト)。 **N** カナダ住所を処理しません。
Database.Canada	カナダ住所処理に使用するデータベースを指定します。Management Console の **[カナダ データベース リソース]** パネルで定義されたデータベースのみが使用可能です。
PerformInternationalProcessing	国際住所 (米国およびカナダ以外の住所) を処理するかどうかを指定します。国際住所処理を有効にすると、GetCandidateAddresses は、国際住所の候補となる住所の取得を試みます。国際住所処理を無効にした場合、Status フィールドに "F" が設定され国際住所は失敗します。出力フィールド Status.Code は、"DisabledCoder" となります。国際住所処理のライセンスを取得していない場合は、ジョブに国際住所が含まれるか否かにかかわらず、国際住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注：** 国際住所を正常に処理するには、国際住所処理の有効なライセンスを取得する必要があります。国際住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、国際住所処理を有効にすると、エラーとなります。 **Y** 国際住所を処理します (デフォルト)。 **N** 国際住所を処理しません。

optionName	説明
Database.International	国際的な住所の検証に使用するデータベースを指定します。Management Console の **[国際データベース リソース]** パネルで定義されたデータベースのみが使用可能です。
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力に大文字と小文字を混在させます (デフォルト)。例: 123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。例: 123 MAIN ST MYTOWN FL 12345
MaximumResults	GetCandidateAddresses が返す候補住所の最大数。デフォルトは 10 です。最大数は 10 になります。
OutputShortCityName	米国住所に対しては、USPS® が承認する都市の略称がある場合に、それを返すかどうかを指定します。USPS® は、14 文字以上の都市名に対し、略称を定めています。都市の略称は 13 文字以下で、宛名ラベルのサイズが限られている場合に使用できます。短い都市名が存在しない都市に対しては、正式な都市名が返されます。 **Y** 短い都市名を返します。 **N** 短い都市名を返しません。

optionName	説明
DualAddressLogic	(米国住所のみ)。ストリート情報と PO Box/地方配送路/Highway Contract 情報の両方が住所に含まれる場合に、GetCandidateAddresses が、ストリート一致を返すか、または PO Box/地方配送路/Highway Contract 一致を返すかを制御します。詳細については、二重住所ロジックについて（422ページ）を参照してください。 **N** (デフォルト) USPS®CASS™ の規則では、以下の優先順位に基づいて、返す住所を決定します。 1. PO Box 2. Firm 3. Highrise 4. ストリート 5. Rural Route 6. General Delivery **S** 住所行に関係なく、ストリート一致を返します。 **P** 住所行に関係なく、PO Box 一致を返します。
StreetMatchingStrictness	ストリート名のマッチングの精度 (米国住所のみ)。 **E** 入力されたストリート名は、データベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です (デフォルト)。 **L** マッチング アルゴリズムは "あいまい" です。
FirmMatchingStrictness	企業名マッチングの精度 (米国住所のみ)。 **E** 入力された企業名は、データベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です (デフォルト)。 **L** マッチング アルゴリズムは "あいまい" です。

optionName	説明
DirectionalMatchingStrictness	道順マッチングの精度。 **E** 入力された道順は、データベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です (デフォルト)。 **L** マッチング アルゴリズムは "あいまい" です。
PerformESM	Enhanced Street Matching (ESM) を実行するかどうかを指定します。ESM は、通常の住所検証プロセスでマッチしなかった任意の入力住所に対して、追加データによる別のマッチング ロジックを適用します。ESM は、米国住所にのみ適用されます。 **Y** ESM 処理を実行します。 **N** ESM 処理を実行しません (デフォルト)。
AddressLineSearchOnFail	ValidateAddress において、住所行で都市、州/省、郵便番号の検索を行うかどうかを指定します。 このオプションにより、ValidateAddress において、City、StateProvince、および PostalCode の各入力フィールドの値を使用して住所にマッチする結果が得られなかった場合に、AddressLine 入力フィールドで都市、州/省、郵便番号、および国を検索することができます。 入力住所において、AddressLine フィールドに都市、州/省、および郵便番号の情報が存在する場合は、このオプションを有効にすることを検討してください。 入力住所において、City、State/Province、および PostalCode フィールドが使用されている場合は、このオプションを無効にしてください。このオプションを有効にしてこれらのフィールドを使用すると、ValidateAddress がこれらのフィールド値の修正 (例えば、スペルミスのある都市名など) に失敗する可能性が高くなります。 **Y** 住所行フィールドを検索します (デフォルト)。 **N** いいえ、AddressLine フィールドを検索しません。

出力

GetCandidateAddresses は、次の出力を返します。

表 69 : GetCandidateAddresses の出力

フィールド名	説明
AddressLine1	最初の住所行。
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。
AddressLine4	4 行目の住所行。
AddressLine5	英国住所専用です。住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 5 行目です。住所の妥当性が確認できなかった場合は、入力住所の 5 行目がそのまま出力されます。
City	都市名。
Country	3 文字の ISO 3116-1 Alpha-3 国コード。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	企業名。
HouseNumberHigh	候補住所の家番号が含まれる範囲の終了家番号。
HouseNumberLow	候補住所の家番号が含まれる範囲の開始家番号。
HouseNumberParity	HouseNumberLow と HouseNumberHigh の間の家番号の番号付けスキームを次のように示します。 **E** 偶数値のみ **O** 奇数値のみ **B** 両方

フィールド名	説明
MatchLevel	米国およびカナダ以外の住所に対し、候補住所のマッチ レベルを示します。米国米国およびカナダの住所では常に "A" になります。次のいずれかです。 **A** 候補はストリートレベルで入力住所にマッチします。 **B** 候補は州/省レベルで入力住所にマッチします。
PostalCode	郵便番号。米国では、ZIP Code™になります。
PostalCode.AddOn	ZIP + 4® Code の末尾 4 桁。米国住所にのみ適用されます。
RecordType	米国およびカナダの郵政当局によって定義されている住所レコードのタイプ (米国およびカナダの住所のみサポート): • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal
RecordType.Default	"デフォルト" マッチを示すコード **Y** 住所はデフォルト レコードにマッチしています。 **NULL** 住所はデフォルト レコードにマッチしていません。
StateProvince	州または省の省略形。
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次のいずれかの値になります。 • DisabledCoder • RequestFailed

フィールド名	説明	
Status.Description	問題の説明 (ある場合)。	
	Did not return multiples	入力住所はデータベース内の 1 つの住所とのみマッチしました。GetCandidateAddresses は、一致する可能性のある住所が複数見つかった場合にデータを返します。
	Number of candidates is not greater than 1	入力住所はデータベース内の複数の住所にマッチしましたが、住所が返されませんでした。
	PerformUSProcessing disabled	Status.Code=DisabledCoder の場合にこの値が表示されます。
	PerformCanadianProcessing disabled	Status.Code=DisabledCoder の場合にこの値が表示されます。
	PerformInternationalProcessing disabled	Status.Code=DisabledCoder の場合にこの値が表示されます。
UnitNumberHigh	候補住所のユニット番号が含まれる範囲の終了ユニット番号。	
UnitNumberLow	候補住所のユニット番号が含まれる範囲の開始ユニット番号。	
UnitNumberParity	UnitNumberLow と UnitNumberHigh の間のユニット番号の番号付けスキームを次のように示します。	
	E	偶数値のみ
	O	奇数値のみ
	B	両方
USUrbanName	妥当性が確認された都市の都市化名。都市化名は、主にプエルトリコ住所に使用されます。	

GetCandidateAddressesLoqate

GetCandidateAddressesLoqate は、与えられた入力住所に一致するとみなされる住所のリストを返します。GetCandidateAddressesLoqate は、入力住所が郵便データベースの複数の住所にマッ

チする場合のみ、候補の住所を返します。入力住所が、郵便データベースの1つの住所のみにマッチする場合は、住所データを返しません。[Country] 入力フィールドは必須です。このフィールドが空白の場合、出力は返されません。

> **注：** デフォルトでは、GetCandidateAddressesLoqate は個々の家番号との一致は確認しません。各ストリートの家番号の範囲とのマッチングを行います。GetCandidateAddressesLoqate はストリート名、都市名、州/省名、および郵便番号を特定した後、入力された家番号が、マッチしたストリート名の家番号の範囲に含まれるかどうかを確認します。ユニット番号についても同様の処理が行われます。

GetCandidateAddressesLoqate は、Universal Addressing モジュールに含まれています。

入力

GetCandidateAddressesLoqate の入力の一覧を以下の表に示します。

表 70：入力フォーマット

フィールド名	説明
AddressLine1	最初の住所行。
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。
AddressLine4	4 行目の住所行。
City	都市名。

フィールド名	説明
Country	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 桁の ISO 国コード • 3 桁の UPU 国コード • 英語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。 **注：** このフィールドは必須です。このフィールドが空白の場合、出力は返されません。
FirmName	会社名または企業名。
PostalCode	住所の郵便番号。米国住所の場合は、次のいずれかの形式の ZIP Code™ になります。
StateProvince	州または省。 米国住所に対しては、州は、StateProvince フィールドではなく、City フィールドに入力することもできます。

オプション

表 71 : GetCandidateAddressesLoqate のオプション

optionName	説明
Database.Loqate	住所処理に使用するデータベースを指定します。Management Console で定義されたデータベースのみが使用可能です。

optionName	説明
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力に大文字と小文字を混在させます (デフォルト)。例: 123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。例: 123 MAIN ST MYTOWN FL 12345
CandidateProcessOption	候補を検索する方法を指定します。次のいずれかです。 **S** 住所の全体または一部を入力し、近似一致結果のリストを出力として返します (デフォルト)。 **V** 住所行、住所コンポーネント、またはその両方を組み合わせて住所情報を入力し、入力により近く一致する結果を出力として返します。

optionName	説明
HomeCountry	デフォルト国を指定します。大部分の住所が存在する国を指定してください。例えば、処理する住所の大部分がドイツにある場合は、ドイツを指定します。GetCandidateAddressLoqate は、[StateProvince]、[PostalCode]、および [Country] の各住所フィールドから国を特定できなかった場合、指定された国を使用して、住所の検証を試みます。有効な国名には次のものがあります。 Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe
OutputCountryFormat	**Country** 出力フィールドに返される国名に使用するフォーマットを指定します。例えば、英語を選択した場合、"Deutschland" という国名は "Germany" として返されます。 **E** 英語の国名を使用します (デフォルト)。 **I** 国名の代わりにその国の 2 文字の ISO の略語を使用します。 **U** 国名の代わりにその国の万国郵便連合 (Universal Postal Union: UPU) の略語を使用します。

optionName	説明
OutputScript	出力がどのアルファベットまたはスクリプトで返されるかを指定します。このオプションは双方向で、通常はネイティブからラテン文字へ、およびラテン文字からネイティブへ実行されます。 **Input** 書き直しを実行せず、入力と同じスクリプトで出力します(デフォルト)。 **Native** 使用可能な場合は、選択した国のネイティブ スクリプトで出力します。 **Latn** 英語の値を使用します。
MaximumResults	GetCandidateAddressesLoqate が返す候補住所の最大数。デフォルトは 10 です。最大数は 99 です。

出力

GetCandidateAddressesLoqate は、次の出力を返します。

表 72 : GetCandidateAddressesLoqate の出力

フィールド名	説明
AddressLine1	最初の住所行。
AddressLine2	2 行目の住所行。
AddressLine3	3 行目の住所行。
AddressLine4	4 行目の住所行。
City	都市名。

フィールド名	説明
Country	3 文字の ISO 3116-1 Alpha-3 国コード。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	企業名。
PostalCode	郵便番号。米国では、ZIP Code™になります。
PostalCode.AddOn	ZIP + 4® Code の末尾 4 桁。米国住所にのみ適用されます。
ProcessedBy	住所を処理した住所コーダーを示します。 **LOQATE** Loqate コーダーが住所を処理しました。
StateProvince	州または省の省略形。
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次のいずれかの値になります。 • RequestFailed
Status.Description	問題の説明 (ある場合)。次のいずれかの値になります。 **Did not return multiples** 入力住所はデータベース内の 1 つの住所とのみマッチしました。GetCandidateAddressesLoqate は、一致する可能性のある住所が複数見つかった場合のみデータを返します。

GetCityStateProvince

GetCityStateProvince は、与えられた入力郵便番号に対する都市および州/省を返します。

注： GetCityStateProvince は、米国およびカナダの住所のみをサポートします。

GetCityStateProvince は、Universal Addressing モジュールに含まれています。

入力

入力フィールドを以下の表に示します。

表 73 : GetCityStateProvince の入力

フィールド名	説明
PostalCode	米国企業のZIP Code™ またはカナダの郵便番号を次のいずれかのフォーマットで示します。 99999 99999-9999 A9A9A9 A9A 9A9

オプション

表 74 : Get City State Province Loqate

オプション名	説明
PerformUSProcessing	米国住所を処理するかどうかを指定します。米国住所処理を有効にした場合、GetCityStateProvince は米国住所の州を返します。米国住所処理を無効にすると、米国住所は失敗します。つまり、Status 出力フィールドに "F" が設定されて返されます。出力フィールド Status.Code は、"DisabledCoder" となります。米国住所処理のライセンスを取得していない場合は、ジョブに米国住所が含まれるか否かにかかわらず、米国住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注 :** 米国住所を正常に処理するには、米国住所処理の有効なライセンスを取得する必要があります。米国住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、米国住所処理を有効にすると、エラーとなります。米国住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、米国住所処理を有効にすると、エラーとなります。 **Y** はい、米国住所を処理します (デフォルト)。 **N** いいえ、米国住所を処理しません。
Database.US	米国住所処理に使用するデータベースを指定します。Management Console の **[米国データベース リソース]** パネルで定義されたデータベースのみが使用可能です。
PerformCanadianProcessing	カナダ住所を処理するかどうかを指定します。カナダ住所処理を有効にした場合、GetCityStateProvince はカナダ住所の州を返します。カナダ住所処理を無効にした場合、Status フィールドに "F" が設定されカナダ住所は失敗します。出力フィールド Status.Code は、"DisabledCoder" となります。カナダ住所処理のライセンスを取得していない場合は、ジョブにカナダ住所が含まれるか否かにかかわらず、カナダ住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注 :** カナダ住所を正常に処理するには、カナダ住所処理の有効なライセンスを取得する必要があります。カナダ住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、カナダ住所処理を有効にすると、エラーとなります。カナダ住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、カナダ住所処理を有効にすると、エラーとなります。 **Y** カナダ住所を処理します (デフォルト)。 **N** カナダ住所を処理しません。

オプション名	説明
Database.Canada	カナダ住所処理に使用するデータベースを指定します。Management Console の **[カナダ データベース リソース]** パネルで定義されたデータベースのみが使用可能です。
OutputVanityCity	非正式な都市名を出力に含めるかどうかを指定します。非正式な都市名は、主要都市名の代替名です。例えば、Hollywood は Los Angeles の非正式な都市名です。 **Y** 非正式な都市名を含めます。 **N** 非正式な都市名を含めません (デフォルト)。
MaximumResults	返される都市と州/省のペアの最大数を指定します。デフォルト値は 10 です。

出力

GetCityStateProvince は、入力郵便番号に一致する都市と州/省、およびマッチングの成功または失敗を示すコードを返します。複数の都市/州または都市/省が入力郵便番号にマッチする場合、複数の出力レコードが返されます。

表 75 : GetCityStateProvince の出力

フィールド名	説明
City	一致した都市名。
City.Type	USPS® によって正規化された都市名のタイプ (米国住所のみ)。住所のみ)。 **V** 非正式 (Non-Mailing) 都市名。 **P** 主要都市名。都市名は、主要な郵送都市名です。 **S** 補助的な都市名。都市名は、代替都市名ですが、許容されます。都市は複数の補助的な都市名を持つことができます。
PostalCode	入力郵便番号。

フィールド名	説明
ProcessedBy	住所を処理した住所コーダーを示します。次のいずれかです。 **USA** 米国住所コーダーが住所を処理しました。 **CAN** カナダ住所コーダーが住所を処理しました。
StateProvince	州または省の省略形。
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次の値のみが有効です。 • DisabledCoder • UnrecognizedPostalCode
Status.Description	失敗の説明。有効な値は次のとおりです。 **Postal code not found** Status.Code=UnrecognizedPostalCode の場合にこの値が表示されます。 **PerformUSProcessing disabled** Status.Code=DisabledCoder の場合にこの値が表示されます。 **PerformCanadianProcessing disabled** Status.Code=DisabledCoder の場合にこの値が表示されます。

GetCityStateProvinceLoqate

GetCityStateProvinceLoqate は、与えられた入力郵便番号に対する都市および州/省を返します。このステージは、Universal Addressing モジュールに含まれています。

入力

入力フィールドを以下の表に示します。

表 76 : GetCityStateProvinceLoqate の入力

フィールド名	説明
Country	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 桁の ISO 国コード • 3 桁の UPU 国コード • 英語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
PostalCode	住所の郵便番号。

オプション

表 77 : GetCityStateProvinceLoqate のオプション

フィールド名	説明 / 有効な値
Database.Loqate	住所処理に使用するデータベースを指定します。Management Console の **[データベース リソース]** パネルで定義されたデータベースのみが使用可能です。
MaximumResults	GetCityStateProvinceLoqate が返す必要がある住所の最大数。デフォルトは 10 です。
OutputScript	出力がどのアルファベットまたはスクリプトで返されるかを指定します。このオプションは双方向で、通常はネイティブからラテン文字へ、およびラテン文字からネイティブへ実行されます。 **Input** 書き直しを実行せず、入力と同じスクリプトで出力します (デフォルト)。 **Native** 使用可能な場合は、選択した国のネイティブ スクリプトで出力します。 **Latn** 英語の値を使用します。

フィールド名	説明 / 有効な値
FailJobOnDataLicenseError	データ ライセンス エラーの発生時に Spectrum Technology Platform がどのように応答するかを指定します。 **ジョブのエラー** データ ライセンス エラーが発生した場合、ジョブ全体をエラーにします。 **レコードのエラー** データ ライセンス エラーの発生原因となったレコードをエラーにし、処理を続行します。

出力

GetCityStateProvinceLoqate は、入力郵便番号に一致する都市と州/省、およびマッチングの成功または失敗を示すコードを返します。複数の都市/州または都市/省が入力郵便番号にマッチする場合、複数の出力レコードが返されます。

表 78 : GetCityStateProvinceLoqate の出力

フィールド名	説明
City	一致した都市名。
Country	OutputCountryFormat で選択した、以下のいずれかのフォーマットで示された国。 • ISO コード • UPU コード • 英語
PostalCode	入力郵便番号。
ProcessedBy	住所を処理した住所コーダーを示します。 **LOQATE** Loqate コーダーが住所を処理しました。
StateProvince	州または省の省略形。

フィールド名	説明
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次の値のみが有効です。 • UnrecognizedPostalCode
Status.Description	失敗の説明。次の値のみが有効です。 **Postal code not found** Status.Code=UnrecognizedPostalCode の場合にこの値が表示されます。

GetPostalCodes

GetPostalCodes では、特定の都市の郵便番号の検索が可能です。このサービスは、都市、州、および国を入力として受け取り、その都市の郵便番号を返します。入力を正しい順序で指定しなければ、郵便番号は返されません。

> **注：** GetPostalCodes は、米国住所にのみ対応します。

GetPostalCodes は、Universal Addressing モジュールに含まれています。

入力

GetPostalCodes は、都市、州/省、および国を入力として受け取ります。

表 79 : GetPostalCodes の入力

フィールド名	説明
City	郵便番号を検索する都市。 City フィールドに都市と州を入力できます。これを行う場合は、StateProvince フィールドを空白のままにする必要があります。 City および StateProvince フィールドの文字数が 100 文字を超えてはなりません。
StateProvince	郵便番号を検索する都市の州または省。 州は、StateProvince フィールドではなく、City フィールドに入力することもできます。 City および StateProvince フィールドの文字数が 100 文字を超えてはなりません。
Country	郵便番号を検索する都市の国コードまたは名前。有効な値は US のみです。

オプション

表 80 : GetPostalCodes のオプション

オプション名	説明
Database.US	郵便番号検索に使用するデータベースを指定します。Management Console の [米国データベース リソース] パネルで定義されたデータベースのみが使用可能です。
IncludeVanityCity	都市の非正式な都市名の郵便番号を含めるかどうかを指定します。非正式な都市名は、主要都市名の代替名です。例えば、Hollywood は Los Angeles の非正式な都市名です。 **Y** 非正式な都市名の郵便番号を含めます。 **N** 非正式な都市名の郵便番号を含めません (デフォルト)。

オプション名	説明
OutputCityType	都市タイプを出力で返すかどうかを指定します。有効にすると、都市タイプが City.Type フィールドに返されます。 **Y** 都市タイプを出力に含めます。 **N** 都市タイプを出力に含めません (デフォルト)。

出力

GetPostalCodes は、指定された都市の郵便番号を返します。各郵便番号は、以下の表に列挙されたデータとともにそれぞれ個別のレコードで返されます。

表 81 : GetPostalCodes の出力

フィールド名	説明
City.Type	USPS® の都市タイプ (米国住所のみ)。住所のみ)。都市タイプを判別するには、ZIP Code と都市名を調べます。例えば、メリーランド州ランハムの郵便番号は、20703、20706、および 20784 です。ランハムは、20703 と 20706 では主要都市ですが、20784 では非正式都市です。 このフィールド列に値が設定されるのは、`OutputCityType=Y`の場合のみです。有効な値を次に示します。 **V** 非正式 (Non-Mailing) 都市名。 **P** 主要都市名。都市名は、主要な郵送都市名です。 **S** 補助的な都市名。都市名は、代替都市名ですが、許容されます。都市は複数の補助的な都市名を持つことができます。
PostalCode	指定された都市の郵便番号。
ProcessedBy	このサービスは米国住所に対してのみ機能するため、ProcessedBy には常に USA という 1 つの値が含まれます。

フィールド名	説明
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次のいずれかです。 • CountryNotSupported • UnableToLookup
Status.Description	失敗の説明。 • 入力された国がサポートされていません • 入力された都市が空白でした • 入力された都市と州 / 省が空白であったか、一致が見つかりませんでした • 都市と州の不一致 (スペルの相違が見つかるか、都市/州が非正式都市であるが、非正式マッチングが許可されていないか、都市/州が ZIP Code と一致しない)

GetPostalCodes Loqate

GetPostalCodesLoqate では、特定の都市の郵便番号の検索が可能です。このサービスは、都市、州、および国を入力として受け取り、その都市の郵便番号を返します。入力を正しい順序で指定しなければ、郵便番号は返されません。

GetPostalCodesLoqate は、Universal Addressing モジュールに含まれています。

入力

GetPostalCodesLoqate は、都市、州/省、および国を入力として受け取ります。

表 82 : GetPostalCodesLoqate の入力

フィールド名	説明 / 有効な値
City	郵便番号を検索する都市。 City 列に都市と州を入力することができます。これを行う場合は、StateProvince 列を空白のままにする必要があります。
Country	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 桁の ISO 国コード • 3 桁の UPU 国コード • 英語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
StateProvince	郵便番号を検索する都市の州または省。 州は、StateProvince 列ではなく、City 列に入力することもできます。

オプション

表 83 : GetPostalCodesLoqate のオプション

オプション名	説明/有効値
Database.Loqate	郵便番号検索に使用するデータベースを指定します。Management Console で定義されたデータベースのみが使用可能です。
FailJobOnDataLicenseError	データ ライセンス エラーの発生時に Spectrum Technology Platform がどのように応答するかを指定します。 **ジョブのエラー** データ ライセンス エラーが発生した場合、ジョブ全体をエラーにします。 **レコードのエラー** データ ライセンス エラーの発生原因となったレコードをエラーにし、処理を続行します。

出力

GetPostalCodesLoqate は、指定された都市の郵便番号を返します。各郵便番号は、以下の表に列挙されたデータとともにそれぞれ個別のレコードで返されます。

表 84 : GetPostalCodesLoqate の出力

フィールド名	説明 / 有効な値
PostalCode	指定された都市の郵便番号。
ProcessedBy	住所を処理した住所コーダーを示します。 **LOQATE** Loqate コーダーが住所を処理しました。
Status	マッチの成功または失敗。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。次のいずれかです。 • InvalidCountry • UnableToLookup
Status.Description	失敗の説明。 • 入力された国がサポートされていません • 入力された都市が空白でした • 入力された都市と州 / 省が空白であったか、一致が見つかりませんでした

ValidateAddress 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。ValidateAddress 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。

ValidateAddress は、ValidateAddress が住所の妥当性を確認したかどうか、返し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、バリデーション処理に関する結果インジケータも返します。

ValidateAddress 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、ValidateAddress 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。

ValidateAddress は、Universal Addressing モジュールに含まれています。

入力

ValidateAddress は、入力として住所を受け取ります。住所がある国にかかわらず、すべての国がこのフォーマットを使用します。米国住所に対する住所行データの処理方法に関する重要な情報については、「**米国住所の住所行処理**（405ページ）」を参照してください。

表 85 : 入力フォーマット

フィールド名	書式	説明
AddressLine1	文字列 [50]	最初の住所行。
AddressLine2	文字列 [50]	2 行目の住所行。
AddressLine3	文字列 [50]	3 行目の住所行。 カナダの住所には適用されません。
AddressLine4	文字列 [50]	4 行目の住所行。 カナダの住所には適用されません。
AddressLine5	文字列 [50]	5 行目の住所行。 英国住所をサポートしていません。ストリート名、ユニット番号、ビルディング番号などを含めることができます。

フィールド名	書式	説明
City	文字列 [50]	都市名。 米国住所に対しては、米国の住所に限り、都市、州、および ZIP Code™ を City フィールドに入力することができます。これを行う場合は、StateProvince フィールドと PostalCode フィールドを空白のままにする必要があります。
StateProvince	文字列 [50]	州または省。 米国住所に対しては、州は、StateProvince フィールドではなく、City フィールドに入力することもできます。
PostalCode	文字列 [10]	住所に対する郵便番号を次のいずれかのフォーマットで示します。 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 米国住所に対しては、米国の住所に限り、 ZIP Code™ を City フィールドに入力することができます。 米国住所に対しては、都市/州/ZIP Code™ が PostalCode フィールドにある場合に、ValidateAddress がデータをパーシングして、住所を正しく処理することができます。最適な結果を得るため、このデータを適切なフィールド (City、StateProvince、PostalCode) に入力してください。
Country	文字列 [50]	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 文字の ISO 3116-1 Alpha-2 国コード • 3 文字の ISO 3116-1 Alpha-3 国コード • 英語の国名 • フランス語の国名 • ドイツ語の国名 • スペイン語の国名 ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	文字列 [50]	会社名または企業名。

フィールド名	書式	説明
USUrbanName	文字列 [50]	米国住所都市化名。主にプエル トリコの住所で使用されます。
CustomerID	文字列 [9]	この郵便物が汎用バーコードを使用している場合、USPS® が割り当てた顧客 ID をこのフィールドに指定します。ValidateAddress の汎用バーコードは、OneCode ACS® サービスを使用する郵便物に使用されます。
CanLanguage	文字列	カナダの住所に限り、`CanFrenchFormat=T` オプションが使用されている場合に、住所が英語かフランス語かを示します。 このフィールドが空白の場合、アドレスは英語でフォーマットされています。このフィールドに空白以外の値が含まれる場合、住所はフランス語でフォーマットされています。ケベックの住所は、このフィールドの値に関係なく常にフランス語でフォーマットされます。

米国住所の住所行処理

米国住所の場合、AddressLine1 から AddressLine4 の入力フィールドの処理方法は、企業名抽出または都市化コード抽出のオプションが有効になっているかどうかによって異なります。2 つのオプションのいずれかが有効になっている場合は、ValidateAddress は、4 つすべてのフィールドのデータを参照して、住所の妥当性を確認し、要求されたデータ (企業名または都市化コード) を抽出します。どちらのオプションも有効でない場合は、ValidateAddress は、空白でない最初の 2 つの住所行フィールドのみを使用して、妥当性を確認します。他の住所行フィールドのデータは、AdditionalInputData 出力フィールドに返されます。例を次に示します。

AddressLine1: A1 Calle A
AddressLine2:
AddressLine3: URB Alamar
AddressLine4: Pitney Bowes

この住所において、企業名抽出または都市化コード抽出が有効である場合は、ValidateAddress は、4 つすべての住所行を確認します。企業名抽出と都市化コード抽出がどちらも有効でない場合は、ValidateAddress は、AddressLine1 と AddressLine3 (空白でない最初の 2 つの住所行) を参照して、そのデータを使用して住所の妥当性を確認します。AddressLine4 のデータは、AdditionalInputData 出力フィールドに返されます。

オプション

出力データ オプション

以下の表に、ValidateAddress が返す情報の種類を制御するオプションの一覧を示します。これらのオプションのうちの一部は、カナダ住所に対してオーバーライドすることができます。詳細については、**カナダ住所のオプション**（439ページ）を参照してください。

表 86 : 出力データ オプション

オプション名	説明
OutputRecordType	出力レコードのタイプ。1 つ以上の場合はリストで提供されます。 **A** 住所データの 1 ～ 4 行に加えて、都市、州、郵便番号、企業名、および都市化名情報を返します。各住所行は、封筒に記載される住所の実際の行に対応しています。詳細については、**出力**（450ページ）を参照してください。住所の妥当性を確認できた場合は、住所行には正規化済み住所が含まれます。正規化済み住所では、句読文字が取り除かれ、方向指示とストリート接尾語には省略形が使用され、住所要素が修正されています。住所の妥当性を確認できなかった場合は、住所行には入力住所がそのまま含まれます ("パス スルー" データ)。`OutputRecordType=A` を指定しなかった場合でも、妥当性が確認されなかった住所は必ず、パス スルー データとして住所行フィールドに含まれます。 **E** パース済み住所要素。家番号、ストリート名、ストリート接尾語、方向指示などの住所の各要素が、個別のフィールドに返されます。詳細については、**パース済み住所要素出力**（452ページ）を参照してください。"E" を指定し、`OutputFormattedOnFail=Y` を指定した場合は、パース済み住所要素に、妥当性が確認できなかった住所の入力住所が含まれることになります。 **I** パース済み入力。このオプションでは、住所の妥当性を確認できたかどうかにかかわらず、入力住所をパース済み形式で返します。家番号、ストリート名、ストリート接尾語、方向指示などの入力住所の各要素が、個別のフィールドに返されます。パース済み入力 (値 "I") は、妥当性を確認できなかった入力だけでなく、すべての入力住所をパース済み形式で返す点で、`OutputRecordType=E` と `OutputFormattedOnFail=Y` の組み合わせとは異なります。詳細については、**パース済み入力**（455ページ）を参照してください。 **P** 郵便データ。出力住所には、妥当性が確認された各住所の追加データが含まれます。詳細については、**郵便データ出力**（457ページ）を参照してください。 **空白** 住所データまたは郵便データを返しません。

オプション名	説明
OutputFieldLevelReturnCodes	フィールドレベルの結果インジケータを含めるかどうかを指定します。フィールドレベルの結果インジケータは、各住所要素をどのように処理したかを示します。フィールドレベルの結果インジケータは、修飾子 "Result" で返されます。例えば、HouseNumber のフィールドレベルの結果インジケータは **HouseNumber.Result** に格納されます。結果インジケータの出力フィールドの完全な一覧は、フィールドレベルの結果インジケータ（464ページ）を参照してください。 **N** フィールドレベルのリターン コードを出力しません (デフォルト)。 **Y** フィールドレベルのリターン コードを出力します。

オプション名	説明
OutputFormattedOnFail	

オプション名	説明
	住所の妥当性を確認できない場合に書式を整えた住所を返すかどうかを指定します。住所には、その国の標準住所書式が設定されます。このオプションを選択しない場合、住所の妥当性を確認できないと、出力住所フィールドは空白になります。 **注：** このオプションは、米国およびカナダの住所のみに適用されます。その他の住所に対して書式を整えたデータは返されません。 **N** 失敗した住所の書式を整えません (デフォルト)。 **Y** 失敗した住所の書式を整えます。 `OutputRecordType` オプションで指定されたフォーマットを使用して、書式を整えた住所が返されます。`OutputRecordType=E` を指定した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddress が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、`OutputRecordType=I` を指定します。 `Option.OutputRecordType` オプションで指定されたフォーマットを使用して、書式を整えた住所が返されます。`Option.OutputRecordType=E` を指定した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddress が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、`Option.OutputRecordType=I` を指定します。 **[標準住所を含める]**、**[住所行の要素を含める]**、および **[郵便情報を含める]** の各チェック ボックスで指定されたフォーマットを使用して、書式を整えた住所が返されます。**[住所行の要素を含める]** を選択した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddress が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、**[正規化された入力住所要素を含める]** を選択します。 Y を指定する場合は、OutputRecordType に対して "A" と "E" またはそのいずれかを指定する必要があります。

オプション名	説明
	Y を指定する場合は、`Option.OutputRecordType` に対して "A" と "E" またはそのいずれかを指定する必要があります。 このオプションをオンにする場合は、**[標準住所を含める]** と **[住所行の要素を含める]** またはそのいずれかを選択する必要があります。
OutputStreetNameAlias	米国住所に対しては、ストリート名のエイリアスを出力に使用するかどうかを指定します。ストリート名のエイリアスとは、ストリートに対する別名で、通常は、ストリート上の特定の範囲の住所のみに対して使用されます。ストリート名のエイリアスを出力に使用しないと指定した場合は、ストリートにエイリアス名があるかどうかにかかわらず、出力ではストリートの "基本" 名が使用されます。基本名とは、ストリート全体に適用される名称です。 **N** 出力において、ストリート名のエイリアスを返しません。 **Y** ストリート名のエイリアスが存在する場合は、出力において、ストリート名のエイリアスを返します (デフォルト)。

オプション名	説明
OutputStreetNameAlias	米国住所に対しては、入力に使用されているストリート名のエイリアスの処理方法を指定します。ストリート名のエイリアスとは、ストリートに対する別名で、通常は、ストリート上の特定の範囲の住所のみに対して使用されます。 このオプションを有効にすると、入力に使用されているストリート名のエイリアスが、出力にも使用されます。このオプションを有効にしない場合は、入力に使用されているストリート名のエイリアスが、出力では基本ストリート名に変換されますが、次の例外があります。 • 入力において、よく使用されるエイリアスが使用されている場合は、そのエイリアスが必ず出力で使用されます。 • 入力で使用されている変更名のエイリアスは、出力では必ず基本ストリート名に変換されます。 これは、ValidateAddress でストリート名のエイリアスを処理する方法を制御する 3 つのオプションのうちの 1 つです。他の 2 つは `OutputPreferredAlias` と `OutputAbbreviatedAlias` です。 **注：** `OutputAbbreviatedAlias` を有効にした場合は、`OutputStreetNameAlias` を無効にした場合でも、必ず省略形エイリアスが出力に使用されます。 **N** 出力において、ストリート名のエイリアスを返しません。 **Y** 入力されたストリート名がエイリアスである場合は、出力において、ストリート名のエイリアスを返します (デフォルト)。

オプション名	説明
OutputAddressBlocks	実際の郵便物に印字される、書式を整えた住所を返すかどうかを指定します。住所の各行が、別々の住所ブロック フィールドに入れて返されます。**AddressBlock1** から **AddressBlock9** まで、最大 9 つの住所ブロック出力フィールドが使用されます。 例えば、以下の住所入力の場合、 AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 以下の住所ブロックが出力されます。 AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA ValidateAddress は、郵便当局の規格に従って、住所の書式を整えて、住所ブロックの形式にします。国名は、万国郵便連合 (UPU) の国名で返されます。`OutputCountryFormat` オプションは、住所ブロックの国名には影響を与えないことに注意してください。このオプションは、**Country** 出力フィールドに返される名前のみに影響を与えます。 米国およびカナダ以外の住所に対しては、ValidateAddress で住所の妥当性が確認できなかった場合、住所ブロックは返されません。米国およびカナダの住所に対しては、妥当性が確認できなかった場合も住所ブロックが返されます。 **N** 住所ブロックを返しません。こちらがデフォルトです。 **Y** 住所ブロックを返します。

オプション名	説明
OutputAMAS	実際の郵便物に印字される、書式を整えた住所を返すかどうかを指定します。住所の各行が、別々の住所ブロック フィールドに入れて返されます。**AddressBlock1** から **AddressBlock9** まで、最大 9 つの住所ブロック出力フィールドが使用されます。 例えば、以下の住所入力の場合、 AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 以下の住所ブロックが出力されます。 AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA ValidateAddress は、郵便当局の規格に従って、住所の書式を整えて、住所ブロックの形式にします。国名は、万国郵便連合 (UPU) の国名で返されます。`OutputCountryFormat` オプションは、住所ブロックの国名には影響を与えないことに注意してください。このオプションは、**Country** 出力フィールドに返される名前のみに影響を与えます。 米国およびカナダ以外の住所に対しては、ValidateAddress で住所の妥当性が確認できなかった場合、住所ブロックは返されません。米国およびカナダの住所に対しては、妥当性が確認できなかった場合も住所ブロックが返されます。 **N** 住所ブロックを返しません。こちらがデフォルトです。 **Y** 住所ブロックを返します。

下院選挙区の取得

ValidateAddress は、住所に対する米下院選挙区を特定できます。

下院選挙区を取得するには、`OutputRecordType` に P を含める必要があります。`OutputRecordType` の詳細については、出力データ オプション（406ページ）を参照してください。

表 87：下院選挙区出力

フィールド名	説明
USCongressionalDistrict	下院選挙区番号。住所が州以外の住所 (プエルトリコやワシントン D.C. など) である場合は、このフィールドは空白になります。

郡名の取得

ValidateAddress は、ある住所が所在する郡を特定し、その郡名を返すことができます。

注： 郡名は、米国住所にのみ適用されます。

郡名を取得するには、`OutputRecordType` に P を含める必要があります。`OutputRecordType` の詳細については、出力データ オプション（406ページ）を参照してください。

表 88：郡名出力

フィールド名	説明
USCountyName	郡名

FIPS 郡番号の取得

連邦情報処理標準 (FIPS) 郡番号は、州の中の各郡を識別する番号です。これらの番号は、州レベルにおいてのみ一意であり、国レベルでは一意ではないことに注意してください。詳細については、**http://www.census.gov** を参照してください。

注： FIPS 郡番号は、米国住所にのみ適用されます。

FIPS 郡番号を取得するには、`OutputRecordType` に P を含める必要があります。`OutputRecordType` の詳細については、出力データ オプション（406ページ）を参照してください。

表 89 : FIPS 郡番号出力

フィールド名	説明
USFIPSCountyNumber	FIPS (連邦情報処理標準) 郡番号

配達ルート コードの取得

配達ルート コードとは、個々の郵便配達者に割り当てられた一意の識別子で、これによって米国の各配達ルートを一意に識別することができます。ValidateAddress は、宛先の配達ルートを表すコードを返すことができます。

> **注 :** 配達ルート コードは、米国住所にのみ適用されます。

配達ルート コードを取得するには、`OutputRecordType` に P を含める必要があります。`OutputRecordType` の詳細については、出力データ オプション（406ページ）を参照してください。

表 90 : 配達ルート コード出力

フィールド名	説明
USCarrierRouteCode	配達ルート コード

配達ポイント バーコードの作成

配達ポイント バーコード (DPBC) は、住所を POSTNET™ バーコードで表記したものです。開始および終了フレーム バーと、ZIP + 4® Code、ストリートの住所の番号に基づいて計算された値、および修正ディジットの 1 桁ごとに 5 本のバーがあり、合計 62 本のバーで構成されます。DPBC により、配達業者の徒歩経路のレベルにまで手紙を自動仕分けすることができます。ValidateAddress は、DPBC の作成に必要なデータを生成します。

> **注 :** 配達ポイント バーコードは、米国住所にのみ適用されます。配達ポイント バーコードの詳細については、**http://www.usps.com** を参照してください。

DPBC の作成に必要なデータを生成するには、`OutputRecordType` に P を含める必要があります。`OutputRecordType` の詳細については、出力データ オプション（406ページ）を参照してください。

表 91 : 配達ポイント バーコード出力

フィールド名	説明
PostalBarCode	配達ポイント バーコードの配達ポイント部分
USBCCheckDigit	11 桁の配達ポイント バーコードのチェック ディジット部分

DPBC を作成するには、ValidateAddress 出力列の値を次のように結合します。

PostalCode.Base + PostalCode.Addon + PostalBarcode + USBCCheckDigit

例えば、次のデータがあるとします。

- **PostalCode.Base** = 49423
- **PostalCode.Addon** = 4506
- **PostalBarcode** = 29
- **USBCCheckDigit** = 2

このデータから作成されるバーコードは、次のようになります。

494234506292

デフォルト オプション

以下の表に、住所のフォーマットと処理を制御するオプションの一覧を示します。これらのオプションはデフォルトですべての住所に適用されるので、"デフォルト オプション" と言います。これらのオプションのうちの一部は、カナダ住所に対してオーバーライドすることができます。詳細については、**カナダ住所のオプション**（439ページ）を参照してください。

表 92 : デフォルト オプション

optionName	説明
OutputCasing	出力住所の大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力に大文字と小文字を混在させます (デフォルト)。例: 123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。例: 123 MAIN ST MYTOWN FL 12345
OutputPostalCodeSeparator	ZIP™ Code またはカナダの郵便番号に区切り文字 (スペースまたはハイフン) を使用するかどうかを指定します。 例えば、区切り文字ありの ZIP + 4® Code は 20706-1844、区切り文字なしは 207061844 になります。区切り文字ありのカナダの郵便番号は P5E"1S7、区切り文字なしは P5E1S7 になります。 **Y** 区切り文字を使用します (デフォルト)。 **N** 区切り文字を使用しません。 **注 :** カナダの郵便番号ではスペースが、米国のZIP + 4® コードではハイフンが使用されます。
OutputMultinationalCharacters	ウムラウト記号やアクセント記号などの付加記号を含む多国籍文字を返すかどうかを指定します(米国住所ではサポートされません)。 **N** 出力に多国籍文字を使用しません (デフォルト)。標準の ASCII 文字のみが返されます。 **Y** 出力に多国籍文字を使用します。
KeepMultimatch	一致する可能性のある住所を複数持つ入力住所に対して複数の住所を返すかどうかを示します。 **Y** 複数のマッチを返します (デフォルト)。 **N** 複数のマッチを返しません。 詳細については、**複数マッチを返す**（423ページ）を参照してください。

optionName	説明
StandardAddressFormat	米国住所の補助的な住所情報を住所をサポートしていません。補助的な住所情報とは、部屋番号やアパート番号などの指定子のことです。例えば、次の住所の補助的な住所情報は "Apt 10E" で、主要な住所情報は "424 Washington Blvd" です。 Apt 10E 424 Washington Blvd Springfield MI 49423 **C** 主要な住所情報と補助的な住所情報の両方を AddressLine1に配置します (デフォルト)。 **S** 主要な住所情報を AddressLine1に、補助的な住所情報を AddressLine2 に配置します。 **D** 主要な住所情報と補助的な住所情報の両方を AddressLine1に配置し、二重住所からドロップされた情報を AddressLine2 に配置します。二重住所とは、ストリート情報と、PO Box/地方配送路/Highway Contract 情報の両方を含む住所のことです。詳細については、**二重住所ロジックについて**（422ページ）を参照してください。
OutputShortCityName	短い都市名または非正式な都市名を代替名として持つ都市名を書式設定する方法を指定します。米国およびカナダの住所のみに適用されます。 **Y** USPS® が承認した都市の略称が 1 つある場合、それを返します。USPS® は、14 文字以上の都市名に対し、略称を定めています。都市の略称は 13 文字以下で、宛名ラベルのサイズが限られている場合に使用できます。短い都市名が存在しない都市に対しては、正式な都市名が返されます。 **N** 長い都市名を返します (デフォルト)。 **S** 入力住所に省略された都市名が使用されている場合にのみ、省略された都市名を返します。入力住所に短い都市名が使用されていない場合は、その都市に対する USPS® の規則によって、長い都市名または短い都市名が返されます。CASS™ テストを実行する場合は、このオプションを選択します。 **V** 入力都市名が非正式な都市名の場合、非正式な都市名 (非正式名) を出力します。例えば、"Hollywood" は "Los Angeles" の非正式な都市名です。このオプションを選択せず、入力都市名が非正式な都市名の場合、長いバージョンの郵送都市名が返されます。

optionName	説明
OutputCountryFormat	**Country** 出力フィールドに返される国名に使用するフォーマットを指定します。例えば、英語を選択した場合、"Deutschland" という国名は "Germany" として返されます。 **E** 英語の国名を使用します (デフォルト)。 **S** スペイン語の国名を使用します。 **F** フランス語の国名を使用します。 **G** ドイツ語の国名を使用します。 **I** 国名の代わりにその国の 2 文字の ISO の略語を使用します。 **U** 国名の代わりにその国の万国郵便連合 (Universal Postal Union: UPU) の略語を使用します。

optionName	説明
HomeCountry	デフォルト国を指定します。大部分の住所が存在する国を指定してください。例えば、処理する住所の大部分がカナダにある場合は、カナダを指定します。ValidateAddress は、[StateProvince]、[PostalCode]、および [Country] の各住所フィールドから国を特定できなかった場合、指定された国を使用して、住所の検証を試みます。有効な国名には次のものがあります。 Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent and the Grenadines, Samoa, San Marino, Sao Tome and Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe

optionName	説明
DualAddressLogic	複数の空白ではない住所行があるか、複数の住所タイプが同じ住所行に設定されている場合にどのタイプの一致を返すかを指定します。(米国住所のみ)。 **N** (デフォルト) USPS®CASS™ の規則では、以下の優先順位に基づいて、返す住所を決定します。 1. PO Box 2. Firm 3. Highrise 4. ストリート 5. Rural Route 6. General Delivery **S** 住所行に関係なく、ストリート一致を返します。 **P** 住所行に関係なく、PO Box 一致を返します。 詳細については、二重住所ロジックについて（422ページ）を参照してください。

二重住所ロジックについて

米国住所に対しては、`DualAddressLogic` オプションは、ストリート情報と PO Box/地方配送路/Highway Contract 情報の両方が住所に含まれる場合に、Validate Address がストリート一致を返すか、または PO Box/地方配送路/Highway Contract 一致を返すかを制御します。

> **注：** ストリート情報が PO Box/地方配送路/Highway Contract 情報と別の住所行入力フィールドに含まれている場合、`DualAddressLogic` オプションは、効果を持ちません。

例えば、次の入力住所が与えられたとします。

AddressLine1: 401 N Main St Apt 1 POB 1
City: Kemp
StateProvince: TX
PostalCode: 75143

ValidateAddress は、次のいずれかを返します。

- `DualAddressLogic` が N または P のいずれかに設定されている場合は、次を返します。

  AddressLine1: PO Box 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-0001

- `DualAddressLogic` が S に設定されている場合は、次を返します。

  AddressLine1: 401 N Main St Apt 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-4806

住所の正規化に使用されない住所データは、次の 2 カ所のいずれかに返すことができます。

- **AddressLine2 — StandardAddressFormat=D** を指定した場合、住所の正規化に使用されない住所情報は `AddressLine2` フィールドに返されます。詳細については、デフォルト オプション（417ページ）を参照してください。例えば、二重住所に対してストリート一致を返すと選択すると、次のようになります。

  AddressLine1: 401 N Main St Apt 1
  AddressLine2: PO Box 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-0001

- **AdditionalInputData —** `StandardAddressFormat=D` を指定しない場合、住所の正規化に使用されない住所情報は **AdditionalInputData** フィールドに返されます。このオプションの詳細については、デフォルト オプション（417ページ）を参照してください。例えば、二重住所に対してストリート一致を返すと選択すると、次のようになります。

  AddressLine1: 401 N Main St Apt 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-0001
  AdditionalInputData: PO Box 1

ドロップされた住所情報を取得するには、`StandardAddressFormat` オプションを D に設定します。詳細については、デフォルト オプション（417ページ）を参照してください。

複数マッチを返す

ValidateAddress が、入力住所に一致する可能性のある複数の住所を郵便データベース内で検出した場合、一致する可能性のある住所を ValidateAddress が返すよう設定できます。例えば、次の住所は米国郵便データベース内の複数の住所にマッチします。

PO BOX 1
New York, NY

オプション

複数マッチを返すには、次の表に示すオプションを使用します。

表 93 : 複数マッチのオプション

オプション名	説明
KeepMultimatch	一致する可能性のある住所を複数持つ入力住所に対して複数の住所を返すかどうかを示します。 **Y** 複数のマッチを返します (デフォルト)。 **N** 複数のマッチを返しません。
MaximumResults	返す住所の最大数を示す 1 ～ 10 の数字を入力します。 デフォルト値は 1 です。 **注 :** Keepmultimatch=N と KeepMultimatch=Y/MaximumResults=1 の違いは、KeepMultimatch=N は複数マッチによって失敗が返され、KeepMultimatch=Y かつ MaximumResults=1 は複数マッチによって 1 つのレコードが返される点です。
OutputFieldLevelReturnCodes	どの出力住所が候補住所かを特定するには、OutputFieldLevelReturnCodes に対して値 **Y** を指定する必要があります。このように設定すると、候補住所のレコードのフィールドレベルの結果インジケータに 1 つ以上の値 "M" が格納されます。

出力

複数マッチを返すよう選択した場合、住所は指定した住所フォーマットで返されます。住所フォーマットの指定については、**出力データ オプション**（406ページ）を参照してください。どのレコードが候補住所であるかを特定するには、フィールドレベルの結果インジケータに複数の値 "M" があるかどうかを調べます。詳細については、**フィールドレベルの結果インジケータ**（464ページ）を参照してください。

米国住所のオプション

オプション名	説明
PerformUSProcessing	米国住所を処理するかどうかを指定します。米国住所処理を有効にすると、ValidateAddress は米国住所の検証を試みます。米国住所処理を無効にすると、米国住所は失敗します。つまり、Status 出力フィールドに "F" が設定されて返されます。出力フィールド Status.Code は、"DisabledCoder" となります。米国住所処理のライセンスを取得していない場合は、ジョブに米国住所が含まれるか否かにかかわらず、米国住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注：** 米国住所を正常に処理するには、米国住所処理の有効なライセンスを取得する必要があります。米国住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、米国住所処理を有効にすると、エラーとなります。 **N** いいえ、米国住所を処理しません。 **Y** はい、米国のアドレスを処理します。こちらがデフォルトです。
Database.US	米国の検証に使用するデータベースを指定します。Management Console の [米国データベース リソース] パネルで定義されたデータベースのみが使用可能です。
PerformLOT	Enhanced Line of Travel (eLOT) 処理は、トラベルライン シーケンス コードを住所に割り当てます。住所は eLOT 順でソートされませんが、住所を eLOT 順にソートするために使えるトラベルライン シーケンス コードが提供されます。 eLOT 処理を実行するには、eLOT データベースをインストールしておく必要があります。 **N** トラベルライン処理を実行しません。こちらがデフォルトです。 **Y** トラベルライン処理を実行します。 このオプションで返される出力フィールドの一覧については、**Enhanced Line of Travel 出力**（482ページ） を参照してください。

オプション名	説明
PerformRDI	Residential Delivery Indicator (RDI™) 処理は、住所が (企業住所ではなく) 個人住所であることを調べます。RDI™ 処理を実行するには、RDI™ データベースをインストールしておく必要があります。 DPV® と RDI™ の両方の処理を有効にしている場合は、RDI™ 情報は、住所が有効な配達ポイントである場合のみ返されます。DPV® で住所の妥当性が確認されなかった場合は、RDI™ データは返されません。 **N** Residential Delivery Indicator 処理を実行しません。こちらがデフォルトです。 **Y** Residential Delivery Indicator 処理を実行します。
PerformESM	Enhanced Street Matching (ESM) は、追加のマッチング ロジックを適用することにより、綴りに誤りがあるストリート名や複雑なストリート名を修正し、マッチ結果を得ます。ESM を使うと、より多くの住所の妥当性を確認できるようになりますが、パフォーマンスは低下します。ASM が有効な場合は、ESM を実行することはできません。 **N** Enhanced Street Matching を実行しません。こちらがデフォルトです。 **Y** Enhanced Street Matching を実行します。
PerformASM	All Street Matching (ASM) は、ESM 処理に加えて追加のマッチング ロジックを適用することにより、ストリート名の誤りを修正し、マッチ結果を得ます。ストリートの最初の文字が誤っている場合のストリートのマッチングに有効です。ASM は、最良の住所検証結果を提供しますが、パフォーマンスは低下します。 **N** All Street Matching を実行しません。 **Y** All Street Matching を実行します。こちらがデフォルトです。

オプション名	説明
PerformDPV	Delivery Point Validation (DPV®) は、特定の住所が有効な住所の範囲内にあるかどうかを確認するのではなく、特定の住所が存在するかどうかを確認します。CMRA 処理は、住所が、Commercial Mail Receiving Agency (CMRA: 民間私書箱) と呼ばれる民間企業が貸し出す私書箱であるかどうかを確認します。 DPV および CMRA 処理を実行するには、DPV データベースをインストールしておく必要があります。DPV データベースには、DPV と CMRA の両方のデータが含まれます。 **N** Delivery Point Validation または CMRA 処理を実行しません。こちらがデフォルトです。 **Y** Delivery Point Validation または CMRA 処理を実行します。 このオプションで返される出力フィールドの一覧については、**DPV および CMRA 出力**（485ページ） を参照してください。
PerformLACSLink	USPS® Locatable Address Conversion System (LACS) は、地方配送路の住所をストリート名に沿った住所に変換した場合、PO Box 番号の再割り当てがあった場合、またはストリート名に沿った住所が変更した場合に、それに伴って変更した住所を修正します。LACS^{Link} 処理を有効にした場合、妥当性が確認できなかった住所、または妥当性が確認され、LACS^{Link} 変換のフラグが付けられた住所に対し、その処理が実行されます。 LACS^{Link} 処理を実行するには、LACS^{Link} データベースをインストールしておく必要があります。 **N** LACS^{Link} 変換を行いません。こちらがデフォルトです。 **Y** LACS^{Link} 変換を行います。 このオプションで返される出力フィールドの一覧については、**LACSLink 出力**（483ページ）を参照してください。

オプション名	説明
PerformEWS	Early Warning System (EWS) は、USPS® EWS ファイルを使用して、ZIP + 4® データベースには存在しない住所の妥当性を確認します。 EWS 処理を実行するには、EWS データベースをインストールしておく必要があります。 入力住所が、EWS ファイルの住所に一致する場合、次のレコード レベルの結果インジケータが返されます。 • Status="F" • Status.Code="EWSFailure" • Status.Description="Address found in EWS table" **N** EWS 処理を実行しません。こちらがデフォルトです。 **Y** EWS 処理を実行します。

オプション名	説明
ExtractFirm	

オプション名	説明

AddressLine1 ～ AddressLine4 から企業名を抽出し、FirmName 出力フィールドに入れるかどうかを指定します。このオプションは、入力レコードの FirmName フィールドが空白で、住所行が複数存在する場合に適用されます。

Y 企業名を抽出します。

N 企業名を抽出しません。こちらがデフォルトです。

住所行の中の企業名を特定するため、住所行をスキャンし、どのフィールドが住所行で、どのフィールドが企業名の行であるかを特定するためのキーワードおよびパターンが検索されます。この処理はパターンに基づいて行われるため、フィールドが誤って認識される場合があります。最適な企業名抽出を行うためのヒントを、以下に示します。

- 可能ならば、主要な住所要素を AddressLine1、補助的な要素を AddressLine2、都市化名を AddressLine3、企業名を AddressLine4 に配置します。住所に都市化コードが存在しない場合は、企業名を AddressLine3 に配置し、AddressLine4 を空白にします。例を次に示します。

  AddressLine1: 4200 Parliament Place
  AddressLine2: Suite 600
  AddressLine3: Pitney Bowes
  AddressLine4: <空白>

- 住所行を 2 行だけ定義する場合は、ほとんどの場合 AddressLine2 には補助的な住所が入ります。AddressLine2 を企業名として処理する確率を上げるには、企業名を AddressLine3 に配置し、AddressLine2 は空白にします。
- 企業名に数字が含まれていると ("1 Stop Software" の "1" など)、そのフィールドが住所行として扱われる可能性が高くなります。

以下に、企業名抽出の例をいくつか示します。

- 次の例では、AddressLine2 が FirmName 出力フィールドに抽出されます。

  FirmName: <空白>
  AddressLine1: 4200 Parliament Place Suite 600
  AddressLine2: International Goose Feathers inc.

- 次の例では、AddressLine3 が FirmName 出力フィールドに抽出されます。

  FirmName: <空白>
  AddressLine1: 4200 Parliament Place
  AddressLine2: Suite 600
  AddressLine3: Pitney Bowes

- 次の例では、AddressLine3 は AdditionalInputData 出力フィールドに配置されます。FirmName 入力フィールドが空白ではないため、企業名は抽出されません。

  FirmName: International Goose Feathers Inc.
  AddressLine1: 4200 Parliament Place
  AddressLine2: Suite 600

オプション名	説明
	AddressLine3: Pitney Bowes • 次の例では、空白でない住所行が 1 行しかなく、その行は必ず主要な住所要素として処理されるため、企業名は抽出されません。 **FirmName:** <空白> **AddressLine1:** 4200 Parliament Place Suite 600 • 次の例では、AddressLine2 は補助的な住所要素として処理されます。数字の "1" が含まれているために、そのフィールドが補助的な住所要素として扱われるためです。 **FirmName:** <空白> **AddressLine1:** 4200 Parliament Place Suite 600 **AddressLine2:** 1 Stop Software
ExtractUrb	AddressLine1 ～ AddressLine4 から都市化名を抽出し、USUrbanName 出力フィールドに入れるかどうかを指定します。このオプションは、入力レコードの USUrbanName フィールドが空白で、住所行が複数存在する場合に適用されます。 **Y** 都市化名を抽出します。 **N** 都市化名を抽出しません。こちらがデフォルトです。 住所行の中の都市化名を特定するため、住所行をスキャンし、どのフィールドが住所行で、どのフィールドが都市化名の行であるかを特定するためのキーワードおよびパターンが検索されます。この処理はパターンに基づいて行われるため、フィールドを誤って認識する場合があります。最適な都市化名抽出を行うには、できる限り、主要な住所要素を AddressLine1、補助的な要素を AddressLine2、都市化名を AddressLine3、企業名を AddressLine4 に配置します。例を次に示します。 **AddressLine1:** A1 Calle A **AddressLine2:** **AddressLine3:** URB Alamar **AddressLine4:** Pitney Bowes

オプション名	説明
PerformSuiteLink	Suite$^{Link™}$ 処理を実行するかどうかを指定します。 Suite^{Link} は、補助的な住所情報の妥当性が確認できなかった米国の企業住所に対し、その補助的な住所情報を修正します。Suite^{Link} 処理が有効になっている場合、既知の企業名とその補助的な住所情報からなるデータベースに対して、企業名のマッチングが行われます。 例を次に示します。 企業名: Pitney Bowes 住所行 1: 4200 Parliament Place Address Line 2: STE 1 郵便番号: 20706 この場合、Suite^{Link} は、Suite 番号を以下の正しい Suite番号に変更します。 企業名: Pitney Bowes 住所行 1: 4200 Parliament Place Address Line 2: **STE 600** Postal Code: 20706-1844 Suite$^{Link™}$ 処理を実行するには、Suite$^{Link™}$ データベースをインストールしておく必要があります。 このオプションは、次の値のいずれかを取ります。 **N** Suite$^{Link™}$ を使用しません。こちらがデフォルトです。 **Y** Suite$^{Link™}$ を使用します。 このオプションで返されるフィールドの一覧については、**SuiteLink出力**（487ページ） を参照してください。

オプション名	説明
OutputPreferredAlias	ストリート名に対してよく使用されるエイリアスを出力に使用するかどうかを指定します。 米国におけるストリート名のエイリアスとは、ストリートの一部に付けられた別名のことです。ストリート名のエイリアスには、次の 4 種類があります。 • **よく使用される名前 —** その地域でよく使用されるストリート名です。通常は、ストリート上の特定の範囲の住所のみに対して使用されます。 • **省略形 —** ストリート名の省略形です。AddressLine1 の長さが 31 文字以上になる場合に使用することができます。例えば、1234 BERKSHIRE VALLEY RD APT 312A というストリート名は、1234 BERKSHIRE VLLY RD APT 312A と省略することができます。 • **変更名 —** ストリート名が正式に変更された場合に、新しい名前を表すエイリアスです。例えば、SHINGLE BROOK RD というストリート名が CANNING DR に変更された場合、CANNING DR が変更済みのエイリアス タイプとなります。 • **その他の名前 —** このストリート名エイリアスには、ストリートの他の名前や、ストリートの一般的な省略形などがあります。 エイリアスではないストリート名のことを、基本ストリート名と呼びます。 入力において、よく使用されるエイリアスが使用されている場合は、このオプションを選択しているかどうかにかかわらず、そのエイリアスが出力のストリート名になります。 これは、ValidateAddress でストリート名のエイリアスを処理する方法を制御する 3 つのオプションのうちの 1 つです。他の 2 つは OutputStreetNameAlias と OutputAbbreviatedAlias です。 多くの場合、OutputPreferredAlias と OutputAbbreviatedAlias の両方が選択されており、ValidateAddress が、郵便データベース内でよく使用されるエイリアスと省略形エイリアスの両方を検出した場合は、省略形エイリアスが出力に使用されます。入力のストリート名がよく使用されるエイリアスである場合は、例外になります。この場合は、よく使用されるエイリアスが出力に使用されます。 **Y** ストリート名に対してよく使用されるエイリアスの処理を実行します。 **N** ストリート名に対してよく使用されるエイリアスの処理を実行しません。こちらがデフォルトです。 **注 :** 入力住所に、"変更名"であるストリート名のエイリアスが含まれている場合は、指定したオプションにかかわらず、出力住所には必ず、基本ストリート名が使用されます。

オプション名	説明
OutputAbbreviatedAlias	出力住所行の長さが 31 文字以上になる場合に、ストリート名に対する省略形エイリアスを出力に使用するかどうかを指定します。 これは、ValidateAddress でストリート名のエイリアスを処理する方法を制御する 3 つのオプションのうちの 1 つです。他の 2 つは OutputStreetNameAlias と OutputPreferredAlias です。 **注 :** 入力において、よく使用されるエイリアスが指定されている場合は、ストリート名に対する省略形エイリアスの処理を有効にしていても、出力のストリート名は必ず、よく使用されるエイリアスになります。 **Y** 省略形エイリアスの処理を実行します。 **N** 省略形エイリアスの処理を実行しません。こちらがデフォルトです。 **注 :** 入力住所に、"変更名"であるストリート名のエイリアスが含まれている場合は、指定したオプションにかかわらず、出力住所には必ず、基本ストリート名が使用されます。
DPVDetermineNoStat	住所の "no stat" ステータスを調べます。住所が存在するが、郵便物を受け取れない場合、その住所は "no stat" とみなされるため、配達ルートに関する配達統計としてカウントされません (そのため "no stat" という用語が使用されます)。例としては、建設中の建物や、郵便物を受け取る可能性が低いと郵便配達業者が識別した建物などがあります。 **N** "no stat" ステータスを調べません。こちらがデフォルトです。 **Y** "no stat" ステータスを調べます。 **注 :** このオプションを使用するには DPV 処理を有効にする必要があります。 結果は DPVNoStat フィールドに返されます。詳細については、**LACSLink 出力**（483ページ） を参照してください。
DPVDetermineVacancy	そのロケーションがすくなとも 90 日間使用されていないかどうかを調べます。 **N** 空家かどうかを調べません。こちらがデフォルトです。 **Y** 空家かどうかを調べます。 **注 :** このオプションを使用するには DPV 処理を有効にする必要があります。 結果は DPVVacant フィールドに返されます。詳細については、**LACSLink 出力**（483ページ） を参照してください。

オプション名	説明
ReturnVerimove	出力に VeriMove 詳細データを返します。 **N** VeriMove 詳細データを返しません。こちらがデフォルトです。 **Y** VeriMove 詳細データを返します。
SuppressZplusPhantomCarrierR777	キャリアルート R777 の住所を抑制するかどうかを指定します。これらの住所は疑似ルートであり、ストリート配達に使用できません。これらの住所には USPS® による ZIP + 4® コードが割り当てられているため、Validate Address はこれらの住所を配達可能と判定します。キャリアルート R777 の住所を配達可能と判定したくない場合は、このオプションを選択します。その場合は、次のように動作します。 • ZIP + 4 コードは割り当てられません • 住所は USPS Form 3553 (CASS Summary Report) から除外されます • DPV 補足コードとして R7 が返されます **N** キャリアルート R777 の住所を抑制しません。 **Y** キャリアルート R777 の住所を抑制します。
StreetMatchingStrictness	入力住所が郵便データベース内の住所にマッチするかどうかを調べる際に使用するアルゴリズムを指定します。次のいずれかです。 **E** 入力されたストリート名は、データベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です (デフォルト)。 **L** マッチング アルゴリズムは "あいまい" です。
FirmMatchingStrictness	入力住所が郵便データベース内の住所にマッチするかどうかを調べる際に使用するアルゴリズムを指定します。次のいずれかです。 **E** 入力された企業名は、データベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です (デフォルト)。 **L** マッチング アルゴリズムは "あいまい" です。

オプション名	説明
DirectionalMatchingStrictness	入力住所が郵便データベース内の住所にマッチするかどうかを調べる際に使用するアルゴリズムを指定します。次のいずれかです。 **E** 123 N Main St. における "N" など、入力された道順がデータベースに完全に一致する必要があります。 **T** マッチング アルゴリズムは "厳格" です。 **M** マッチング アルゴリズムは "中" です。こちらがデフォルトです。 **L** マッチング アルゴリズムは "あいまい" です。
DPVSuccessfulStatusCondition	DPV 結果がレコードの失敗の原因とならない一致条件を選択します。 **F** 全体一致 **P** 部分一致 **A** 常に一致こちらがデフォルトです。 **注 :** このオプションを使用するには DPV 処理を有効にする必要があります。
FailOnCMRAMatch	民間私書箱 （CMRA） との一致をマッチとみなしませんか。 **N** いいえ、CMRA との一致をマッチとみなします。こちらがデフォルトです。 **Y** はい、CMRA との一致をマッチとみなしません。 **注 :** このオプションを使用するには DPV 処理を有効にする必要があります。
StandardAddressPMBLine	私書箱 (PMB) の情報をどこに配置するかを指定します。 **N** なし標準住所出力に PMB 情報を含めません (デフォルト)。 **1** PMB 情報を AddressLine1 に配置します。1 を指定した場合、StandardAddressFormat に C または D をセットする必要があります。 **2** PMB 情報を AddressLine2 に配置します。

オプション名	説明
PreferredCity	優先する最終行都市名を格納するかどうかを指定します。 **Z** USPS ZIP+4 ファイルからの Preferred Last Line City Name を格納します (都市名を上書き)。 **注：** このオプションを選択すると、Validate Address は CASS 認定の設定と USPS 3553 レポートを生成します。 **C** USPS City/State ファイルからの USPS-preferred City Name を格納します **注：** このオプションを選択すると、Validate Address は CASS 認定の設定と USPS 3553 レポートを生成しません。 **P** USPS City/State ファイルからの Primary City Name を格納します **注：** このオプションを選択すると、Validate Address は CASS 認定の設定と USPS 3553 レポートを生成しません。

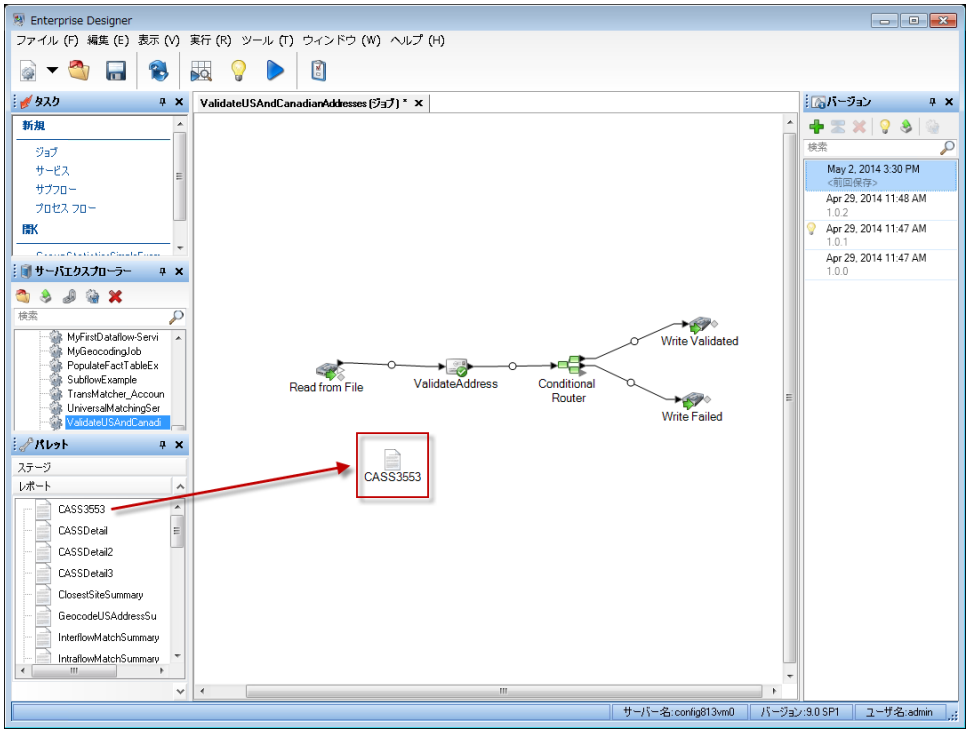
CASS 認定処理

また、CASS 認定™処理では USPS CASS 詳細レポートも生成されます。このレポートに含まれる情報は 3553 レポートと同じものですが、DPV、LACS、および SuiteLink に関する大幅に詳しい統計情報が含まれます。USPS CASS 詳細レポートは、郵便料金の値引きを受けるために必ずしも必要ではなく、郵便物と一緒に提出する必要はありません。

1. Validate Address を CASS 認定™モードにする必要があります。ウィンドウの一番上に**(CASS 認定でないもの)** と表示される場合は、**[CASS 有効]** ボタンをクリックしてください。**[CASS ルールを強制]** チェック ボックスが表示されます。
2. **[CASS 3553 を設定]** をクリックします。**[CASS レポート フィールド]** ダイアログ ボックスが表示されます。
3. **[リスト処理元]** の会社名、**[リスト名または ID 番号]**、およびこのジョブで処理する **[リスト数]** を入力します。
4. **[差出人名]**、**[住所]**、および**[都市]**、**[州]**、**[ZIP]** を入力します。
5. **[OK]** をクリックします。

   生成された USPS® CASS Form 3553 のセクション B にリストの情報が、セクション D に差出人の情報が表示されます。

6. Enterprise Designer で、**CASS3553** レポートをレポートのパレットからキャンバスにドラッグします。



7. キャンバスの **[CASS3553]** アイコンをダブルクリックします。
8. **[ステージ]** タブで、**[Validate Address]** チェックボックスをオンにします。Validate Address ステージを何か別の名前に変更している場合は、住所検証ステージに指定した名前のチェック ボックスをオンにする必要があります。
9. **[パラメータ]** タブで、レポートのフォーマットを選択します。PDF、 HTML、またはプレーン テキストのフォーマットでレポートを作成できます。
10. **[OK]** をクリックします。
11. CASS 詳細レポートを生成する場合は、**CASSDetail** に対して手順 6 ～ 10 を繰り返します。

カナダ住所のオプション

optionName	説明
PerformCanadianProcessing	カナダ住所を処理するかどうかを指定します。カナダ住所処理を有効にした場合、ValidateAddress はカナダ住所の検証を試みます。カナダ住所処理を無効にすると、Status 出力フィールドに"F" が設定されて返され、カナダ住所処理は失敗します。その際、出力フィールド Status.Code は、"DisabledCoder" となります。カナダ住所処理のライセンスを取得していない場合は、ジョブにカナダ住所が含まれるか否かにかかわらず、カナダ住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注:** カナダ住所を正常に処理するには、カナダ住所処理の有効なライセンスを取得する必要があります。カナダ住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、カナダ住所処理を有効にすると、エラーとなります。 **N** カナダ住所を処理しません。 **Y** カナダ住所を処理します (デフォルト)。
Database.Canada	カナダ住所の検証に使用するデータベースを指定します。カナダ住所検証用のデータベースを指定するには、**[データベース]** ドロップダウン リストからデータベースを選択します。Management Console の **[カナダ データベース リソース]** パネルで定義されたデータベースのみが使用可能です。

optionName	説明
CanFrenchFormat	住所及び方向指示の書式設定に使用する言語 (英語またはフランス語) の判断方法を指定します。以下に、英語およびフランス語で書式設定された住所の例を示します。 英語: 123 Main St W フランス語: 123 Rue Main O このパラメータは、住所の書式設定を制御します。また、方向指示の綴りにも影響を与えますが、接尾語の綴りには影響を与えません。 **C** マッチング処理によって返されるストリート接尾語によって、言語を判断します。マッチング処理によって返されるストリート接尾語は、ValidateAddress が処理において内部で使用するもので、入力住所のものとは異なる場合があります。あいまいなレコードは、入力と同様に書式設定されます。こちらがデフォルトです。ケベック州の住所はすべて、フランス語で書式設定されます。 **S** カナダ データベースによって、言語を判断します。カナダ データベースには、Canada Post Corporation (CPC: カナダ郵政公社) からのデータが含まれています。ケベック州の住所はすべて、フランス語で書式設定されます。 **T** CanLanguage 入力フィールドによって、言語を判断します。このフィールドに空白以外の値が設定されている場合は、住所はフランス語で書式設定されます。
CanEnglishApartmentLabel	英語の住所に対し、入力住所にアパートメント ラベルが存在しない場合に、出力に使用するデフォルト アパートメント ラベルを指定します。`CanStandardAddressFormat=F` と指定した場合、この設定は無視されます。 **Apt** ラベルとして "Apt" を使用します。こちらがデフォルトです。 **Apartment** ラベルとして "Apartment" を使用します。 **Suite** ラベルとして "Suite" を使用します。 **Unit** ラベルとして "Unit" を使用します。

optionName	説明
CanFrenchApartmentLabel	フランス語の住所に対し、入力住所にアパートメント ラベルが存在しない場合に、出力に使用するデフォルト アパートメント ラベルを指定します。`CanStandardAddressFormat=F` と指定した場合、この設定は無視されます。 **App** "App" をラベルとして使用します。こちらがデフォルトです。 **Appartement** Use "Appartement" as the label. **Bureau** Use "Bureau" as the label. **Suite** Use "Suite" as the label. **Unite** Use "Unite" as the label.
ForceCorrectionLVR	正式情報やスイート情報を変更して、Large Volume Receiver (LVR) または Single-Single レコードをマッチさせます (その郵便番号/ストリート名/ストリート タイプに対して 1 つしかレコードがない場合に使用します)。 **N** LVR または Single-Single レコードをマッチさせるために正式情報やスイート情報を変更しません。LVR レコードは、有効だが修正不可能なレコード (VN) としてマーク付けされます。Single-Single レコードは可能ならば修正されます。または、修正不可能なレコードとして処理されます。 **Y** LVR または Single-Single レコードをマッチさせるために正式情報やスイート情報を変更します。 **注：** このチェック ボックスをオンにする場合は、SERP 認定の設定 **ではない** ため、Statement of Address Accuracy は印刷されません。

optionName	説明
CanPreferHouseNum	家番号と郵便番号がともに有効であるが、競合する場合、`CanPreferHouseNum=Y`と指定することによって、家番号に合わせて郵便番号を強制的に修正できます。このオプションを選択しない場合、郵便番号に合わせて家番号が変更されます。 **N** 郵便番号に合わせて家番号を変更します。こちらがデフォルトです。 **Y** 家番号に合わせて郵便番号を変更します。
CanOutputCityAlias	入力住所に都市名のエイリアスがある場合、そのエイリアスを返すかどうかを指定します。`CanOutputCityFormat=D`を指定している場合は、このオプションは無効です。 **Y** 入力に都市名のエイリアスがある場合は、都市名のエイリアスを出力します。こちらがデフォルトです。 **N** 入力に都市名のエイリアスがあっても、都市名のエイリアスを出力しません。
CanNonCivicFormat	出力において、正式住所ではないキーワードを短縮するかどうかを指定します。例えば、Post Office Box とPO Box のどちらを使用するかが決まります。 **A** 正式住所ではないキーワードを短縮します。こちらがデフォルトです。 **F** 正式住所ではないキーワードを短縮しません。正式なキーワードを使用します。
EnableSERP	SERP オプションを使用するかどうかを指定します。 **Y** SERP オプションを有効にします。 **N** SERP オプションを有効にしません。こちらがデフォルトです。

optionName	説明
CanStandardAddressFormat	出力住所における、補助的な住所情報の配置場所を指定します。補助的な住所情報とは、部屋番号やアパート番号などの指定子のことです。 **D** アパート情報を、StandardAddressFormat オプションで指定された場所に配置します。こちらがデフォルトです。 **B** アパート情報を、AddressLine1 フィールドの末尾 (最後) に配置します。 **F** アパート番号のみ (ラベルは除く) を、AddressLine1 フィールドの先頭に配置します。例えば、400-123 Rue Main とします。 **E** アパート番号とラベルを、AddressLine1 フィールドの先頭に配置します。例えば、Apt 400 123 Rue Main とします。 **S** アパート情報を別の行に配置します。 **S** アパート情報を入力住所と同じ場所に配置します。
CanOutputCityFormat	都市の名前が長い場合に、long、medium、もしくは short のどの都市名を使用するかを指定します。例を次に示します。 Long: BUFFALO HEAD PRAIRIE Medium: BUFFALO-HEAD-PR Short: BUFFALO-HD-PR **D** `OutputShortCityName` オプションで指定されたデフォルト オプションを使用します。こちらがデフォルトです。`OutputShortCityName=V` を指定した場合、都市は、このオプションで `L` を選択し (以下を参照)、`CanOutputCityAlias` で **Y** を選択した場合と同じように書式設定されます。 **S** 短い都市名を出力します。 **L** 長い都市名を出力します。 **M** 中間の長さの都市名を出力します。 **I** 入力住所と同じ都市フォーマットを使用します。出力は、L、M、または S です。

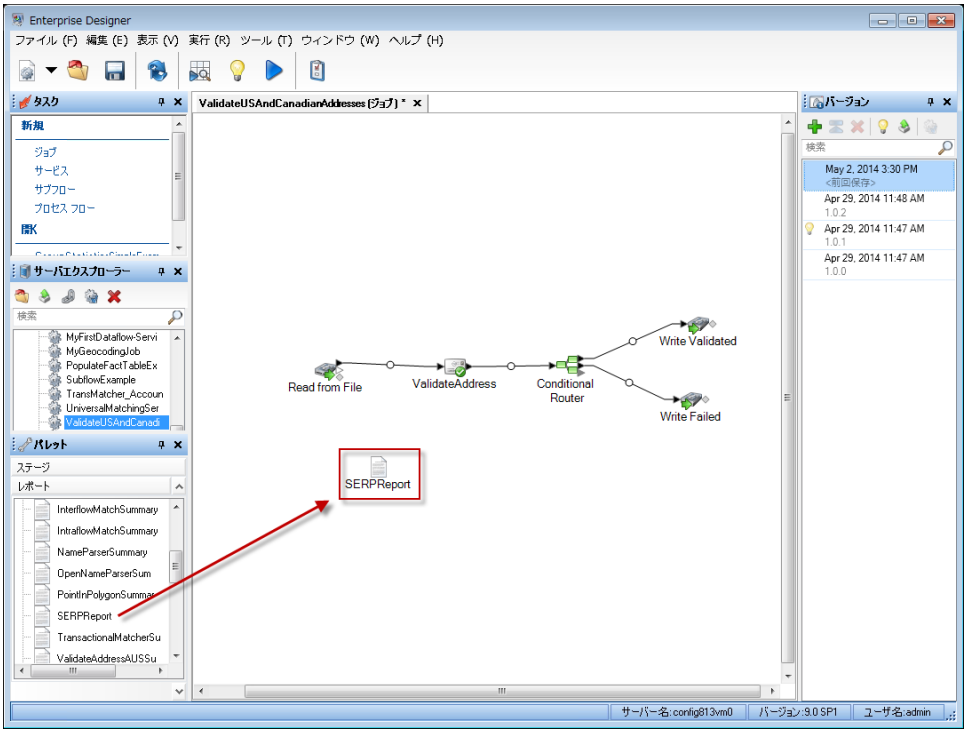
optionName	説明
CanRuralRouteFormat	地方配送路の配達情報を配置する場所を指定します。地方配送路の配達情報を含む住所の例を以下に示します。 36 GRANT RD RR 3 ANTIGONISH NS この住所において、"RR 3" は地方配送路の配達情報です。 **A** 地方配送路の配達情報を、住所と同一行の住所情報の後に配置します。こちらがデフォルトです。例を次に示します。 36 GRANT RD RR 3 **S** 地方配送路の配達情報を、別の住所行に配置します。例を次に示します。 36 GRANT RD RR 3
CanDeliveryOfficeFormat	配達局情報の配置場所を指定します。配達局情報を含む住所の例を次に示します。 PO BOX 8625 STN A ST.JOHN'S NL **I** 配達局情報を、入力住所と同じ場所に配置します。こちらがデフォルトです。 **A** 配達局情報を、住所と同一行の住所情報の後に配置します。例を次に示します。 PO BOX 8625 STN A **S** 配達局情報を、別の住所行に配置します。例を次に示します。 PO BOX 8625 STN A

optionName	説明
CanDualAddressLogic	住所に、正式情報と非正式情報の両方が含まれている場合に、ValidateAddress がストリート一致と、PO Box/非正式一致のどちらを返すかを指定します。次のいずれかです。 **D** DualAddressLogic のグローバル オプションを使用します。こちらがデフォルトです。 **P** PO Box などストリート以外のデータとマッチングします。 **S** ストリートとマッチングします。 例えば、次の入力住所が与えられたとします。 AddressLine1: 36 GRANT RD AddressLine2: RR 4 City: ANTIGONISH StateProvince: NS ValidateAddress は、次のいずれかを返します。 • `CanDualAddressLogic` が S に設定されている場合、ValidateAddress は次を返します。 AddressLine1: 36 GRANT RD AddressLine2: RR 3 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L1 • `CanDualAddressLogic` が P に設定されている場合、ValidateAddress は次を返します。 AddressLine1: RR 4 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L2 住所の正規化に使用されない住所データは、**AdditionalInputData** フィールドに返されます。詳細については、出力データ オプション（406ページ）を参照してください。

SERP の処理

1. Validate Address を SERP 認定™モードにする必要があります。ウィンドウの一番上に (SERP 認定でないもの) と表示される場合は、**[SERP 設定を有効にする]** ボタンをクリックしてください。**[SERP を設定]** ボックスが表示されます。

2. **[SERP を設定]** をクリックします。**[SERP レポート フィールド]** ダイアログ ボックスが表示されます。
3. 荷主の **[CPC 番号]** を入力します。
4. 差出人の **[名前]**、**[住所]**、および**[都市]**、**[州]**、**[ZIP]** を入力します。
5. **[OK]** をクリックします。
6. Enterprise Designer で、SERP レポートをレポートのパレットからキャンバスにドラッグします。



7. キャンバスの **[SERPReport]** アイコンをダブルクリックします。
8. **[ステージ]** タブで、**[Validate Address]** チェックボックスをオンにします。Validate Address ステージを何か別の名前に変更している場合は、住所検証ステージに指定した名前のチェック ボックスをオンにする必要があります。
9. **[パラメータ]** タブで、レポートのフォーマットを選択します。PDF、 HTML、またはプレーンテキストのフォーマットでレポートを作成できます。デフォルトではPDF フォーマットで作成されます。
10. **[OK]** をクリックします。

SERP リターン コードの取得

SERP リターン コードは、カナダ郵政公社の Software Evaluation and Recognition Program の規定によって定められる入力住所の品質を表します。

SERP リターン コードを取得するには、OutputRecordType=P を指定します。出力データ オプション（406ページ）OutputRecordTypeの詳細については、を参照してください。

SERP リターン コードは、以下の出力フィールドに返されます。

表 94 : SERP リターン コード出力

フィールド名	説明
CanadianSERPCode	検証/修正リターン コード (カナダ住所のみ)。 **V** 入力は有効です。カナダ郵政公社は、以下のすべての条件を満たす住所を、「有効な」住所であると定義しています。 **注 :** 一部例外があります。詳細については、CPC にお問い合わせください。 • 住所は、CPC の Postal Code Data Files に示されるとおりに、すべての必須コンポーネントを含む必要があります。 • 住所は、CPC の Postal Code Data Files の単一の住所のみに対し、すべてのコンポーネントにおいて完全に一致する必要があります。ただし、CPC Postal Code Data Files に示されている、許容される別の語および名前が使用されていてもかまいません。 • 住所コンポーネントは、あいまいな部分がなく、はっきりと認識できる形式である必要があります。一部のコンポーネントには、それらを識別するための "修飾子" が必要な場合があります。例えば、Route Service の住所には、同じ番号の "Suburban Service" または "SS" の住所と区別するために、"Rural Route" または "RR" のキーワードが必要です。 **I** 入力は無効です。「無効な」住所とは、有効な住所に対する CPC の条件を満たさない住所のことです (上記を参照)。例としては、住所コンポーネントが欠落している、無効である、または矛盾が存在する場合が挙げられます。 **C** 入力は修正可能です。「修正可能な」住所とは、修正することによって、単一の住所のみに一致させることのできる住所のことです。 **N** 入力は修正不可能です。「修正不可能な」住所とは、複数の異なる修正方法があり得るために ValidateAddress が単一の修正住所を特定できない住所のことです。 **F** 入力住所は外国 (カナダ以外) の住所です。

国際住所オプション

米国とカナダ以外の住所は "国際" 住所と呼ばれます。以下に、国際住所の処理をコントロールするオプションについて説明します。

オプション名	説明
PerformInternationalProcessing	国際住所 (米国およびカナダ以外の住所) を処理するかどうかを指定します。国際住所処理を有効にした場合、ValidateAddress は国際住所の妥当性を確認します。国際住所処理を無効にした場合、Statusフィールドに "F" が設定され国際住所は失敗します。出力フィールド Status.Code は、"DisabledCoder" となります。国際住所処理のライセンスを取得していない場合は、ジョブに国際住所が含まれるか否かにかかわらず、国際住所処理を無効にしなければ、ジョブを正常に実行することはできません。 **注 :** 国際住所を正常に処理するには、国際住所処理の有効なライセンスを取得する必要があります。国際住所処理のライセンスを取得していないか、ライセンスの期限が切れているにもかかわらず、国際住所処理を有効にすると、エラーとなります。 **N** 国際住所を処理しません。 **Y** 国際住所を処理します (デフォルト)。
Database.International	国際住所の妥当性の確認に使用するデータベースを指定します。国際住所検証用のデータベースを指定するには、**[データベース]** ドロップダウン リストからデータベースを選択します。Management Console の **[国際データベース リソース]** パネルで定義されたデータベースのみが使用可能です。

オプション名	説明
InternationalCityStreetSearching	デフォルトでは、ValidateAddress は、住所マッチングの精度とパフォーマンスのバランスをうまくとります。マッチング精度を犠牲にしてパフォーマンスを向上させる場合は、InternationalCityStreetSearching オプションを使用して、処理速度を上げます。これを実行すると、精度はやや低下します。このオプションは、米国およびカナダ以外の住所のパフォーマンスのみを制御します。この設定が影響を与えるレコードの割合は少なく、大部分が英国の住所です。米国およびカナダ住所処理のパフォーマンスは制御できません。 GetCandidateAddresses を使用した場合に GetCandidateAddresses が返す候補住所は、国際住所のパフォーマンス チューニング オプションを 100 以外の任意の値に設定した場合に ValidateAddress が返す複数マッチと異なることがあります。 パフォーマンスを制御するには、0 ～ 100 の値を指定してください。100 を設定すると精度が最大化し、0 を設定すると速度が最大化します。デフォルト値は 100 です。
AddressLineSearchOnFail	このオプションにより、ValidateAddress において、City、StateProvince、および PostalCode の各入力フィールドの値を使用して住所にマッチする結果が得られなかった場合に、AddressLine 入力フィールドで都市、州/省、郵便番号、および国を検索することができます。 入力住所において、AddressLine フィールドに都市、州/省、および郵便番号の情報が存在する場合は、このオプションを有効にすることを検討してください。 入力住所において、City、State/Province、および PostalCode フィールドが使用されている場合は、このオプションを無効にしてください。このオプションを有効にしてこれらのフィールドを使用すると、ValidateAddress がこれらのフィールド値の修正 (例えば、スペルミスのある都市名など) に失敗する可能性が高くなります。 **N** いいえ、AddressLine フィールドを検索しません。 **Y** はい、住所行フィールドを検索します。こちらがデフォルトです。

出力

ValidateAddress からの出力には、選択した出力カテゴリに応じて異なる情報が含まれます。

標準住所出力

標準住所出力は、宛名ラベルに表記される住所に対応する 4 行の住所で構成されます。都市、州/省、郵便番号などのデータも、標準住所出力に含まれます。`OutputRecordType=A` と設定した場合、妥当性を確認した住所に対し、標準住所出力が返されます。妥当性が確認できなかった住所に対しては、標準住所フィールドが必ず返されます。妥当性が確認されなかった住所に対しては、標準住所出力フィールドには、入力住所がそのまま含まれます ("パス スルー" データ)。妥当性が確認できなかった場合に、郵便当局の規格に従って住所を正規化するには、リクエスト時に `OutputFormattedOnFail=Y` を指定します。

表 95 : 標準住所出力

フィールド名	説明
AdditionalInputData	住所検証プロセスで使用されない入力データ。詳細については、**AdditionalInputData について**（488ページ）を参照してください。
AddressLine1	住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 1 行目です。住所の妥当性が確認できなかった場合は、入力住所の 1 行目がそのまま出力されます。
AddressLine2	住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 2 行目です。住所の妥当性が確認できなかった場合は、入力住所の 2 行目がそのまま出力されます。
AddressLine3	住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 3 行目です。住所の妥当性が確認できなかった場合は、入力住所の 3 行目がそのまま出力されます。
AddressLine4	住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 4 行目です。住所の妥当性が確認できなかった場合は、入力住所の 4 行目がそのまま出力されます。

フィールド名	説明
AddressLine5	英国住所にのみ適用されます。住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の5行目です。住所の妥当性が確認できなかった場合は、入力住所の 5 行目がそのまま出力されます。
City	妥当性が確認された都市名。
Country	OutputCountryFormat で選択した、以下のいずれかのフォーマットで示された国。 • ISO コード • UPU コード • 英語 • フランス語 • ドイツ語 • スペイン語
DepartmentName	企業内の下位区分(英国住所においてのみ)。例えば、エンジニアリング部門などです。
FirmName	妥当性が確認された企業名。
PostalCode	妥当性が確認された ZIP Code™ または郵便番号。
PostalCode.AddOn	ZIP Code™ の 4 桁のアドオン部分。例えば、60655-1844 という ZIP Code™ において、4 桁のアドオン部分は 1844 になります(米国住所のみ)。
PostalCode.Base	5 桁の ZIP Code™。住所のみ)。
StateProvince	妥当性が確認された州または省の略称。
USUrbanName	妥当性が確認された都市の都市化名。(米国住所のみ)。主にプエル トリコの住所で使用されます。

パース済み住所要素出力

`OutputRecordType=E` を設定した場合、出力住所は、パース済み住所の形式で書式設定されます。妥当性が確認できなかった場合に、パース済み住所形式で書式設定されたデータ (正規化済み住所) を返すには、`OutputFormattedOnFail=Y` を指定します。

> **注：** 妥当性が確認できたかどうかにかかわらず、常にパースした入力データを返すには、`OutputRecordType=I` を指定します。詳細については、「**パース済み入力**（455ページ）」を参照してください。

表 96：パース済み住所出力

columnName	説明
AdditionalInputData	ValidateAddress で使用されない入力データ。詳細については、「**AdditionalInputData について**（488ページ）」を参照してください。
AdditionalInputData.Base	ValidateAddress によって正規化済み住所に出力されなかった入力データ。詳細については、「**AdditionalInputData について**（488ページ）」を参照してください。
AdditionalInputData.Unmatched	マッチャーに引き渡されたが、ValidateAddress による検証に使用されなかった入力データ。詳細については、「**AdditionalInputData について**（488ページ）」を参照してください。
ApartmentLabel	アパート指定子 (STE や APT など)。例: 123 E Main St **APT** 3
ApartmentLabel2	補助的なアパート指定子。例: 123 E Main St Apt 3, 4th **Floor** **注：** このリリースでは、このフィールドは常に空白になります。
ApartmentNumber	アパート番号。例: 123 E Main St APT **3**

columnName	説明
ApartmentNumber2	補助的なアパート番号。例: 123 E Main St APT 3, **4th** Floor **注 :** このリリースでは、このフィールドは常に空白になります。
CanadianDeliveryInstallationAreaName	配達施設名 (カナダ住所のみ)
CanadianDeliveryInstallationQualifierName	配達施設の修飾子 (カナダ住所のみ)
CanadianDeliveryInstallationType	配達施設の種類 (カナダ住所のみ)
City	妥当性が確認された都市名
Country	国。フォーマットは、`OutputCountryFormat` で選択したものになります。 • ISO コード • UPU コード • 英語 • フランス語 • ドイツ語 • スペイン語
DepartmentName	英国(英国住所においてのみ)。例えば、エンジニアリング部門などです。
FirmName	妥当性が確認された企業名
HouseNumber	家番号 1。例: **123** E Main St Apt 3
LeadingDirectional	接頭方向指示。例: 123 **E** Main St Apt 3

columnName	説明
POBox	私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode	妥当性が検証された郵便番号。米国住所に対しては、ZIP Code と呼びます。
PrivateMailbox	私設私書箱のインジケータ。
PrivateMailbox.Type	私設私書箱の種別。次のような値があります。 • Standard • Non-Standard **注：** PrivateMailboxType (フィールド名にピリオドなし) に換わるものです。これに従って API 呼び出しを修正してください。
RRHC	地方配送路/幹線請負契約のインジケータ
StateProvince	妥当性が確認された州または省の名前
StreetName	通り名。例: 123 E **Main St** Apt 3
StreetSuffix	通り接尾語。例: 123 E Main **St** Apt 3
TrailingDirectional	接尾方向指示。例: 123 Pennsylvania Ave **NW**
USUrbanName	USPS® 都市化名。プエルトリコ住所のみ。

パース済み入力

出力には、パース済み形式で入力住所を含めることができます。このようなタイプの出力は、"パース済み入力" と呼ばれます。パース済み入力フィールドには、ValidateAddress が住所の妥当性を検証したかどうかにかかわらず、入力として使用される住所データが含まれます。パース済み入力は、住所の妥当性を検証できた場合にパース済み住所要素に妥当性が検証された住所が含まれ、オプションで、住所の妥当性が検証できなかった場合には入力データが含まれるという点で、"パース済み住所要素" 出力と異なります。パース済み入力には、ValidateAddress が住所の妥当性を検証したかどうかにかかわらず、常に入力住所が含まれます。

パース済み入力フィールドを出力に含めるには、`OutputRecordType=I` を設定します。

表 97 : パース済み入力

フィールド名	説明
ApartmentLabel.Input	アパート指定子 (STE や APT など)。例: 123 E Main St **APT** 3
ApartmentNumber.Input	アパート番号。例: 123 E Main St APT **3**
CanadianDeliveryInstallationAreaName.Input	配達施設名 (カナダ住所のみ)
CanadianDeliveryInstallationQualifierName.Input	配達施設の修飾子 (カナダ住所のみ)
CanadianDeliveryInstallationType.Input	配達施設の種類 (カナダ住所のみ)
City.Input	妥当性が確認された都市名

フィールド名	説明
Country.Input	国フォーマットは、OutputCountryFormat で選択したものになります。 • ISO コード • UPU コード • 英語 • フランス語 • ドイツ語 • スペイン語
FirmName.Input	妥当性が確認された企業名
HouseNumber.Input	家番号。例: **123** E Main St Apt 3
LeadingDirectional.Input	接頭方向指示。例: 123 **E** Main St Apt 3
POBox.Input	私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode.Input	妥当性が検証された郵便番号。米国住所に対しては、ZIP Code と呼びます。
PrivateMailbox.Input	私設私書箱のインジケータ
PrivateMailbox.Type.Input	私設私書箱の種別。次のような値があります。 • 標準 • 非標準
RRHC.Input	地方配送路/Highway Contract のインジケータ
StateProvince.Input	妥当性が確認された州または省の名前

フィールド名	説明
StreetName.Input	ストリート名。例: 123 E **Main St** Apt 3
StreetSuffix.Input	ストリート接尾語。例: 123 E Main St Apt 3
TrailingDirectional.Input	接尾方向指示。例: 123 Pennsylvania Ave **NW**
USUrbanName.Input	USPS® 都市化名

郵便データ出力

`OutputRecordType` に P が含まれる場合、以下のフィールドが出力として返されます。

表 98 : 郵便データ出力

columnName	説明
CanadianSERPCode	検証/修正リターン コード (カナダ住所のみ)。詳細については、**SERP リターン コードの取得**（446ページ）を参照してください。
IntHexaviaCode	ストリートを表す数値コード (フランスの住所の場合のみ)。Hexavia コードの詳細については、**www.laposte.fr** を参照してください。
IntINSEECode	都市を表す数値コード (フランスの住所の場合のみ)。INSEE コードの一覧については、**www.insee.fr** を参照してください。
PostalBarCode	配達ポイント バーコードの 2 桁の配達ポイント部分 (米国住所のみ)。詳細については、**配達ポイント バーコードの作成**（416ページ）を参照してください。

columnName	説明
USAltAddr	他の住所マッチング ロジックを使用したかどうか、使用した場合はどのロジックを使用したかを表します (米国住所のみ)。次のいずれかです。 **NULL** 他の住所スキームを使用していません。 **D** 別の配達ポイント ロジックを使用しました。 **E** 別の高層マッチ ロジックを使用しました。 **S** 小都市デフォルト ロジックを使用しました。 **U** ユニークな ZIP Code ロジックを使用しました。
USBCCheckDigit	11 桁の配達ポイント バーコードのチェック ディジット部分 (米国住所のみ)。詳細については、**配達ポイント バーコードの作成**（416ページ）を参照してください。
USCarrierRouteCode	配達ルート コード (米国住所のみ)。詳細については、**配達ルート コードの取得**（416ページ）を参照してください。
USCongressionalDistrict	下院選挙区出力 (米国住所のみ)。詳細については、**下院選挙区の取得**（414ページ）を参照してください。
USCountyName	郡名出力 (米国住所のみ)。詳細については、**郡名の取得**（415ページ）を参照してください。
USFinanceNumber	住所の所在地の Finance Number (米国住所のみ)。Finance Number とは、複数の ZIP Code を含む地域に USPS が割り当てた番号です。住所の Finance Number が米国データベースの候補住所の Finance Number に一致した場合のみ、住所の妥当性確認に成功します。
USFIPSCountyNumber	FIPS (連邦情報処理標準) 郡番号 (米国住所のみ)。詳細については、**FIPS 郡番号の取得**（415ページ）を参照してください。

columnName	説明
USLACS	住所が、LACS^{Link} 変換の候補であるかどうかを表します (米国住所のみ)。次のいずれかです。 **Y** 住所は LACS^{Link} 処理の候補です。LACS^{Link} が有効である場合、LACS^{Link} データベースを使用して住所を変換しようとします。変換に成功した場合、出力住所は LACS^{Link} データベースから取得した新しい住所になります。変換できなかった場合は、住所は変換されません。 **N** 住所は LACS^{Link} 処理の候補ではありません。ただし、LACS^{Link} 処理が要求され、LACS^{Link} データベースがインストールされており、かつ、次の条件のいずれかが満たされている場合は、LACS^{Link} 処理が行われる場合があります。 • 住所が地方配送路住所にマッチし、RecordType.Default フィールドで Y が返された場合。 • 入力住所が、米国郵便データベースのいずれの住所にもマッチしなかった場合 (複数にマッチしたことによる失敗は、LACS^{Link} の候補にはなりません)。
USLastLineNumber	主要都市が同一である複数の ZIP Code を同一グループにまとめる 6 文字の英数字の値。例えば、最終行が次の 2 つのいずれかである住所は、最終行番号が同一になります。 Chantilly VA 20151 Chantilly VA 20152

結果インジケータ

結果インジケータは、住所に対して実行した処理の種類に関する情報を提供します。結果インジケータには、次の 2 種類があります。

レコード レベルの結果インジケータ

レコード レベルの結果インジケータは、各レコードに対する ValidateAddress 処理の結果に関するデータを提供します。例えば、マッチング試行の成功または失敗、住所を処理したコーダーなどの詳細情報を示します。以下の表に、ValidateAddress が返すレコード レベルの結果インジケータの一覧を示します。

表 99 : レコード レベル インジケータ

columnName	説明
AddressFormat	返された住所データのタイプ。 **F** フランス語フォーマット (例: 123 Rue Main) **E** 英語フォーマット (例: 123 Main St)
Confidence	返された住所に割り当てられた確信レベル。範囲は 0 ～ 100 です。0 は失敗を表し、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表します。複数のマッチ結果がある場合、確信レベルは 0 です。この数値の計算方法については、**Validate Address 確信アルゴリズムの概要**（562ページ） を参照してください。
CouldNotValidate	マッチしなかった場合の、妥当性が確認できなかった住所コンポーネント。 • ApartmentNumber • HouseNumber • StreetName • PostalCode • 都市 • Directional • StreetSuffix • Firm • POBoxNumber • RuralRoute **注 :** 複数のコンポーネントがカンマ区切りリストとして返されることがあります。

columnName	説明
CountryLevel	利用可能な住所マッチングのカテゴリ。米国およびカナダの住所に対しては、常に「A」です。次のいずれかです。 **A** 住所は非常に詳細な郵便データを利用できる国にあります。このマッチ レベルにある住所では、以下の住所要素を検証および修正でき、入力から欠落している場合は追加できます。 • 郵便番号 • 都市名 • 州/郡名 • 通り住所要素 • 国名 **B** 住所は中程度の詳細さの郵便データを利用できる国にあります。このマッチ レベルにある住所では、以下の住所要素を検証および修正でき、入力から欠落している場合は追加できます。 • 郵便番号 • 都市名 • 州/郡名 • 国名 **C** 住所は郵便データが詳細ではない国にあります。このマッチ レベルにある住所に対して、以下のアクションを実行することができます。 • 国名の検証および修正 (欠落している国名を補うことはできません) • 郵便番号のフォーマットの検証 (欠落している郵便番号を補ったり、番号を検証することはできません)

columnName	説明
MatchScore	MatchScore は、出力住所がどの程度正しいかを示します。MatchScore は、マッチ結果を得るために入力住所をどれだけ変更したかを表す Confidence とはまったく異なるものです。MatchScore の意味は、米国住所と米国以外の住所で異なります。 米国住所に対しては、MatchScore は 0 ～ 9 の段階に対応する 1 桁のスコアで、ストリート名マッチの近さを反映します (ValidateAddress による変換があれば実行後)。0 は完全一致を意味し、9 は最も可能性の低い一致を意味します。マッチしなかった場合、このフィールドは空白です。 米国とカナダ以外の住所では、MatchScore は 5 桁のスコアで、最大値は 00999 です。数字が大きいほど、より近い一致を意味します。 このフィールドは、カナダの住所には適用されません。 米国住所のマッチ スコアと米国以外の住所のマッチ スコアは、同等と見なすことはできないことに注意してください。例えば、米国住所に対するマッチ スコア 4 は、米国以外の住所に対する 00004 と同じマッチ レベルを意味するものではありません。 **注 :** Validate Address および Advanced Matching モジュールのコンポーネントは、どちらも MatchScore フィールドを使用します。データフローの出力の MatchScore フィールドの値は、出力ステージに送られる前に最後に値を変更したステージによって決まります。データフローに Validate Address および Advanced Matching モジュールのコンポーネントが含まれ、各ステージの MatchScore 出力フィールドを確認したい場合は、Transformer ステージを使用して、MatchScore 値を他のフィールドにコピーしてください。例えば、Validate Address によって MatchScore という出力フィールドが作成され、Transformer ステージによって Validate Address の MatchScore フィールドが AddressMatchScore というフィールドにコピーされます。マッチャー ステージを実行すると、マッチャーから得た値が MatchScore フィールドに設定され、Validate Address から得た AddressMatchScore の値が 引き渡されます。
MultimatchCount	複数のマッチが検出された場合、一致する可能性のあるレコードの数を示します。

columnName	説明
MultipleMatches	複数のマッチが検出された場合に、複数のマッチを持つ次の住所コンポーネントを示します。 • Firm • LeadingDirectional • PostalCode • StreetName • StreetSuffix • TrailingDirectional • Urbanization **注：** 複数のコンポーネントがカンマ区切りリストとして返されることがあります。
ProbableCorrectness	検出された全体のマッチについて推定される相対的な正確性 **空白** マッチが見つかりません。 **0** マッチは正確である可能性が最も高いです。 **1-8** マッチ レベルは中間で、状況により変動します。 **9** マッチは正確である可能性が最も低いです。 **注：** これらの値は、プログラムによる相対的な正確性の推定のみを反映しています。場合により、スコア 0 と評価されたマッチが正確でなかったり、スコア 9 と評価されたマッチが正確であったりする可能性もあります。
ProcessedBy	住所を処理した住所コーダーです。 **USA** 米国住所コーダー **CAN** カナダ住所コーダー **INT** 国際住所コーダー
RecordType	米国およびカナダの郵政当局が定義した住所レコードのタイプ (米国およびカナダの住所のみサポート): • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal

columnName	説明
RecordType.Default	"デフォルト" マッチを示すコード **Y** 住所はデフォルト レコードにマッチしています。 **NULL** 住所はデフォルト レコードにマッチしていません。
Status	マッチの成功または失敗。複数のマッチがある場合、一致する可能性のあるすべてのものに対してこのフィールドが "F" になります。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。複数のマッチがある場合、一致する可能性のあるすべてのものが "MultipleMatchesFound" になります。 • DisabledCoder • InsufficientInputData • MultipleMatchesFound • UnableToValidate
Status.Description	問題の説明 (ある場合)。 **Possible Multiple Addresses Found** Status.Code=MultipleMatchesFound の場合にこの値が表示されます。 **Address Not Found** Status.Code=UnableToValidate の場合にこの値が表示されます。 **PerformUSProcessing disabled** Status.Code=DisabledCoder の場合にこの値が表示されます。 **PerformCanadianProcessing disabled** Status.Code=DisabledCoder の場合にこの値が表示されます。 **PerformInternationalProcessing disabled** Status.Code=DisabledCoder の場合にこの値が表示されます。

フィールドレベルの結果インジケータ

フィールドレベルの結果インジケータは、ValidateAddress が各住所要素をどのように処理したかを示します。フィールドレベルの結果インジケータは、修飾子 "Result" で返されます。例えば、

HouseNumber のフィールドレベルの結果インジケータは **HouseNumber.Result** に格納されます。

フィールドレベルの結果インジケータを有効にするには、`OutputFieldLevelReturnCodes=Y` を指定します。詳細については、出力データ オプション（406ページ）を参照してください。

次の表に、フィールドレベルの結果インジケータの一覧を示します。特定のフィールドが住所に適用されない場合、結果インジケータが空白になる場合があります。

表 100：フィールドレベルの結果インジケータ

columnName	説明
AddressRecord.Result	これらの結果コードは国際住所のみに適用されます。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
ApartmentLabel.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国およびカナダの住所のみをサポートします。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国およびカナダの住所のみをサポートします。 **R** アパートラベルが必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。カナダの住所には適用されません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
ApartmentNumber.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。カナダの住所のみ。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国EWS にマッチする米国の住所には、P の値が割り当てられます。米国およびカナダの住所のみをサポートします。 **R** アパート番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
City.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **F** ハイフンの欠落または句読文字エラー。カナダの住所のみ。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。 **R** 都市名が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。 **U** マッチしない。カナダの住所には適用されません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
Country.Result	これらの結果コードは、米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
FirmName.Result	**C** 修正済み。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国およびカナダの住所のみをサポートします。 **U** マッチしない。米国およびカナダの住所のみをサポートします。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。米国住所にのみ適用されます。
HouseNumber.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。 **C** 修正済み。カナダの住所のみ。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **O** 範囲外。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **R** 家番号が必須ですが、入力住所から欠落しています。カナダの住所のみ。 **S** 正規化。このオプションには、標準の略語が含まれます。米国またはカナダの住所には適用されません。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
LeadingDirectional.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。空白のない入力が、空白のない値に修正されました。米国住所にのみ適用されます。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。カナダの住所には適用されません。

columnName	説明
POBox.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。 **C** 修正済み。カナダの住所のみ。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数マッチ。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **R** 私書箱番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
PostalCode.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。カナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国住所をサポートしていません。 **R** 郵便番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国またはカナダの住所には適用されません。 **U** マッチしない。例えば、ストリート名と郵便番号が一致しない場合、StreetName.Result と PostalCode.Result の両方に U が割り当てられます。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
PostalCodeCity.Result	これらの結果コードは国際住所のみに適用されます。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
PostalCode.Source	これらの結果コードは米国住所にのみ適用されます。 **FinanceNumber** 入力の ZIP Code™ は、USPS® Finance Number グループを使って検証されました。 **ZIPMOVE** 入力住所の ZIP Code™ は、USPS® が改訂した ZIP Code™ 境界に基づいて修正され、住所に別の ZIP Code™ が設定されました。
PostalCode.Type	**P** ZIP Code™ には、PO Box 住所のみが含まれます。米国住所にのみ適用されます。 **U** ZIP Code™ は、特定の会社または場所に割り当てられたユニークな ZIP Code™ です。米国住所にのみ適用されます。 **M** ZIP Code™ は、軍施設の住所です。米国住所にのみ適用されます。 **NULL** ZIP Code™ は、標準 ZIP Code™ です。

columnName	説明
RRHC.Result	**C** 修正済み。カナダの住所のみ。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **M** 複数マッチ。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **R** 地方配送路/Highway Contract が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国およびカナダの住所のみをサポートします。 **U** マッチしない。米国およびカナダの住所のみをサポートします。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。米国およびカナダの住所のみをサポートします。
RRHC.Type	これらの結果コードは米国住所にのみ適用されます。 **HC** 住所は、Highway Contract 住所です。 **RR** 住所は、地方配送路住所です。

columnName	説明
StateProvince.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国住所にのみ適用されます。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。 **R** アパートラベルが必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。 **U** マッチしない。カナダの住所には適用されません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
Street.Result	これらの結果コードは国際住所のみに適用されます。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。 **R** ストリートが修正済みです。家番号が範囲外にあります。フランス、英国、および日本のレコードのみに適用。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
StreetName.AbbreviatedAlias.Result	省略形エイリアス処理の結果を示します。次のいずれかです。 **NULL** 省略形エイリアス処理が実行されませんでした。 **B** StreetName フィールドに基本ストリート名が格納されています。 **L** 正規化された住所長が 31 文字未満なので、StreetName フィールドに基本名が格納されています。 **N** 省略形エイリアスが見つかりませんでした。 **Y** 省略形エイリアスが入力住所に見つかりました。StreetName フィールドに省略形エイリアスが格納されています。

columnName	説明
StreetName.Alias.Type	この結果コードは米国住所にのみ適用されます。 **注：** 以前のリリースでは、このフィールドは "Alias" と "Type" の間に "." がない StreetName.AliasType という名前でした。この古い名前は廃止されました。新しい名前 StreetName.Alias.Type を使用するよう、プロセスを更新してください。 **Abbreviated** エイリアスはストリート名の省略形です。例えば、HARTS-NM RD は HARTSVILLE NEW MARLBORO RD の省略形エイリアスです。 **Changed** ストリート名が正式に変更された場合に、新しい名前を表すエイリアスです。例えば、SHINGLE BROOK RD というストリート名が CANNING DR に変更された場合、CANNING DR が変更済みのエイリアス タイプとなります。 **Other** このストリート名エイリアスには、ストリートの他の名前や、ストリートの一般的な省略形などがあります。 **Preferred** ストリート名エイリアスはその地域でよく使用されるエイリアスです。例えば、あるストリートが "South Shore Dr." という名前なのは、湖の南岸を通っているためで、地方自治体の境界線の南にあるからではありません。この場合、"South" は前置方位記号ではないので、"S" と短縮してはいけません。したがって、"South Shore Dr." がよく使用されるエイリアスになります。

columnName	説明
StreetName.PreferredAlias.Result	よく使用されるエイリアス処理の結果を示します。次のいずれかです。 **NULL** よく使用されるエイリアス処理が実行されませんでした。 **A** 入力住所がエイリアスにマッチしたため、よく使用されるエイリアス処理が実行されませんでした。よく使用されるエイリアス処理は、基本住所に対してのみ実行されます。 **N** よく使用されるエイリアスが見つかりませんでした。 **Y** 入力住所に対してよく使用されるエイリアスが見つかりました。StreetName フィールドによく使用されるエイリアスが格納されています。

columnName	説明
StreetName.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国住所にのみ適用されます。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国住所をサポートしていません。 **S** 正規化。このオプションには、標準の略語が含まれます。米国およびカナダの住所のみをサポートします。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
StreetSuffix.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国およびカナダの住所のみをサポートします。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。米国住所をサポートしていません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
TrailingDirectional.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **D** ドロップ済み入力に与えられたフィールドが削除されました。米国およびカナダの住所のみをサポートします。詳細については、**AdditionalInputData について**（488ページ）を参照してください。 **F** 書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。カナダの住所には適用されません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
USUrbanName.Result	これらの結果コードは米国住所にのみ適用されます。 **A** 追加済み。フィールドが空白の入力フィールドに追加されました。 **C** 修正済み。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

オプションによる出力

ValidateAddress は、選択したオプションに応じて、追加のデータを返します。各オプションによって生成される出力については、以下のセクションのオプションの一覧を参照してください。

Enhanced Line of Travel 出力

Enhanced Line of Travel 処理では、以下の出力を生成します。

フィールド名	説明
USLOTCode	Line of Travel の順序コードと、USPS® LOT 順序を表すインジケータ。このフィールドは、"nnnnY" という形式で、以下の要素で構成されます。 **nnnn** 4 桁の LOT コード。 **Y** 次のいずれかです。 • **A** — 昇順の LOT 順序 • **D** — 降順の LOT 順序
USLOTHex	昇順でのみファイルのソートが可能な 16 進値です。16 進値の範囲は、昇順で 0 ～ FF に続き、降順で FF ～ 0 です。
USLOTSequence	アドオンの DPC の代わりに最終仕分けに使用される 2 バイトの値。大文字 1 文字の後に 0 ～ 9 の数字が 1 つ続きます。値の範囲は、A0 (99 降順) ～ J9 (00 降順)、および K0 (00 昇順) ～ T9 (99 昇順) です。

LACS^{Link} 出力

フィールド名	説明
USLACS	住所が、LACS^{Link} 変換の候補であるかどうかを表します (米国住所のみ)。次のいずれかです。 **Y** 住所は LACS^{Link} 処理の候補です。LACS^{Link} が有効である場合は、ValidateAddress は、LACS^{Link} データベースを使用して住所を変換します。変換に成功した場合、出力住所は LACS^{Link} データベースから取得した新しい住所になります。変換できなかった場合は、住所は変換されません。 **N** 住所は LACS^{Link} 処理の候補ではありません。ただし、LACS^{Link} 処理が要求され、LACS^{Link} データベースがインストールされており、かつ、次の条件のいずれかが満たされている場合は、LACS^{Link} 処理が行われる場合があります。 • 住所が地方配送路住所にマッチし、RecordType.Default フィールドで Y が返された場合。 • 入力住所が、米国郵便データベースのいずれの住所にもマッチしなかった場合 (複数にマッチしたことによる失敗は、LACS^{Link} の候補にはなりません)。
USLACS.ReturnCode	LACS^{Link} 処理の成功または失敗を表します(米国住所のみ)。 **A** LACS^{Link} 処理は成功しました。LACS^{Link} 処理によってレコードがマッチしました。 **00** LACS^{Link} 処理は失敗しました。LACS^{Link} 処理において、マッチするレコードは見つかりませんでした。 **09** LACS^{Link} 処理において、入力住所は、古い高層のデフォルト住所にマッチしました。住所は変換されています。不明確な住所の提供を避け、LACS^{Link} 処理では、新しい住所を提供しません。 **14** LACS^{Link} 処理は失敗しました。LACS^{Link} 処理において、マッチする結果が検出されましたが、他の USPS® の規則に基づき、変換は行われませんでした。 **92** LACS^{Link} 処理は成功しました。LACS^{Link} 処理によってレコードがマッチしました。入力のユニット番号はドロップされました。 **NULL** LACS^{Link} はレコードを処理しなかったか、または LACS^{Link} 処理が実行されませんでした。

RDI 出力

フィールド名	説明
RDI	住所の種類を表す値を返します。 **B** 住所は、企業住所です。 **R** 住所は、個人住所です。 **M** 住所は、個人住所であるとともに企業住所でもあります。 **NULL** 住所が ZIP + 4® レベルでコード化されなかったか、または RDI™ が実行されなかったため、確認されていません。

DPV および CMRA 出力

フィールド名	説明
DPV	Delivery Point Validation (DPV) 処理の結果を表します。 **Y** DPV の確認済みです。この住所に郵便物を配達できます。 **N** この住所に郵便物を配達できません。 **S** 建物番号の妥当性は確認できましたが、ユニット番号は確認できませんでした。建物番号は、建物の主要な住所番号です。ユニット番号は、建物内のアパート、スイート、階など、各ユニットの郵便住所番号です。例えば、以下の住所の場合、424 は建物番号、12 はユニット番号です。 424 Washington Blvd.Apt.12 Oak Park IL 60302 USA **D** 建物番号の妥当性は確認できましたが、ユニット番号は入力から欠落していました。建物番号は、建物の主要な住所番号です。ユニット番号は、建物内のアパート、スイート、階など、各ユニットの郵便住所番号です。例えば、以下の住所の場合、424 は建物番号、12 はユニット番号です。 424 Washington Blvd.Apt.12 Oak Park IL 60302 USA **M** 住所は複数の有効な配達ポイントにマッチします。 **U** 住所は、ZIP + 4® レベルでコード化されなかったため、確認できませんでした。 **V** 住所は、誤検出違反を起こしました。
CMRA	住所が Commercial Mail Receiving Agency (CMRA: 民間私書箱) であることを表します。 **Y** 住所は CMRA です。 **N** 住所は CMRA ではありません。 **U** 確認できません。

フィールド名	説明
DPVFootnote	DPV 補足コード。 **AA** 入力住所は、ZIP + 4® ファイルにマッチしました。 **A1** 入力住所は、ZIP + 4® ファイルにマッチしませんでした。 **BB** 入力住所は、DPV にマッチしました (すべてのコンポーネント)。 **CC** 入力住所の主要な番号は DPV にマッチしましたが、補助的な番号はマッチしませんでした (存在しましたが有効ではありませんでした)。 **F1** 入力住所は軍関係の住所。DPV は省かれます。 **G1** 入力住所は一般的な配達住所。DPV は省かれます。 **M1** 入力住所の主要な番号が欠落しています。 **M3** 入力住所の主要な番号が無効です。 **N1** 入力住所の主要な番号は DPV にマッチしましたが、高層住所に補助的な番号が欠落しています。 **P1** 入力住所に、RR または HC Box 番号がありません。 **P3** 入力住所に、PO、RR、または HC Box 番号がありません。 **RR** 入力住所は、CMRA にマッチしました。 **R1** 入力住所は CMRA にマッチしましたが、補助的な番号が存在しません。 **U1** 入力住所はユニーク ZIP。DPV は省かれます。
DPVVacant	建物が空家 (90 日間使用されていない) かどうかを表します。次のいずれかです。 **Y** 建物は空家です。 **N** 建物は空家ではありません。 **NULL** `DPVDetermineVacancy` オプションが選択されていません。
DPVNoStat	建物が、郵便物を受け取ることのできない "no stat" の建物であるかどうかを表します。次のいずれかです。 **Y** 建物は、郵便物を受け取ることのできない "no stat" の建物です。 **N** 建物は、郵便物を受け取ることのできない "no stat" の建物ではありません。 **NULL** `DPVDetermineNoStat` オプションが選択されていません。

SuiteLink 出力

フィールド名	説明
SuiteLinkReturnCode	ValidateAddress が、補助的な住所情報を修正したかどうかを表します (米国住所のみ)。次のいずれかです。 **A** ValidateAddress は、補助的な住所情報を修正しました。 **00** ValidateAddress は、補助的な住所情報を修正しませんでした。 **NULL** SuiteLink は実行されませんでした。 **XX** SuiteLink 処理においてエラーが発生しました。例えば、SuiteLink データベースの有効期限が切れている場合にエラーが発生します。
SuiteLinkMatchCode	SuiteLink のマッチング処理に関する追加情報を提供します(米国mail stop (郵便物集配所)(米国住所のみ) **A** ValidateAddress は、補助的な住所情報を修正しました。 **B** ValidateAddress は、補助的な住所情報を修正しませんでした。マッチング処理に関するその他の詳細情報はありません。 **C** FirmName フィールドの語はすべて、「ノイズ」語です。ノイズ語は、USPS® によって定義されており、企業名のマッチングの際には無視されます。ノイズ語の例としては、"company" や "corporation" があります。ValidateAddress は、ノイズ語のみで構成される企業名に対し、補助的な住所情報を修正できません。例えば、"Company and Corporation" という表記は、ノイズ語のみで構成されています。 **D** 住所は、高層のデフォルト住所ではありません。SuiteLink マッチングは、高層のデフォルト住所に対してのみ行われます。高層デフォルトとは、住所に有効な補助的情報が含まれていない (アパート番号やアパート種別が欠落している) 場合に使用されるデフォルトです。 **E** SuiteLink データベースの有効期限が切れているため、SuiteLink 処理は失敗しました。 **NULL** SuiteLink は実行されなかったか、エラーが発生しました。

フィールド名	説明
SuiteLinkFidelity	ValidateAddress における、SuiteLink データベースの企業名に対するマッチング精度を表します。 **1** 企業名は、SuiteLink データベースに完全に一致しました。 **2** 精度の高いマッチです。企業名に含まれる語が、1 語を除いてすべて SuiteLink データベースの企業名に一致しました。 **3** 精度の低いマッチです。企業名の中の複数の語が、SuiteLink データベースの企業名に一致しませんでした。 **NULL** SuiteLink が企業名のマッチングに失敗したか、実行されなかったか、またはエラーが発生しました。

VeriMove 出力

フィールド名	説明
VeriMoveDataBlock	ValidateAddress が、VeriMove Express に渡される入力データを含む 250 バイトのフィールドを返すかどうかを示します。このフィールドには、VeriMove で必要とされる詳細結果インジケータ データが含まれます。このフィールドの内容については、VeriMove のユーザー ガイドを参照してください。次のいずれかです。 **Y** フィールド VeriMoveDataBlock を返します。 **N** フィールド VeriMoveDataBlock を返しません。

AdditionalInputData について

ValidateAddress は、住所正規化プロセスにおいて、一部の入力データを無視します。この余分なデータ ("ドロップ データ" と呼ばれることもあります) は、AdditionalInputData 列に返されます。ドロップ データの例としては、次のものがあります。

- 配達指示 (「勝手口に置いてください」など)
- 電話番号 ("555-135-8792" など)
- 注意書き ("Attn: John Smith" など)

このようなデータは通常、住所に混在していることはありません。混在している場合、ValidateAddress はほとんどの場合にこの余分なデータを認識することができ、AdditionalInputData 列に返します。

注 : ValidateAddress は、split indicia 住所からのドロップ データを返しません。split indicia 住所とは、主要な住所が複数の住所行に分割されている住所のことです。例えば、主要な住所が "1 Green River Valley Rd" である場合、次のようになります。

1 Green River
Valley Rd
01230

住所に複数のドロップ データがある場合、各データは、米国住所の場合はセミコロンと空白 ("; ")、米国外の住所の場合は空白で区切られます。AdditionalInputData におけるドロップ データの順序は、次のようになります。

1. care of (気付)、mail stop (郵便物集配所)(米国住所のみ)
2. 住所行に検出されたその他の余分なデータ
3. まったく未使用のデータ行

例えば、入力住所が次のとおりであるとします。

123 Main St C/O John Smith
Apt 5 Drop at back dock
jsmith@example.com
555-123-4567
05674

この場合、AdditionalInputData には次のデータが含まれます。

C/O John Smith; Apt 5 Drop At Back Dock; 555-123-4567; Jsmith@g1.Com; 555-123-4567

ValidateAddress では、以下の種類の余分なデータを処理できます。

Care Of (気付) データ

米国住所に対しては、"care of" データが AdditionalInputData に返されます。以下の住所には、"care of" データの例が含まれています。

123 Main St C/O John Smith
Apt 5
05674

123 Main St
Apt 5 ATTN John Smith
05674

123 Main St Apt 5
MailStop 2
05674

独立した住所行に存在する余分なデータ

ValidateAddress は、米国およびカナダの住所に対し、独立した住所行に余分なデータを返します。

米国住所に対しては、住所行の空白でない最初の 2 行を使用して、住所の正規化を行います。ただし、企業名抽出または都市化コード抽出のオプションが有効である場合を除きます (詳細については、「**米国住所の住所行処理**（405ページ）」を参照してください)。他の住所行に存在するデータは、AdditionalInputData に返されます。以下の住所において、"John Smith" は AdditionalInputData に返されます。"John Smith" は 空白でない 3 つめの住所行に存在しており、ValidateAddress は米国住所に対して、空白でない最初の 2 つの住所行のみを使用するためです。

123 Main St
Apt 5
John Smith
05674

空白でない最初の 2 つの住所行に余分なデータが含まれる場合、そのデータは AdditionalInputData に返されます。例えば、以下の住所において、"John Smith" は AdditionalAddressData に返されます。

123 Main St
John Smith
05674

John Smith
123 Main St
05674

以下の住所では、"John Smith" と "Apt 5" の両方が AdditionalAddressData に返されます。"John Smith" が返されるのは、これが、最初の 2 つの住所行の 1 つに存在する余分なデータであるためです。"Apt 5" が返されるのは、米国住所データは、空白でない最初の 2 行に記載しなければならないためです。

John Smith
123 Main St
Apt 5
05674

住所行に混在する余分なデータ

住所行に混在する余分なデータは、AdditionalInputData に返されます。例えば、以下の住所において、"John Smith" は AdditionalInputData に返されます。

123 Main St John Smith
05674

123 Main St Apt 5 John Smith
05674

123 Main St John Smith
Apt 5
05674

123 Main St
Apt 5 John Smith
05674

米国住所に対しては、住所行の末尾に存在する余分なデータのみが AdditionalInputData に返されます。米国住所において、住所行の末尾以外に存在する余分なデータは返されません。例えば、以下の住所において、"John Smith" は返されません。

John Smith 123 Main St
05674

123 Main John Smith St
05674

マッチさせるためにストリート名が変更され、ストリート名または接尾語が行の末尾にあった場合は、AdditionalInputData 列には、元のストリート名や接尾語が含まれることがあります。例えば、次の住所があるとします。

Pitney Bowes
4200 Parlament
Lanham MD

ValidateAddress は、ストリート名の綴りを修正し、接尾語を追加して、修正済みのストリートの住所として "4200 Parliament Pl" を返し、"Parlament" を AdditionalInputData に返します。

二重住所

二重住所とは、ストリート情報と、PO Box/地方配送路/Highway Contract 情報の両方を含む住所のことです。選択した処理オプションに応じて、住所の正規化に使用されない二重住所の一部が、AdditionalInputData に返される場合があります。詳細については、**二重住所ロジックについて**（422ページ）を参照してください。

ValidateAddressAUS

ValidateAddressAUS は、オーストラリア郵便公社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。また、郵便番号、都市名、州/準州名など、欠落している郵便情報を追加します。

ValidateAddressAUS は、ValidateAddressAUS が住所の妥当性を確認したかどうかや、住所の妥当性が確認できなかった場合はその理由など、バリデーション処理に関する結果インジケータも返します。

ValidateAddressAUS 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。

ValidateAddressAUS は、Universal Addressing モジュールに含まれています。

入力

ValidateAddressAUS は、入力として標準住所を受け取ります。すべての住所がこのフォーマットを使用します。

表 101：入力フォーマット

フィールド名	書式	説明
AddressLine1	文字列 [288]	最初の住所行。
AddressLine2	文字列 [288]	2 行目の住所行。
AddressLine3	文字列 [288]	3 行目の住所行。
AddressLine4	文字列 [288]	4 行目の住所行。
City	文字列 [48]	都市/地方/郊外の名前。これは、省や郵便番号とともに、AddressLine フィールドのいずれかにオプションで入力できます。
StateProvince	文字列 [4]	州。これは、都市や郵便番号とともに、AddressLine フィールドのいずれかにオプションで入力できます。
PostalCode	文字列 [8]	郵便番号。これは、州や都市とともに、AddressLine フィールドのいずれかにオプションで入力できます。

オプション

ValidateAddressAUS には、住所の処理方法と返す情報のタイプを制御するオプションがいくつかあります。

表 102 : オプション

オプション名	説明/有効値
Database	国際住所処理に使用するデータベースを指定します。指定できるのは、Management Console の **[Australia Database リソース]** パネルで定義されたデータベースに限られます。
OutputFieldLevelReturnCodes	特定の出力要素に関連付けられている結果フィールドを出力します。結果コード（496ページ） を参照してください。 有効な値は、次のとおりです。 **N** 出力に個々のフィールドの結果コードを含めません (デフォルト)。 **Y** 出力に個々のフィールドの結果コードを含めます。
OutputOriginalInputFields	元の入力データを返します。元の入力データ（498ページ） を参照してください。 有効な値は、次のとおりです。 **N** 出力に元の入力データを含めません (デフォルト)。 **Y** 出力に元の入力データを含めます。
OutputMatchedAddressFields	パース済み住所要素を返します。パース済み住所要素（497ページ） を参照してください。 有効な値は、次のとおりです。 **N** 出力にパース済み住所要素を含めません (デフォルト)。 **Y** 出力にパース済み住所要素を含めます。

オプション名	説明/有効値
AmasFormatting	Address Matching Approval System (AMAS) 表記を使用して出力住所データをフォーマットすることを指定します。 このオプションを使用すると、Validate Address AUS は、住所を正規化するときに AMAS ルールを使用するようになります。AMAS は、オーストラリア郵政公社が定める、住所規格を徹底するためのプログラムです。AMAS 書式設定表記の詳細については、『Address Matching Approval System (AMAS) Handbook』を参照してください。 このオプションを使用すると、出力データは次のように変更されます。 • 数値フィールドにはゼロが付加されます。この影響を受けるのは、HouseNumber、HouseNumber2、PostalDeliveryNumber、および DPID の各出力フィールドです。例えば、入力フィールドが 298 New South Head Rd Double Bay NSW 2028 の場合、HouseNumber フィールドの形式は 298 から 00298 に変更されます。 • 一致しない場合、DPID フィールドの桁はすべてゼロになります。例えば、00000000 などです。 • 一致しない場合、すべてゼロを含む数値フィールドを除き、すべてのリターンフィールド (パース済み住所要素) が空白になります。 • CCD フィールドは出力されません。 有効な値は、次のとおりです。 **N** AMAS 表記を使用して出力データをフォーマットしません (デフォルト)。 **Y** AMAS 表記を使用して出力データをフォーマットします。

出力

最小限、ValidateAddressAUS の出力は、**標準出力フィールド**（494ページ）に示す標準出力フィールドで構成されます。これらの標準フィールドに加えて、出力には、選択した出力オプションに応じて他の情報も含まれることがあります。オプションの出力フィールドの詳細については、**結果コード**（496ページ）、**パース済み住所要素**（497ページ）、および**元の入力データ**（498ページ）を参照してください。

標準出力フィールド

次の表に、ValidateAddressAUS が出力する標準フィールドを示します。

表 103 : 出力フィールド

フィールド名	説明
AddressLine1	フォーマット済みの住所行。
BuildingName	建物名。
City	都市/地方/郊外の名前 1。
City2	都市/地方/郊外の名前 2 - 分割された名前 (VIA など)。
StateProvince	州。
PostalCode	郵便番号。
CCD	Census 収集区。Census データのコレクション、処理、および出力用の地理的な基本単位。一般的に、CCD あたり約 200 ～ 250 の世帯と、オーストラリア全体で約 37,000 の CCD があります。
DPID	配達ポイント識別子。ストリート住所などの郵便物配達ポイントを一意に識別する 8 桁の数字。オーストラリア郵政公社郵便住所ファイルに規定されています。
Status	マッチング試行の成功または失敗。 **F** 失敗 (DPID または CCD が見つからない) **NULL** 成功
Status.Code	失敗の原因 (ある場合)。 • UnableToValidate • InsufficientInputData
Status.Description	問題の説明 (ある場合)。

フィールド名	説明
AMAS.ResultCode	基本エンジンが返す結果コード。
AMAS.ResultMessage	基本エンジンが返す任意の結果メッセージ。

結果コード

このオプションは、各結果フィールドの結果コードのほか (該当する場合)、特定の出力要素に関連付けられた結果フィールドを出力します。結果フィールドに付属の結果コードが含まれていない場合は、次のいずれかを示していると考えられます。

- パース済み要素に対して変更は行われていない。
- パース済み要素が正規化された (例えば、'Street' が 'ST' に変更された)。
- 対応するパース済み住所要素に対してデータはパースされていない。

表 104 : 結果コード

フィールド名	結果コード	
City.Result	**C**	修正済み
HouseNumber.Result	**U**	マッチしない、欠落、あいまい
PostalCode.Result	**C**	修正済み
PostalDelivery.Result	**C**	修正済み
	D	ドロップ済み
	U	マッチしない
StateProvince.Result	**C**	修正済み

フィールド名	結果コード	
StreetName.Result	**C**	修正済み
	U	マッチしない、欠落、あいまい
StreetSuffix.Result	**C**	修正済み

パース済み住所要素

このオプションは、パース済み住所要素を出力します。

表 105 : パース済み住所要素

フィールド名	説明
ApartmentLabel	フラットまたはユニット タイプ (STE や APT など)。例: 123 E Main St **Apt** 3
ApartmentNumber	フラットまたはユニット番号。例: 123 E Main St Apt **3**
FloorLabel	フロア/レベル タイプ。例: 123 E Main St Apt 3, 4th **Floor**
FloorNumber	フロア/レベル番号。例: 123 E Main St Apt 3, **4th** Floor
LotNumber	敷地番号。例: Lot 7 Caldwell Hwy
PostalDeliveryLabel	郵便配達タイプ。例: PO Box 42
PostalDeliveryNumber	郵便配達番号。例: PO Box 42
PostalDeliveryPrefix	郵便配達番号接頭語。例: PO Box A42

フィールド名	説明
PostalDeliverySuffix	郵便配達番号接尾語。例: PO Box 42B
HouseNumber	家番号 1。例: 298A-1B New South Head Rd
HouseSuffix	家番号 1 接尾語。例: 298A-1B New South Head Rd
HouseNumber2	家番号 2。例: 298A-1B New South Head Rd
HouseSuffix2	家番号 2 接尾語。例: 298A-1B New South Head Rd
StreetName	建物が存在するストリートの名前。例: 123 E **Main** St Apt 3
StreetSuffix	ストリート接尾語。例: 123 E Main **St** Apt 3
TrailingDirectional	接尾方向指示。例: 123 Pennsylvania Ave **NW**

元の入力データ

このオプションは、元の入力データを <フィールド名>.Input フィールドに出力します。

表 106 : 入力データ

フィールド名	説明
AddressLine1.Input	入力に渡される 1 番目の住所行。
AddressLine2.Input	入力に渡される 2 番目の住所行。

フィールド名	説明
AddressLine3.Input	入力に渡される 3 番目の住所行。
AddressLine4.Input	入力に渡される 4 番目の住所行。
City.Input	入力に渡される都市/地方/郊外の名前。
StateProvince.Input	入力に渡される州。
PostalCode.Input	入力に渡される郵便番号。

ValidateAddressGlobal

ValidateAddressGlobal は、米国およびカナダ以外の住所のパフォーマンスのみを制御します。ValidateAddressGlobal 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、ValidateAddressGlobal の使用を検討してください。

ValidateAddressGlobal は Universal Addressing モジュールの一部です。

ValidateAddressGlobal は、書き直し、パーシング、バリデーション、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。

文字セットのマッピングと書き直し

ValidateAddressGlobal は他国の文字列と、それらの複雑な問題を処理します。Unicode に完全対応の文字列処理を使用するため、アルファベット以外の文字をラテン文字セットに書き直したり、異なる文字セット間でマッピングしたりできます。

文字セットのマッピング、および書き直しについて、以下の機能があります。

- UTF-8、 ISO 8859-1、GBK、BIG5、JIS、EBCDIC など、30 以上の文字セットをサポート
- 言語のルールに従って、付加記号を正しく "除去"
- さまざまなアルファベットをラテン スクリプトに書き直し
- ギリシャ文字 (BGN/PCGN 1962、ISO 843 - 1997)

- キリル文字 (BGN/PCGN 1947、ISO 9 - 1995)
- ヘブライ語
- 日本語の片仮名、平仮名、漢字
- 中国語のピンイン (標準中国語、 広東語)
- 韓国語のハングル文字

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、(企業や個人名をストリート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。ValidateAddressGlobal 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 検証

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。ValidateAddressGlobal 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。ValidateAddressGlobal 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。

入力

ValidateAddressGlobal は、入力として標準住所を受け取ります。どの国の住所であるかにかかわらず、すべての住所がこのフォーマットを使用します。

表 107 : ValidateAddressGlobal の入力

columnName	書式	説明
AddressLine1 から AddressLine6	文字列	これらのフィールドには住所行データが格納されます。AddressLine1 には最初の住所行、AddressLine2 には 2 行目の住所行が格納されます。以降もこれと同様になります。都市、州/省、および郵便番号情報は、住所行フィールドではなく、それぞれのフィールドに配置する必要があります。例: **AddressLine1:** 17413 Blodgett Road **AddressLine2:** PO Box 123 **City:** Mount Vernon **StateProvice:** WA **PostalCode:** 97273 **Country:** USA 入力住所が適切な住所行および City、StateProvice、PostalCode フィールドにまだパースされていない場合は、住所行フィールドの代わりに UnformattedLine フィールドを使用してください。
City	文字列	都市名
StateProvince	文字列	州または省。
PostalCode	文字列 99999 99999-9999 A9A9A9 A9A 9A9 9999 999	住所の郵便番号。米国では、ZIP Code®になります。
Contact	文字列	受取人の名前。例えば、"Mr.Jones"。
Country	文字列	国名。`Input.ForceCountryISO3` または `Input.DefaultCountryISO3` オプションに値が指定されていない場合、国を指定する必要があります。
FirmName	文字列	会社名または企業名。

columnName	書式	説明
Street	文字列	ストリート
Number	Building [79]	Number
Building	文字列	Building
SubBuilding	文字列	従属する建物
DeliveryService	文字列	配送サービス
UnformattedLine1 から UnformattedLine10	文字列	入力住所が完全にパースされていない場合、かつ ValidateAddressGlobal によって住所を適切なフィールドにパースしたい場合は、このフィールドを使用してください。例: **UnformattedLine1:** 17413 Blodgett Road **UnformattedLine2:** PO Box 123 **UnformattedLine3:** Mount Vernon WA 97273 **UnformattedLine4:** USA この住所は、以下の出力フィールドにパーシングされます。 **AddressLine1:** 17413 Blodgett Road **AddressLine2:** PO Box 123 **City:** Mount Vernon **StateProvice:** WA **PostalCode:** 97273 **Country:** USA **注 :** フォーマットされていない行フィールドへの入力を指定した場合は、住所全体をフォーマットされていない行フィールドだけを使用して指定する必要があります。City や StateProvince など、その他のフィールドを、フォーマットされていない行フィールドと合わせて使用することはできません。

オプション

入力オプション

表 108 : ValidateAddressGlobal の入力オプション

オプション名	説明/有効値
Database.AddressGlobal	住所検証で使用する郵便データを含むデータベース リソースを指定します。指定できるのは、Management Console の **[グローバル データベース リソース]** パネルで定義されたデータベースに限られます。詳細については、『*Spectrum™ Technology Platform 管理ガイド*』を参照してください。
Input.DefaultCountryISO3	入力レコードに明示的な国情報が含まれない場合に使用するデフォルト国を指定します。指定する際、ISO3 の国コードを使用してください。デフォルト国を指定しない場合、各入力レコードの Country 入力フィールドに国を指定する必要があります。ISO コードの一覧は、「**ISO 国コードとモジュール サポート**（598ページ）」を参照してください。
Input.ForceCountryISO3	住所レコードが、常にここで指定された国から発送されたものとして扱われるようにします。住所レコードの国やデフォルト国は上書きされます。指定する際、ISO3 の国コードを使用してください。ISO コードの一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
Input.FormatDelimiter	入力ファイルで複数行の住所に標準以外の書式設定を使用できます。このフィールドで指定できる値を次に示します。 • CRLF (デフォルト) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW ; WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW , WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • PIPE (2101 MASSACHUSETTS AVE NW | WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) **注 :** 入力オプションと出力オプションの両方に同じ値を選択する必要があります。

出力オプション

表 109 : ValidateAddressGlobal の出力オプション

optionName	説明
Result.MaximumResults	このオプションには、返される候補住所の最大数を指定します。このフィールドは、バッチ処理には無効です。それ以外の処理モードの場合、デフォルト値は 1、最大値は 99 です。FastCompletion モードを使用している場合は、1 よりも大きい数を入力して、フィールド入力のための選択肢が複数返されるようにするとよいでしょう。
Result.IncludeInputs	出力の中に、入力データを含めるかどうかを指定します。有効にすると、.Input で終わるフィールド (対応する入力フィールドを含む) が出力に含まれます。例えば、出力フィールドの AddressLine1.Input に、入力フィールドの AddressLine1 に指定したデータが含まれます。 **TRUE** 入力データを出力に含めます。 **FALSE** 出力に元の入力データを含めません (デフォルト)。
Result.StateProvinceType	StateProvince フィールドのフォーマットを指定します。次のいずれかを選択します。 **ABBREVIATION** 州または省の省略形を返します。例えば、North Carolina であれば "NC" が返ります。 **COUNTRY_STANDARD** その国の郵便当局で使用されるフォーマットに応じて、省略形または完全な名前が返ります（デフォルト）。 **EXTENDED** 州または省の (省略形ではなく) 完全な名前が返ります (例えば、"North Carolina" など)。

optionName	説明	
Result.CountryType	ValidateAddressGlobal から返される国名で使用する言語またはコードを指定します。	
	ISO2	その国の 2 文字の ISO コード
	ISO3	その国の 3 文字の ISO コード
	ISO_NUMBER	ISO 国番号
	NAME_CN	中国語
	NAME_DA	デンマーク語
	NAME_DE	ドイツ語
	NAME_EN	英語 (デフォルト)
	NAME_ES	スペイン語
	NAME_FI	フィンランド語
	NAME_FR	フランス語
	NAME_GR	ギリシャ文字
	NAME_HU	ハンガリー語
	NAME_IT	イタリア語
	NAME_JP	日本語
	NAME_KR	韓国語
	NAME_NL	オランダ語
	NAME_PL	ポーランド語
	NAME_PT	ポルトガル語
	NAME_RU	ロシア語
	NAME_SA	サンスクリット語
	NAME_SE	スウェーデン語

optionName	説明
Result.PreferredScript	出力がどのアルファベットで返されるかを指定します。データがどのアルファベットで返されるかは、国によって異なります。言語の設定で何を選択したかに関わらず、ほとんどの国で出力は Latin I になります。 **ASCII_Extended** 特殊文字 (Ã– = OE など) で拡張された ASCII 文字 **ASCII_Simplified** ASCII 文字 **Database** (デフォルト) Latin I または ASCII 文字 (参照データベースの標準による) **Latin** Latin I 文字 **Latin_Alt** Latin I 文字 (代替の書き直し) **Postal_Admin_Alt** Latin I または ASCII 文字 (現地郵便局の代替) **Postal_Admin_Pref** Latin I または ASCII 文字 (現地郵便局の選択による) Latin I 以外のアルファベットを使用する国の場合、返されるアルファベットは国によって異なります。詳細については、**Latin 1 以外のアルファベットを使用する国**（507ページ）を参照してください。
Result.PreferredLanguage	出力がどの言語で返されるかを指定します。データがどのアルファベットで返されるかは、国によって異なりますが、ほとんどの国で、言語の設定に何を選択したかに関わらず、出力は Latin I になります。 **DATABASE** 各住所の参照データから得られた言語で返されます。こちらがデフォルトです。 **ENGLISH** 地方および州/省の名前が (可能であれば) 英語で出力されます。
Result.Casing	出力の大文字と小文字の区別を指定します。 **NATIVE** 参照データベースの標準に基づいて出力されます。 **UPPER** すべての国で、大文字で出力されます。 **LOWER** すべての国で、小文字で出力されます。 **MIXED** 国固有のルールに従って大文字と小文字が判断されます。 **NOCHANGE** パース モードの場合、データは入力された方法で返されます。バリデーション モードの場合、大文字と小文字の区別には参照データに見つかったものや、郵便のルールに従ったものを使用します。参照データを使用してチェックできなかった値は、入力時の大文字と小文字の区別を保持します。

optionName	説明
Result.FormatDelimiter	出力で複数行の住所に標準以外の書式設定を使用できます。このフィールドで指定できる値を次に示します。 • CRLF (デフォルト) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW ; WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW , WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • PIPE (2101 MASSACHUSETTS AVE NW | WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) **注：** 入力オプションと出力オプションの両方に同じ値を選択する必要があります。

Latin 1 以外のアルファベットを使用する国

Latin I 以外のアルファベットを使用する国の場合、返されるアルファベットは国によって異なります。それらの国々で、出力がどのように返されるかを以下の表に示します。ここに含まれないすべての国では、 フィールド `Result.PreferredScript` オプションに指定された値が使用されます。

Country	データベース	Postal_Admin_Pref	Postal_Admin_Alt	ラテン文字	Latin_Alt	ASCII_Simplified	ASCII_Extended
RUS	キリル文字	キリル文字	キリル文字	CYRILLIC_ISO	CYRILLIC_BGN	CYRILLIC_ISO + LATIN_SIMPLE	CYRILLIC_ISO + LATIN
JPN	漢字	漢字	かな	JAPANESE	JAPANESE	JAPANESE + LATIN_SIMPLE	JAPANESE + LATIN
CHN	Hanzi	Hanzi	Hanzi	CHINESE_ MANDARIN	CHINESE_ CANTONESE	CHINESE_ MANDARIN + LATIN_SIMPLE	CHINESE_ MANDARIN + LATIN
HKG	Hanzi	Hanzi	Hanzi	CHINESE_ CANTONESE	CHINESE_ MANDARIN	CHINESE_ CANTONESE + LATIN_SIMPLE	CHINESE_ CANTONESE + LATIN

Country	データベース	PostalAdminPref	PostalAdminAlt	ラテン文字	Latin_Alt	ASCII_Simplified	ASCII_Extended
TWN	Hanzi	Hanzi	Hanzi	CHINESE_CANTONESE	CHINESE_MANDARIN	CHINESE_CANTONESE + LATIN_SIMPLE	CHINESE_CANTONESE + LATIN
GRC	ギリシャ文字	ギリシャ文字	ギリシャ文字	GREEK_ISO	GREEK_BGN	GREEK_ISO + LATIN_SIMPLE	GREEK_ISO + LATIN
KOR	ラテン文字	ハングル	Hanja	KOREAN	KOREAN	KOREAN + LATIN_SIMPLE	KOREAN + LATIN
ISR	ラテン文字	ヘブライ語	ヘブライ語	HEBREW	HEBREW	HEBREW + LATIN_SIMPLE	HEBREW + LATIN
ROM	Latin-3	Latin-3	Latin-3	Latin-3	Latin-3	LATIN_SIMPLE	LATIN
POL	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CZE	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CRI	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
HUN	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
MDA	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
SVK	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
LAT	Latin-7	Latin-7	Latin-7	Latin-7	Latin-7	LATIN_SIMPLE	LATIN

プロセス オプション

表 110 : ValidateAddressGlobal のプロセス オプション

optionName	説明
Process.OptimizationLevel	このオプションを使用して、処理速度と品質が適切なバランスを保つよう設定します。次のいずれかです。 **NARROW** パーサーは、House Number を Street の情報から分割することを除き、入力の割り当てに厳密に従います。 **STANDARD** パーサーは、以下に示すように住所要素をよりアクティブに分割します。 • Province は Locality の情報から分割される • PostalCode は Locality の情報から分割される • House Number は Street の情報から分割される • SubBuilding は Street の情報から分割される • DeliveryService は Street の情報から分割される • SubBuilding は Building の情報から分割される • Locality は PostalCode の情報から分割される **WIDE** パーサーによる分割はStandardに似た方法で行われますが、追加で最大10のパーシング候補が検証に渡されて処理されます。検証は検索のツリーを広げ、参照データのエントリを対象に追加してマッチングを行います。 最適化レベルを調整しても、ここで説明したような分割で必要とされる郵便参照データ情報が揃っていない国においては、効果が得られない場合があります。 分類の精度を Narrowから Standard に増やしても処理パワーはある程度消費されますが、最適化レベルを Wideに設定し、検索ツリーを大きくして検証を行った場合、指定された入力データから最も精密な結果を得ようとデータのアクセスや比較の件数が増加するため、処理速度が非常に遅くなります。

optionName	説明
Process.Mode	住所に対して実行する処理のタイプを指定します。次のいずれかです。 **BATCH** このモードは手動でのデータの入力または選択が不可能なバッチ処理環境で使用します。処理速度を重視して最適化されているため、自動で修正できないあいまいなデータが見つかった場合は、住所修正の試行が停止します。指定の国がデータベースに見つからない場合、バッチ処理モードはパース モードに変更されます。 **注 :** プロセス ステータスとして I3 の値が返された場合、試行は失敗とみなされ、ステータスとして F の値が返されます。 **CERTIFIED** このモードはオーストラリア郵便当局のバッチ処理環境で使用します。Validate Address Global はオーストラリア郵便公社の Address Matching Approval System (AMAS) で認定されています。これは郵便住所ファイルに照らして郵便の正規化と検証を行うことで、郵便料金の割引と不達郵便物の最小化を実現します。 **FASTCOMPLETION** 高速実行モードは、切り捨て処理されたデータを住所フィールドに入力して、Validate Address Global による提案の生成を行いたい場合に使用します。例えば、コール センターや店頭などで作業をしている場合に住所要素の一部だけを入力し、高速実行の機能を使用して、完全な住所の候補を正しく入手することができます。 **INTERACTIVE** このモードは、インタラクティブな環境で処理を行い、住所入力があいまいな場合に提案を生成する場合に使用します。このタイプの検証は、とりわけ顧客や潜在顧客から入手したデータを入力した環境で使用されます。住所がほぼ完全に入力されている必要があり、その入力データの検証や修正を試行します。あいまいなデータが検出された場合、この検証 タイプでは最大で 20 の提案が生成され、それを選択候補の一覧として使用できます。指定の国がデータベースに見つからない場合、インタラクティブ モードはパース モードに変更されます。 **PARSE** このモードは住所入力をトークンに分割し、そのあとに他のシステムで行われる処理に送る場合、すなわち検証をバイパスする場合に使用します。例えば、住所データの品質がすでに高く、単に素早くトークンに分割して外部のシステムにエクスポートするか、あるいは下流のステージで使用する場合にこのモードを利用できます。

optionName	説明
Process.MatchingScope	住所が検証されるために、住所がどの程度参照データと一致する必要があるかを指定します。次のいずれかです。 **注：** 以下の設定は、必要とされる詳細情報が郵便参照データに見つからない国に対しては効果がない場合があります。 **ALL** すべての住所要素が一致する必要があります。 **DELIVERYPOINT_LEVEL** Validate Global Address で州/省、郵便番号、都市/地方/郊外、ストリート、家番号、および従属する建物が一致する必要があります。 **STREET_LEVEL** Validate Global Address で州/省、郵便番号、都市/地方/郊外、およびストリートが一致する必要があります。 **LOCALITY_LEVEL** Validate Global Address で州/省、郵便番号、および都市/地方/郊外が一致する必要があります。

出力

住所データ

表 111：パース済み住所要素

columnName	説明
AddressBlock1-9	AddressBlock 出力フィールドには、正規化済み、または標準化済みの住所が、実際の郵便物に印刷される形式にフォーマットされて入ります。Validate Address Global は住所を郵便当局の規格に従って住所ブロックにフォーマットします。住所の各行が、別々の住所ブロック フィールドに入れて返されます。AddressBlock1 から AddressBlock9 まで、最大 9 つの住所ブロック出力フィールドが使用されます。例えば、以下の住所入力の場合、 AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 以下の住所ブロックが出力されます。 AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882

columnName	説明
AddressLine1-6	住所が検証された場合、住所行フィールドには検証済みで正規化済みの住所行が入ります。住所が検証できなかった場合、住所行フィールドには入力された住所が変更されずに入ります。ただし、住所の最後の行は LastLine フィールドに入ります。例: AddressLine1: 4200 PARLIAMENT PL STE 600 LastLine: LANHAM MD 20706-1882
AdministrativeDistrict	州/省より小さいが、都市よりも大きいエリア
ApartmentLabel	フラットまたはユニット タイプ (STE や APT など)。例: 123 E Main St **Apt** 3
ApartmentNumber	フラットまたはユニット番号。例: 123 E Main St Apt **3**
BlockName	地所やブロックの名前。
BuildingName	建物の名前。例えば Sears Tower など。
City	都市の名前。例えば **Vancouver**, BC など。
City.AddInfo	都市の追加情報。
City.SortingCode	例えばプラハやダブリンなど、特定の国の広い地方で、郵便当局が配達時間を短縮するために使用するコード。
Contact	受取人の名前。例えば **Mr.Jones** など。
Country	国が、言語または`Result.CountryType`オプションで指定されたコードで入ります。
County	州や省に従属する情報。州や省をさらに細かく分割します。米国の郡はその一例です。
FirmName	会社名。

columnName	説明
Floor	部屋番号やアパート番号など、建物をさらに細かく分割する情報。例: 123 E Main St Apt 3, **4th Floor**
HouseNumber	家番号 1。例: 298A-1B New South Head Rd
LastLine	完成された最終の住所行 (都市、州/省、および郵便番号)。
LeadingDirectional	ストリート名の前に付けてストリートの方向を表します。例えば、138 N Main Street の N がこれに該当します。
Locality	場所の名前に従属し、地方をさらに細かく分割します。例としては、メキシコの Colonia や、スペインの Urbanisaciones などがあります。
POBox	郵便受けの記述子 (POBox、Postfach、Case Postale など) と番号。
PostalCode	住所の郵便番号。郵便番号のフォーマットは国によって異なります。
PostalCode.AddOn	郵便番号の 2 番目の部分。例えば、カナダの住所ではこれは LDU です。米国住所に対しては、これは ZIP + 4 アドオンです。ほとんどの国ではこのフィールドを使用しません。
PostalCode.Base	郵便番号の基本部分。
Room	建物の部屋番号。
SecondaryStreet	補助的なストリートまたは地方集配路の名前。
StateProvince	州または省の名前。
StreetName	建物が存在するストリートの名前。例: 123 E **Main** St Apt 3
StreetSuffix	ストリート接尾語。例: 123 E Main **St** Apt 3
SubBuilding	部屋番号など建物の一部。例えば Suite 102 など。

columnName	説明
Suburb	場所の名前に従属し、地方をさらに細かく分割します。例えばトルコの Mahalle など。
Territory	地域の名前。地域は州/省より大きいものです。
TrailingDirectional	接尾方向指示。例: 123 Pennsylvania Ave **NW**

元の入力データ

このオプションは、元の入力データを <フィールド名>.Input フィールドに出力します。

表 112 : 元の入力データ

columnName	書式	説明
AddressLine1.Input	文字列	最初の住所行。
AddressLine2.Input	文字列	2 行目の住所行。
AddressLine3.Input	文字列	3 行目の住所行。
AddressLine4.Input	文字列	4 行目の住所行。
AddressLine5.Input	文字列	5 行目の住所行。
AddressLine6.Input	文字列	6 行目の住所行。
City.Input	文字列	都市名
StateProvince.Input	文字列	州または省

columnName	書式	説明
PostalCode.Input	文字列	住所の郵便番号。米国では、ZIP Code になります。次のいずれかのフォーマットです。 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
Contact.Input	文字列	受取人の名前。例えば、"Mr.Jones"。
Country.Input	文字列	入力の国フォーマットに選択したフォーマットを使用して国を指定します (英語名、ISO コード、または UPU コード)。有効な値の一覧は、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName.Input	文字列	会社名または企業名。
Street.Input	文字列	ストリート
Number.Input	Building [79]	Number
Building.Input	文字列	Building
SubBuilding.Input	文字列	従属する建物
DeliveryService.Input	文字列	配送サービス

結果コード

これらの出力フィールドには、検証処理の結果に関する情報が格納されます。

表 113 : 結果コード

フィールド名	結果コード
AddressType	米国とカナダの住所の場合のみ、AddressType フィールドは住所のタイプを示します。次のいずれかです。 **F** 住所の妥当性が確認され社名まで修正されました。 **B** 住所の妥当性が確認され建物名まで修正されました。 **G** 住所は局留めの住所です。 **H** 住所の妥当性が確認され高層のデフォルトまで修正されました。 **L** 住所は、Large Volume Receiver (LVR) です。 **M** 住所は、軍施設の住所です。 **P** 住所の妥当性が確認され私書箱まで修正されました。 **R** 住所の妥当性が確認され地方配送路まで修正されました。 **S** 住所の妥当性が確認されストリートの住所まで修正されました。 **U** タイプが不明なために住所を検証/修正できませんでした。
Confidence	返された住所に割り当てられた確信レベル。範囲は 0 ～ 100 です。0 は失敗を表し、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表します。
CountOverflow	候補住所数が返される数より多いかどうかを示します。次のいずれかです。 **はい** 他の候補住所があります。他の候補を取得するには、`MaximumResults` の値を増やします。 **いいえ** いいえ、これ以外の候補はありません。
ElementInputStatus	ElementInputStatus は入力要素を参照データに対してマッチングした際の要素ごとの情報を提供します。このフィールドの値は、バッチとパースのどちらのモードを使用するかによって変わります。このフィールドの値については、**ElementInputStatus、ElementResultStatus、および ElementRelevance の解釈**（521ページ）を参照してください。
ElementRelevance	その土地の郵便当局の基準から基準から判断して、実際にどの住所要素が関連しているかを示します。このフィールドの値については、**ElementInputStatus、ElementResultStatus、および ElementRelevance の解釈**（521ページ）を参照してください。

フィールド名	結果コード
ElementResultStatus	ElementResultStatus は、出力フィールドが入力フィールドから変更されているか、またどのように変わっているかを示すことによって、結果を ProcessStatus フィールドよりも詳細に分類します。このフィールドの値については、**ElementInputStatus、ElementResultStatus、および ElementRelevance の解釈**（521ページ）を参照してください。
MailabilityScore	郵便が住所にどの程度確実に配達されるかの予測を示します。次のいずれかです。 **5** 確実に配達される **4** ほぼ確実に配達される **3** おそらく配達される **2** 五分五分 **1** 配達されない可能性が高い **0** 配達されない
ModeUsed	使用された処理モードを示します。処理モードは、`Process.Mode` オプションで指定します。モードの説明については、**プロセス オプション**（509ページ）を参照してください。
MultimatchCount	住所が参照データ内の複数の候補住所と一致した場合、このフィールドには一致した候補の数が含まれます。

フィールド名	結果コード
ProcessStatus	

フィールド名	結果コード

出力の品質に関する一般的な記述が含まれます。出力の品質について、詳しくは ElementResultStatus フィールドを参照してください。

次のいずれかです。

V4 検証済み。正しい入力データです。すべての要素がチェックされ、入力データが完全に一致しました。

V3 検証済み。入力されたデータは正しいものですが、一部またはすべての要素が正規化されたか、あるいは入力データに旧式の名前または外名 (エクソニム) が含まれます。

V2 検証済み。正しい入力データですが、一部の要素について、参照データが完全でないために検証できていません。

V1 検証済み。正しい入力データですが、ユーザによる正規化によって、正しく配達される可能性が損なわれています(たとえば選択された郵便番号の長さが短すぎるなど、ユーザによって要素の正規化が正しく行われていない)。検証によってセットされません。

C4 修正済み。すべての要素がチェック済みです。

C3 修正済みですが、一部の要素をチェックできませんでした。

C2 修正済みですが、配達のステータスがわかりません (参照データがありません)。

C1 修正済みですが、ユーザによる正規化が正しくなかったため、配達のステータスがわかりません。検証によってセットされません。

I4 データを完全には修正できませんでしたが、かなりの確率で正しく配達されます。単一の一致が見つかりました (例えば HNO は誤りだが、参照データに HNO が 1 件だけ見つかるような場合)。

I3 データを完全には修正できませんでしたが、かなりの確率で正しく配達されます。複数の一致が見つかりました (例えば HNO は誤りだが、参照データに複数の HNO が見つかるような場合)。

I2 データを修正できませんでしたが、この住所で正しく配達される可能性がわずかにあります。

I1 データを修正することができず、正しく配達される可能性もほとんどありません。

RA 国が強制国の設定から認識されました。

R9 国が DefaultCountryISO3 の設定から認識されました。

R8 国が名前からエラーなしで認識されました。

R7 国が名前からエラー付きで認識されました。

R6

フィールド名	結果コード
	国が地域から認識されました。
R5	国が省から認識されました。
R4	国が主要な都市から認識されました。
R3	国がフォーマットから認識されました。
R2	国がスクリプトから認識されました。
R1	国が認識されませんでした。複数の一致が見つかりました。
R0	国が認識されませんでした。
S4	完全にパースされました。
S3	パースされ、複数の結果が得られました。
S2	パースされ、エラーが発生しました。要素の位置が変わっています。
S1	パース エラー。入力フォーマットが一致しません。
N1	バリデーション エラー: 国が認識されなかったため、検証が実行されませんでした。
N2	バリデーション エラー: 必要な参照データベースを使用できないため、検証が実行されませんでした。
N3	バリデーション エラー: 国をロック解除できなかったため、検証が実行されませんでした。
N4	バリデーション エラー: 参照データベースが破損しているか、フォーマットが正しくないため、検証が実行されませんでした。
N5	バリデーション エラー: 参照データベースが古すぎるため、検証が実行されませんでした。
N6	バリデーション エラー: 入力データが十分でないため、検証が実行されませんでした。
Q3	高速実行ステータス: 提案 (完全な住所) を利用可能です。
Q2	高速実行ステータス: 完全な住所が提案されましたが、入力データの要素と組み合わせられています (追加または削除)。
Q1	高速実行ステータス: 提案された住所は完全ではありません (情報を追加してください)。
Q0	高速実行ステータス: 十分でない情報から提案が生成されました。

フィールド名	結果コード	
Status	処理試行が成功したか失敗したかをレポートします。	
	NULL	成功
	F	失敗
Status.Code	失敗したものがあれば、その理由を示します。	
Status.Description	失敗したものがあれば、その理由を説明する記述が入ります。	

ElementInputStatus、 ElementResultStatus、および ElementRelevance の解釈

ElementInputStatus、ElementResultStatus、および ElementRelevance の出力フィールドには、検証操作の結果を詳細に示す一連の数字が含まれます。ElementInputStatus にはパーシング操作に関するいくつかの情報が含まれます。

ElementInputStatus の値は以下のようになります。

44606040600000000060

ElementResultStatus の値は以下のようになります。

88F0F870F00000000040

ElementRelevance の値は以下のようになります。

11101010100000000000

これらのフィールドの値について理解するためには、各ポジションの数字がどの要素を表すか、および各ポジションの値の意味を知る必要があります。たとえば、先頭の数字は PostalCode.Base 出力フィールドの結果を示します。以下に各ポジションの意味を一覧で示します。

- ポジション 1—PostalCode.Base
- ポジション 2—PostalCode.AddOn
- ポジション 3—City
- ポジション 4—Locality および Suburb
- ポジション 5—StateProvice
- ポジション 6—County
- ポジション 7—StreetName
- ポジション 8—SecondaryStreet
- ポジション 9—HouseNumber
- ポジション 10—Number レベル 1

- ポジション 11—POBox
- ポジション 12—Delivery サービス レベル 1
- ポジション 13—Building レベル 0
- ポジション 14—BuildingName
- ポジション 15—Sub building レベル 0
- ポジション 16—Floor and Room
- ポジション 17—FirmName
- ポジション 18—Organization レベル 1
- ポジション 19—Country
- ポジション 20—Territory

ElementInputStatus の場合、検証の値として以下のいずれかが入ります。

- 0—空
- 1—見つからない
- 2—チェックなし (参照データなし)
- 3—誤り - 検証によってのみセット。参照データベースによれば、Number または DeliveryService のいずれかの数が正しい範囲内にない。入力はコピーされ、バッチモードでは修正されていない。インタラクティブ モードと FastCompletion では、提案が提供されている
- 4—この要素内でエラーありで一致
- 5—変更ありで一致 (挿入または削除)。例:
  - パーシング: "MainSt 1" の家番号を分割するなど
  - バリデーション: 外名 (エクソニム) が入力された場合に置換したり、フィールドのサイズを超えて入力され、国の参照データベースによって無効とされたデータを短くしたなど
- 6—エラーなしで一致

ElementInputStatus の場合、パーシングの値には以下のいずれかが入ります。

- 0—空
- 1—要素の位置を変更する必要があった
- 2—一致したが、正規化する必要があった
- 3—一致した

ElementRelevance の場合、パーシングの値には以下のいずれかが入ります。

- 0—空
- 1—要素の位置を変更する必要があった
- 2—一致したが、正規化する必要があった
- 3—一致した

ElementResultStatus の場合、(国を除くすべての住所要素に対して) 以下のいずれかの値が入ります。

- 0—空
- 1—検証されず、変更されていない。元のデータがコピーされている
- 2—検証されなかったが正規化された
- 3—検証されたが、入力が無効 (データベースによれば、数値が正しい範囲内にない) であるため変更されなかった入力がコピーされ、修正されていない (このステータス値はバッチ モードでのみセットされる)
- 4—検証されたが、参照データが見つからないため変更されていない
- 5—検証されたが、複数の一致が見つかったため変更されていない。バッチ モードでのみセットされる。そうでない場合、入力を置換する複数の提案が修正済みとしてマークされる (ステータス値 7)
- 6—検証され、入力値の除外という変更が行われている
- 7—検証され、参照データに基づく修正という変更が行われている
- 8—検証され、参照データに基づく値の追加という変更が行われている
- 9—検証され、変更はされていないが、配達ステータスがわからない (例えば、DPV の値で、指定の数値範囲が参照データと部分的にしか一致せず正しくないなど)
- C—検証され、妥当性確認済みだが名前が旧式であるため変更されている
- D—検証され、妥当性確認済みだが外名から公式の名前に変更されている
- E—検証され、妥当性確認済みだが大文字と小文字の区別、または言語に基づく正規化のため変更されている。入力が別の言語と完全に一致する場合に、検証でのみこのステータスがセットされる
- F—検証され、妥当性確認済みで、完全に一致したため変更されなかった

国 (ポジション 19 と 20) に関しては、以下のいずれかの値が入ります。

- 0—空
- 1—国が認識されない
- 4—国が DefaultCountryISO3 の設定から認識された
- 5—国が認識されない。複数の一致が見つかった
- 6—国がスクリプトから認識された
- 7—国がフォーマットから認識された
- 8—国が主要な都市から認識された
- 9—国が省から認識された
- C—国が地域から認識された
- D—国がエラーありで名前から認識された
- E—国が名前からエラーなしで認識された
- F—国が ForceCountryISO3 の設定から認識された

ValidateAddressLoqate

ValidateAddressLoqate 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。ValidateAddress 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。

ValidateAddressLoqate は、ValidateAddressLoqate 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。

ValidateAddressLoqate 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、ValidateAddressLoqate 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。

入力

表 114 : 入力フォーマット

columnName	書式	説明
AddressLine1	String	最初の住所行。
AddressLine2	String	2 行目の住所行。
AddressLine3	String	3 行目の住所行。
AddressLine4	String	4 行目の住所行。
City	String	都市名。

columnName	書式	説明
Country	String	国コードまたは名前を、以下のいずれかのフォーマットで入力します。 • 2 文字の ISO 3116-1 Alpha-2 国コード • 3 文字の ISO 3116-1 Alpha-3 国コード • 英語の国名 ISO コードの一覧については、**ISO 国コードとモジュール サポート**（598ページ）を参照してください。
FirmName	String	会社名または企業名。
PostalCode	String	住所の郵便番号は、次のフォーマットのいずれかで表されます。 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
StateProvince	String	州または省。

オプション

以下の表に、ValidateAddressLoqate が返す情報の種類を制御するオプションの一覧を示します。

表 115 : 出力データ オプション

オプション名	説明
Database.Loqate	国際住所の妥当性の確認に使用するデータベースを指定します。国際住所検証用のデータベースを指定するには、**[データベース]** ドロップダウン リストからデータベースを選択します。

オプション名	説明
OutputFieldLevelReturnCodes	フィールドレベルの結果インジケータを含めるかどうかを指定します。フィールドレベルの結果インジケータは、ValidateAddressLoqate が各住所要素をどのように処理したかを示します。フィールドレベルの結果インジケータは、修飾子 "Result" で返されます。例えば、HouseNumber のフィールドレベルの結果インジケータは **HouseNumber.Result** に格納されます。結果インジケータの出力フィールドの完全な一覧は、**結果インジケータ**（544ページ）を参照してください。 **N** フィールドレベルのリターン コードを出力しません (デフォルト)。 **Y** フィールドレベルのリターン コードを出力します。

オプション名	説明
OutputFormattedOnFail	住所の妥当性を確認できない場合に書式を整えた住所を返すかどうかを指定します。住所には、その国の標準住所書式が設定されます。このオプションを選択しない場合、ValidateAddressLoqate が住所の妥当性を確認できないと、出力住所フィールドは空白になります。 **N** 失敗した住所の書式を整えません (デフォルト)。 **Y** 失敗した住所の書式を整えます。 **[標準住所を含める]**、**[住所行の要素を含める]**、および **[郵便情報を含める]** の各チェック ボックスで指定されたフォーマットを使用して、書式を整えた住所が返されます。**[住所行の要素を含める]** を選択した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddressLoqate が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、**[正規化された入力住所要素を含める]** を選択します。 このオプションをオンにする場合は、**[標準住所を含める]** と **[住所行の要素を含める]** またはそのいずれかを選択する必要があります。 **OutputRecordType** オプションで指定されたフォーマットを使用して、書式を整えた住所が返されます。**OutputRecordType=E** を指定した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddressLoqate が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、**OutputRecordType=I** を指定します。 Y を指定する場合は、OutputRecordType に対して "A" と "E" またはそのいずれかを指定する必要があります。 **Option.OutputRecordType** オプションで指定されたフォーマットを使用して、書式を整えた住所が返されます。**Option.OutputRecordType=E** を指定した場合は、妥当性が確認できた住所に対しては、パース済みで妥当性を確認済みの住所が、パース済み住所要素に含まれることに注意してください。住所の妥当性が確認できなかった場合には、パース済み住所要素には、入力住所がパース済み形式で含まれることになります。ValidateAddressLoqate が住所の妥当性を確認できたかどうかにかかわらず、必ず入力住所をパース済み形式で出力したい場合は、**Option.OutputRecordType=I** を指定します。

オプション名	説明
OutputAddressBlocks	実際の郵便物に印字される、書式を整えた住所を返すかどうかを指定します。住所の各行が、別々の住所ブロック フィールドに入れて返されます。**AddressBlock1** から **AddressBlock9** まで、最大 9 つの住所ブロック出力フィールドが使用されます。 例えば、以下の住所入力の場合、 AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 以下の住所ブロックが出力されます。 AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA ValidateAddressLoqate は、郵便当局の規格に従って住所の書式を整え、住所ブロックの形式にします。国名は、万国郵便連合 (UPU) の国名で返されます。OutputCountryFormat オプションは、住所ブロックの国名には影響を与えないことに注意してください。[国フォーマット] オプションは、**Country** 出力フィールドに返される名前のみに影響を与えます。 次のいずれかです。 **N** 住所ブロックを返しません。こちらがデフォルトです。 **Y** 住所ブロックを返します。

オプション名	説明
AmasFormatting	Address Matching Approval System (AMAS) 表記を使用して出力住所データをフォーマットすることを指定します。 このオプションを使用すると、Validate Address Loqate は、住所を正規化するときに AMAS ルールを使用するようになります。AMAS は、オーストラリア郵政公社が定める、住所規格を徹底するためのプログラムです。AMAS 書式設定表記の詳細については、『Address Matching Approval System (AMAS) Handbook』を参照してください。 このオプションを使用すると、出力データは次のように変更されます。 • 数値フィールドにはゼロが付加されます。この影響を受けるのは、HouseNumber、HouseNumber2、PostalDeliveryNumber、および DPID の各出力フィールドです。例えば、入力フィールドが 298 New South Head Rd Double Bay NSW 2028 の場合、HouseNumber フィールドの形式は 298 から 00298 に変更されます。 • 一致しない場合、DPID フィールドの桁はすべてゼロになります。例えば、00000000 などです。 • 一致しない場合、すべてゼロを含む数値フィールドを除き、すべてのリターン フィールド (パース済み住所要素) が空白になります。 • CCD フィールドは出力されません。 有効な値は、次のとおりです。 **N** AMAS 表記を使用して出力データをフォーマットしません (デフォルト)。 **Y** AMAS 表記を使用して出力データをフォーマットします。 **注 :** このオプションを選択すると、**[許容レベル]** フィールドと **[最小マッチ スコア]** フィールドの選択に関わらず、AMAS フォーマットで結果が返されます。
OutputCasing	出力データの大文字と小文字の区別を指定します。次のいずれかです。 **M** 出力には、大文字と小文字が混在させます(デフォルト)。例: 123 Main St Mytown FL 12345 **U** 出力に大文字を使用します。例: 123 MAIN ST MYTOWN FL 12345

オプション名	説明
HomeCountry	デフォルト国を指定します。大部分の住所が存在する国を指定してください。例えば、処理する住所の大部分がドイツにある場合は、ドイツを指定します。ValidateAddressLoqate は、[StateProvince]、[PostalCode]、および [Country] の各住所フィールドから国を特定できなかった場合、指定された国を使用して、住所の検証を試みます。有効な国名には次のものがあります。 Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome And Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe

オプション名	説明
OutputCountryFormat	**Country** 出力フィールドに返される国名に使用するフォーマットを指定します。例えば、英語を選択した場合、"Deutschland" という国名は "Germany" として返されます。 **E** 英語の国名を使用します (デフォルト)。 **I** 国名の代わりにその国の 2 文字の ISO の略語を使用します。 **U** 国名の代わりにその国の万国郵便連合 (Universal Postal Union: UPU) の略語を使用します。
OutputScript	出力がどのアルファベットまたはスクリプトで返されるかを指定します。このオプションは双方向で、通常はネイティブからラテン文字へ、およびラテン文字からネイティブへ実行されます。 **Input** 書き直しを実行せず、入力と同じスクリプトで出力します (デフォルト)。 **Native** 使用可能な場合は、選択した国のネイティブ スクリプトで出力します。 **Latn** 英語の値を使用します。

オプション名	説明
許容レベル AcceptanceLevel	

オプション名	説明
	レコードが正常に処理されたとみなされるために達成する必要のある、最小検証レベルを指定します。このフィールドの値は、“処理後検証マッチ レベル” と呼ばれる、Address Verification Code の 2 つめの文字に対応します。 • **5** — 配達ポイント (建物または郵便受け)。入力レコードの ApartmentNumber、HouseNumber、Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。ApartmentNumber は正しいがその他のフィールドが正しくない場合、確信レベルは中程度になりますが、ApartmentNumber は他のフィールドよりも細かいレベルであるため、Loqate エンジンはこの場合に、ApartmentNumber を特定できるはずです。Loqate エンジンが ApartmentNumber とその他のフィールドをパーシングできない場合は、確信レベルは 0 となります。 • **4** — 敷地または建物。入力レコードの HouseNumber、Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。HouseNumber は正しいがその他のフィールドが正しくない場合、確信レベルは中程度になりますが、HouseNumber は他のフィールドよりも細かいレベルであるため、Loqate エンジンはこの場合に、HouseNumber を特定できるはずです。Loqate エンジンが HouseNumber とその他のフィールドをパーシングできない場合は、確信レベルは 0 となります。 • **3** — 大ストリート、道路、またはストリート。入力レコードの Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。City は正しいが StateProvince が正しくない場合、確信レベルは中程度になりますが、City は StateProvince に含まれるため、Loqate エンジンはこの場合に、StateProvince を特定できるはずです。Loqate エンジンが City または両方のフィールド (City と StateProvince) をパーシングできない場合は、確信レベルは 0 となります。 • **2** — 地方 (都市または町)。入力レコードの City と StateProvince の両方が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。City は正しいが StateProvince が正しくない場合、確信レベルは中程度になりますが、City は StateProvince に含まれるため、Loqate エンジンはこの場合に、StateProvince を特定できるはずです。Loqate エンジンが City または両方のフィールド (City と StateProvince) をパーシングできない場合は、確信レベルは 0 となります。 • **1** — 行政区域 (州または地域)。入力レコードの StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。 • **0** — なし。これは、最も制限の緩いマッチ オプションと同じです。

オプション名	説明
IsDuplicateHandlingMaskEnable	重複処理マスクを有効にし、重複レコードの処理および削除の方法を指定します。次のオプションから 1 つ以上を選択します。 **S** デフォルトで選択されています。入力の前処理により、単一フィールドで発生している重複を削除します。 **C** デフォルトで選択されています。入力の前処理により、すべてのフィールドわたって重複を削除します。 **T** 入力の前処理により、標準住所フィールドでないフィールド内の重複を削除します。 **F** デフォルトで選択されています。検証の出力の後処理により、検証されていないフィールドから重複を削除します。
MinimumMatchScore	Validate Address Loqate において、Loqate 参照データベースでマッチ結果を得るために、住所に加える変更の度合いを、0 ～ 100 の間の数値で指定します。数値が小さいほど、大きな変更が許容されます。100 の値は、パーシング後に入力住所と検証済み住所がほぼ同一であることを意味します。0 の値は、検証済み住所を得るためにパーシング後の入力住所を大きく変更してしまってもよいことを意味します。
KeepMultimatch	一致する可能性のある住所を複数持つ入力住所に対して複数の住所を返すかどうかを指定します。 **Y** 複数のマッチを返します (デフォルト)。 **N** 複数のマッチを返しません。 詳細については、**複数マッチを返す**（534ページ）を参照してください。
FailMultipleMatches	一致する可能性のある住所が複数存在する入力住所に対して、複数の住所を許可しません。

複数マッチを返す

ValidateAddressLoqate が、入力住所に一致する可能性のある複数の住所を郵便データベース内で検出した場合に、ValidateAddressLoqate がそれらの複数の住所を返すように設定できます。例えば、次の住所は米国郵便データベース内の複数の住所にマッチします。

PO BOX 1 New York, NY

オプション

複数マッチを返すには、次の表に示すオプションを使用します。

表 116 : 複数マッチのオプション

オプション名	説明/有効値
KeepMultimatch	一致する可能性のある住所を複数持つ入力住所に対して複数の住所を返すかどうかを示します。 **Y** 複数のマッチを返します (デフォルト)。 **N** 複数のマッチを返しません。
MaximumResults	A を入力します。返す住所の最大数を示す 1 ～ 10 の数字を入力します。デフォルト値は 1 です。 **注 :** Keepmultimatch=N と KeepMultimatch=Y/MaximumResults=1 の違いは、KeepMultimatch=N は複数マッチによって失敗が返され、KeepMultimatch=Y かつ MaximumResults=1 は複数マッチによって 1 つのレコードが返される点です。
OutputFieldLevelReturnCodes	どの出力住所が候補住所かを特定するには、OutputFieldLevelReturnCodes に対して値 **Y** を指定する必要があります。このように設定すると、候補住所のレコードのフィールドレベルの結果インジケータに 1 つ以上の値 "M" が格納されます。

出力

複数マッチを返すよう選択した場合、住所は指定した住所フォーマットで返されます。住所フォーマットの指定については、**オプション**（525ページ）を参照してください。どのレコードが候補住所であるかを特定するには、フィールドレベルの結果インジケータに複数の値 "M" があるかどうかを調べます。詳細については、**結果インジケータ**（544ページ）を参照してください。

マッチ スコアのしきい値オプション

マッチ スコアのしきい値を設定するための 2 つのオプションがあります。

> **注 :** これらのオプションは Validate Address Loqate のユーザ インターフェイスにはなく、以下のファイルの中にあります。
>
> *SpectrumDirectory*/server/modules/loqate/env.properties

[MatchScoreAbsoluteThreshold] オプションは、レコードがマッチングの候補とみなされるために達しなければならない最小マッチ スコアを指定するために使用されます。デフォルト値は 60 で、最大値は 100 です。

[MatchScoreThresholdFactor] は、最も高いマッチング結果を 100 とした場合の係数を表す値です。この値は、結果候補を検討する際のボーダーラインとして使用されます。係数の値が高いほど、良い検証結果が得られる確率が高くなります。デフォルト値は 95 で、最大値は 100 です。

出力

ValidateAddressLoqate からの出力には、選択した出力カテゴリに応じてさまざまな情報が含まれます。

標準住所出力

標準住所出力は、宛名ラベルに表記される住所に対応する 4 行の住所で構成されます。都市、州/省、郵便番号などのデータも、標準住所出力に含まれます。**[標準住所を含める]** チェック ボックスを選択 **OutputRecordType = A** と設定した場合、ValidateAddressLoqate は、妥当性を確認した住所に対し、標準住所出力を返します。**OutputRecordType = A** と設定したかどうかにかかわらず、妥当性が確認できなかった住所に対しては、標準住所フィールドが必ず返されます。妥当性が確認されなかった住所に対しては、標準住所出力フィールドには、入力住所がそのまま含まれます ("パス スルー" データ)。ValidateAddressLoqate において、妥当性が確認できなかった場合に、郵便当局の規格に従って住所を正規化するには、要求において **OutputFormattedOnFail = Y** を指定します。

表 117 : 標準住所出力

columnName	説明
AdditionalInputData	特定の住所コンポーネントに一致しなかった入力データ。詳細については、**その他の入力データについて**を参照してください。
AddressLine1-4	住所の妥当性が確認された場合は、妥当性が確認され、正規化された住所の 1 行目です。住所の妥当性が確認できなかった場合は、入力住所の 1 行目がそのまま出力されます。住所ブロック出力フィールドは最大で、AddressLine1 から AddressLine4 の 4 つになります。
City	妥当性が確認された都市名。

columnName	説明
Country	OutputCountryFormat で選択した、以下のいずれかのフォーマットで示された国。 • ISO コード • UPU コード • 英語
FirmName	妥当性が確認された企業名。
PostalCode	妥当性が確認された ZIP Code™ または郵便番号。
PostalCode.AddOn	ZIP Code™ の 4 桁のアドオン部分。例えば、60655-1844 という ZIP Code™ において、4 桁のアドオン部分は 1844 になります
PostalCode.Base	5 桁の ZIP Code™。例: 20706。
StateProvince	妥当性が確認された州または省の略称。

パース済み住所要素出力

OutputRecordType = E を設定した場合、出力住所は、パース済み住所の形式で書式設定されます。ValidateAddressLoqate で、妥当性が確認できなかった場合に、パース済み住所形式で書式設定されたデータ (正規化済み住所) を返すには、**OutputFormattedOnFail** = Y を指定します。

> **注:** ValidateAddressLoqate で、妥当性が確認できたかどうかにかかわらず、常にパースした入力データを返すには、**OutputRecordType = I** を指定します。詳細については、パース済み入力（540ページ）を参照してください。

表 118 : パース済み住所出力

columnName	説明
AddressBlock1-9	AddressBlock 出力フィールドには、正規化済み、または標準化済みの住所が、実際の郵便物に印刷される形式にフォーマットされて入ります。Validate Address Global は住所を郵便当局の規格に従って住所ブロックにフォーマットします。住所の各行が、別々の住所ブロック フィールドに入れて返されます。AddressBlock1 から AddressBlock9 まで、最大 9 つの住所ブロック出力フィールドが使用されます。例えば、以下の住所入力の場合、 AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 以下の住所ブロックが出力されます。 AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882
ApartmentLabel	アパート指定子 (STE や APT など)。例: 123 E Main St **APT** 3
ApartmentNumber	アパート番号。例: 123 E Main St APT **3**
ApartmentNumber2	補助的なアパート番号。例: 123 E Main St APT 3, **4th** Floor **注 :** このリリースでは、このフィールドは常に空白になります。
Building	個々の場所を識別するためのわかりやすい名前。
City	妥当性が確認された都市名

columnName	説明
Country	国。フォーマットは、OutputCountryFormat で選択したものになります。 • ISO コード • UPU コード • 英語
County*	国における最小の地理的データ要素。例: **米国の郡**
FirmName	妥当性が確認された企業名
HouseNumber	家番号 1。例: **123** E Main St Apt 3
LeadingDirectional	接頭方向指示。例: 123 **E** Main St Apt 3
POBox	私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode	妥当性が検証された郵便番号。米国住所に対しては、ZIP Code と呼びます。
Principality *	国における最大の地理的データ要素
StateProvince	妥当性が確認された州または省の名前
StreetAlias	ストリート名のエリアス。通常は、ストリート上の特定の範囲の住所のみに対して使用されます。ストリート名のエイリアスを出力に使用しないと指定した場合は、ストリートにエイリアス名があるかどうかにかかわらず、出力ではストリートの "基本" 名が使用されます。例: 123 E **Main** St Apt 3

columnName	説明
StreetName	ストリート名。例: 123 E **Main St** Apt 3
StreetSuffix	ストリート接尾語。例: 123 E Main **St** Apt 3
Subcity*	[Locality] フィールドの内容に依存する、より小さな居留区データ要素。例: **トルコ人居留区**。
Substreet*	国における従属ストリートまたはブロック データ要素。例: **英国の従属ストリート**。
TrailingDirectional	接尾方向指示。例: 123 Pennsylvania Ave **NW**

*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。

パース済み入力

出力には、パース済み形式で入力住所を含めることができます。このようなタイプの出力は、"パース済み入力" と呼ばれます。パース済み入力フィールドには、ValidateAddress が住所の妥当性を検証したかどうかにかかわらず、入力として使用される住所データが含まれます。パース済み入力は、住所の妥当性を検証できた場合にパース済み住所要素に妥当性が検証された住所が含まれ、オプションで、住所の妥当性が検証できなかった場合には入力データが含まれるという点で、"パース済み住所要素" 出力と異なります。パース済み入力には、ValidateAddress が住所の妥当性を検証したかどうかにかかわらず、常に入力住所が含まれます。

パース済み入力フィールドを出力に含めるには、**OutputRecordType = I** を設定します。

表 119 : パース済み入力

columnName	説明
ApartmentLabel.Input	アパート指定子 (STE や APT など)。例: 123 E Main St **APT** 3

columnName	説明
ApartmentNumber.Input	アパート番号。例: 123 E Main St APT **3**
City.Input	妥当性が確認された都市名
Country.Input	国。フォーマットは、OutputCountryFormat で選択したものになります。 • ISO コード • UPU コード • 英語
County.Input*	国における最小の地理的データ要素。例: **米国の郡**
FirmName.Input	妥当性が確認された企業名
HouseNumber.Input	家番号 1。例: **123** E Main St Apt 3
LeadingDirectional.Input	接頭方向指示。例: 123 **E** Main St Apt 3
POBox.Input	私書箱番号。住所が地方配送路住所である場合は、地方配送路の私書箱番号がここに表示されます。
PostalCode.Input	妥当性が検証された郵便番号。米国住所に対しては、ZIP Code と呼びます。
Principality.Input *	国における最大の地理的データ要素
StateProvince.Input	妥当性が確認された州または省の名前

columnName	説明
StreetAlias.Input	ストリート名のエリアス。通常は、ストリート上の特定の範囲の住所のみに対して使用されます。ストリート名のエイリアスを出力に使用しないと指定した場合は、ストリートにエイリアス名があるかどうかにかかわらず、出力ではストリートの "基本" 名が使用されます。基本名とは、ストリート全体に適用される名称です。例えば、StreetName が “N MAIN ST” の場合、StreetAlias フィールドには “MAIN” が含まれ、StreetSuffix フィールドには大ストリートタイプである "ST" が返されます。
StreetName.Input	ストリート名。例: 123 E **Main St** Apt 3
StreetSuffix.Input	ストリート接尾語。例: 123 E Main St Apt 3
Subcity.Input*	[Locality] フィールドの内容に依存する、より小さな居留区データ要素。例: **トルコ人居留区**。
Substreet.Input*	国における従属ストリートまたはブロック データ要素。例: **英国の従属ストリート**。
TrailingDirectional.Input	接尾方向指示。例: 123 Pennsylvania Ave **NW**

*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。

ジオコード出力

ValidateAddressLoqate は、緯度/経度、ジオコーディング マッチ コード、従属する地方、従属地方に含まれる地方、従属する大ストリート、下位行政区画と上位行政区画、検索距離を出力として返します。マッチ コードは、入力住所がどの程度まで既知の住所に一致したかを表すとともに、マッチングの全体的なステータスを示します。検索距離コードは、ジオコードが住所の実際の物理的な位置にどの程度近いかを表します。

表 120：ジオコード住所出力

columnName	説明
Geocode.MatchCode	住所に対するジオコード マッチングのステータスとレベルを表す 2 バイト コード。 最初のバイトはジオコーディング ステータスを表し、次のいずれかになります。 **A** 入力住所に一致する複数のジオコード候補が検出され、それらの平均が返されました。 **I** ある範囲の入力住所のロケーションからジオコードを補間することができました。 **P** 入力住所に一致する単一のジオコードが検出されました。 **U** 入力住所に対するジオコードを生成できませんでした。 2 つめのバイトはジオコード マッチングのレベルを表し、次のいずれかになります。 **5** 配達ポイント (郵便受けまたは従属する建物) **4** 敷地または建物 **3** Thoroughfare **2** 地方 **1** 行政区画 **0** なし
Latitude	小数第 5 位まで計算される 8 桁の度数 (指定したフォーマットで表記されます)。
Longitude	小数第 5 位まで計算される 8 桁の度数 (指定したフォーマットで表記されます)。
SearchDistance	メートル単位で表した精度の半径。与えられたジオコードと実際の物理的位置との間の推測最大距離を表します。このフィールドは、基盤の参照データから導き出され、その精度と対象範囲に依存します。

表 121：都市/ストリート/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
ストリートセントロイド	A4/P3
郵便番号/都市セントロイド	A3/P2/A2

注： Geocode.Match.Code は、ストリート セグメントに対して 2 つの座標を返すことはしません (ストリートの一部の開始と終了など)。代わりに、入力に対するリターン コードが I3 になった (大ストリートまたはストリート レベルに補間され、敷地番号は入力されていない) 場合は、ストリート全体が計算に使用されます。

結果インジケータ

結果インジケータは、住所に対して実行した処理の種類に関する情報を提供します。結果インジケータには、次の 2 種類があります。

レコード レベルの結果インジケータ

レコード レベルの結果インジケータは、各レコードに対する ValidateAddressLoqate 処理の結果に関するデータを提供します。例えば、マッチングの成功または失敗、住所を処理したコーダーなどの詳細情報を示します。以下の表に、ValidateAddressLoqate が返すレコード レベルの結果インジケータの一覧を示します。

表 122：レコード レベル インジケータ

フィールド名	説明
Confidence	返された住所に割り当てられた確信レベル。範囲は 0 ～ 100 です。0 は失敗を表し、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表します。複数のマッチ結果がある場合、確信レベルは 0 です。この数値の計算方法については、**Validate Address Loqate 確信アルゴリズムの概要** を参照してください。

フィールド名	説明
CouldNotValidate	マッチしなかった場合の、妥当性が確認できなかった住所コンポーネント。 • ApartmentNumber • HouseNumber • StreetName • PostalCode • City • Directional • StreetSuffix • Firm • POBoxNumber **注：** 複数のコンポーネントがカンマ区切りリストとして返されることがあります。
MatchScore	MatchScore は、入力データと、最も近い一致として検出された参照データとの間の類似性を表します。MatchScore は、マッチ結果を得るために入力住所をどれだけ変更したかを表す Confidence とはまったく異なるものです。MatchScore の意味は米国住所と米国以外の住所で異なります。 int getFieldMatchscore (unit record, const char*) 関数は、入力データと、最も近い一致として検出された参照データとの間の類似性を表す 0 ～ 100 の整数値です。100 は、入力データに、エイリアス、大文字小文字、付加記号以外の変更を加えなかったことを表します。0 は、入力データと、最も近い一致として検出された参照データとの間にまったく類似性がないことを表します。 **注：** Validate Address Loqate および Advanced Matching モジュールのコンポーネントは、どちらも MatchScore フィールドを使用します。データフローの出力の MatchScore フィールドの値は、出力ステージに送られる前に最後に値を変更したステージによって決まります。データフローに Validate Address Loqate および Advanced Matching モジュールのコンポーネントが含まれ、各ステージの MatchScore 出力フィールドを確認したい場合は、Transformer ステージを使用して、MatchScore 値を他のフィールドにコピーしてください。例えば、Validate Address Loqate によって MatchScore という出力フィールドが作成され、Transformer ステージによって Validate Address Loqate の MatchScore フィールドが AddressMatchScore というフィールドにコピーされます。マッチャー ステージを実行すると、マッチャーから得た値が MatchScore フィールドに設定され、Validate Address Loqate から得た AddressMatchScore の値が引き渡されます。
ProcessedBy	住所を処理した住所コーダーです。 **LOQATE** Loqate コーダーが住所を処理しました。

フィールド名	説明
Status	マッチの成功または失敗。複数のマッチがある場合、一致する可能性のあるすべてのものに対してこのフィールドが "F" になります。 **NULL** 成功 **F** 失敗
Status.Code	失敗の原因 (ある場合)。 • UnableToValidate
Status.Description	問題の説明 (ある場合)。 **Address Not Found** Status.Code=UnableToValidate の場合にこの値が表示されます。

フィールドレベルの結果インジケータ

フィールドレベルの結果インジケータは、ValidateAddressLoqate が各住所要素をどのように処理したかを示します。フィールドレベルの結果インジケータは、修飾子 "Result" で返されます。例えば、HouseNumber のフィールドレベルの結果インジケータは **HouseNumber.Result** に格納されます。

フィールドレベルの結果インジケータを有効にするには、**OutputFieldLevelReturnCodes=Y** を指定します。

次の表に、フィールドレベルの結果インジケータの一覧を示します。特定のフィールドが住所に適用されない場合、結果インジケータが空白になる場合があります。

表 123 : フィールドレベルの結果インジケータ

columnName		説明
ApartmentLabel.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。米国およびカナダの住所のみをサポートします。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国およびカナダの住所のみをサポートします。
	R	アパート ラベルが必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。
	S	正規化。このオプションには、標準の略語が含まれます。
	U	マッチしない。カナダの住所には適用されません。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
ApartmentNumber.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。カナダの住所のみ。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国EWS にマッチする米国の住所には、P の値が割り当てられます。米国およびカナダの住所のみをサポートします。
	R	アパート番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。
	S	正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。
	U	マッチしない。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName	説明
City.Result	**A** 追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。 **C** 修正済み。米国およびカナダの住所のみをサポートします。 **F** ハイフンの欠落または句読文字エラー。カナダの住所のみ。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国またはカナダの住所には適用されません。 **P** パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。 **R** 都市名が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。 **S** 正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。 **U** マッチしない。カナダの住所には適用されません。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
Country.Result	これらの結果コードは、米国またはカナダの住所には適用されません。 **M** 複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。 **S** 正規化。このオプションには、標準の略語が含まれます。 **U** マッチしない。 **V** 妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
County.Result*	国における最小の地理的データ要素。例: **米国の郡**

columnName		説明
FirmName.Result	**C**	修正済み。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国およびカナダの住所のみをサポートします。
	U	マッチしない。米国およびカナダの住所のみをサポートします。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。米国住所にのみ適用されます。
HouseNumber.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。
	C	修正済み。カナダの住所のみ。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	O	範囲外。米国またはカナダの住所には適用されません。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。
	R	家番号が必須ですが、入力住所から欠落しています。カナダの住所のみ。
	S	正規化。このオプションには、標準の略語が含まれます。米国またはカナダの住所には適用されません。
	U	マッチしない。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName		説明
LeadingDirectional.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。空白のない入力が、空白のない値に修正されました。米国住所にのみ適用されます。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。
	S	正規化。このオプションには、標準の略語が含まれます。
	U	マッチしない。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。カナダの住所には適用されません。
POBox.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。
	C	修正済み。カナダの住所のみ。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数マッチ。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。
	R	私書箱番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。
	S	正規化。このオプションには、標準の略語が含まれます。
	U	マッチしない。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

columnName		説明
PostalCode.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。米国およびカナダの住所のみをサポートします。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。カナダの住所には適用されません。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国住所をサポートしていません。
	R	郵便番号が必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。
	S	正規化。このオプションには、標準の略語が含まれます。米国またはカナダの住所には適用されません。
	U	マッチしない。例えば、ストリート名と郵便番号が一致しない場合、StreetName.Result と PostalCode.Result の両方に U が割り当てられます。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
PostalCode.Type	**P**	ZIP Code™ には、PO Box 住所のみが含まれます。米国住所にのみ適用されます。
	U	ZIP Code™ は、特定の会社または場所に割り当てられたユニークな ZIP Code™ です。米国住所にのみ適用されます。
	M	ZIP Code™ は、軍施設の住所です。米国住所にのみ適用されます。
	NULL	ZIP Code™ は、標準 ZIP Code™ です。
Principality.Result *		国における最大の地理的データ要素

columnName	説明	
StateProvince.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。米国住所にのみ適用されます。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国またはカナダの住所には適用されません。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国およびカナダの住所のみをサポートします。
	R	アパート ラベルが必須ですが、入力住所から欠落しています。米国住所にのみ適用されます。
	S	正規化。このオプションには、標準の略語が含まれます。米国住所をサポートしていません。
	U	マッチしない。カナダの住所には適用されません。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
StreetAlias.Result	ストリート名のエリアス。通常は、ストリート上の特定の範囲の住所のみに対して使用されます。ストリート名のエイリアスを出力に使用しないと指定した場合は、ストリートにエイリアス名があるかどうかにかかわらず、出力ではストリートの "基本" 名が使用されます。基本名とは、ストリート全体に適用される名称です。例えば、StreetName が “N MAIN ST” の場合、StreetAlias フィールドには “MAIN” が含まれ、StreetSuffix フィールドには大ストリートタイプである "ST" が返されます。	

columnName	説明	
StreetName.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。カナダの住所のみ。
	C	修正済み。米国およびカナダの住所のみをサポートします。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。米国住所をサポートしていません。
	S	正規化。このオプションには、標準の略語が含まれます。米国およびカナダの住所のみをサポートします。
	U	マッチしない。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
StreetSuffix.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。米国およびカナダの住所のみをサポートします。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。
	S	正規化。このオプションには、標準の略語が含まれます。
	U	マッチしない。米国住所をサポートしていません。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。
Subcity.Result*	[Locality] フィールドの内容に依存する、より小さな居留区データ要素。例: **トルコ人居留区**。	

columnName	説明	
Substreet.Result*	国における従属ストリートまたはブロック データ要素。例: **英国の従属ストリート**。	
TrailingDirectional.Result	**A**	追加済み。フィールドが空白の入力フィールドに追加されました。米国およびカナダの住所のみをサポートします。
	C	修正済み。米国およびカナダの住所のみをサポートします。
	F	書式設定済み。郵便規格に準拠するよう、スペースや句読文字が変更されました。米国またはカナダの住所には適用されません。
	M	複数。入力住所が郵便データベース内の複数レコードにマッチし、マッチする各レコードのこのフィールドの値が異なります。米国住所にのみ適用されます。
	P	パススルー。データは検証プロセスで使用されませんでしたが、出力に保持されました。カナダの住所のみ。
	S	正規化。このオプションには、標準の略語が含まれます。
	U	マッチしない。カナダの住所には適用されません。
	V	妥当性が確認されました。データは正しいことが確認され、入力から変更されていません。

*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。

AVC コード

Address Verification Code (AVC) は、住所に対する精度インジケータで構成される 11 バイトのコードです。これらのコードは、処理結果の品質を表し、必要に応じて入力データを修正する方法に関するガイドラインを示します。個別住所のそれぞれに、独自のコードが付与されます。このコードは、データフローの出力内で自動的に返されます。AVC は、以下のような形式です。

V44-I44-P6-100

AVC は、8 つの部分で構成されます。

- 検証ステータス
- 処理後検証マッチ レベル
- 処理前検証マッチ レベル
- パーシング ステータス
- 辞書識別マッチ レベル
- コンテキスト識別マッチ レベル

- 郵便番号ステータス
- マッチスコア

検証ステータス

住所の検証レベル。

- **V** — 検証済み。入力データは、使用可能な参照データからの単一のレコードと完全に一致しています。シンプルな住所検証では、このコードが返されることが最良の結果とみなされます。
- **P** — 部分的に検証済み。入力データは、使用可能な参照データからの単一のレコードと部分的に一致しています。住所情報の詳細なデータが提供されているが、完全な検証を行うためには追加情報が必要であることを意味する可能性があります。
- **A** — 曖昧。入力に一致する可能性のある複数の住所があります。
- **U** — 検証不可。住所を検証するための十分な情報がない場合や、入力クエリが読み取れない場合に、これが返されます。出力フィールドには入力データが含まれます。
- **R** — 元に戻されました。レコードを、指定された最小許容レベルで検証できませんでした。元に戻すための最小レベルなどの詳細設定オプションが、処理において設定されている場合に生じます。出力フィールドには入力データが含まれます。
- **C** — 矛盾。相反する値を持つ複数の参照データと近似一致します。

処理後検証マッチ レベル

使用可能な参照データに対する、入力データの処理後のマッチ レベル。

- **5** — 配達ポイント (建物または郵便受け)。入力レコードの ApartmentNumber、HouseNumber、Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。ApartmentNumber は正しいがその他のフィールドが正しくない場合、確信レベルは中程度になりますが、ApartmentNumber は他のフィールドよりも細かいレベルであるため、Loqate エンジンはこの場合に、ApartmentNumber を特定できるはずです。Loqate エンジンが ApartmentNumber とその他のフィールドをパーシングできない場合は、確信レベルは 0 となります。
- **4** — 敷地または建物。入力レコードの HouseNumber、Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。HouseNumber は正しいがその他のフィールドが正しくない場合、確信レベルは中程度になりますが、HouseNumber は他のフィールドよりも細かいレベルであるため、Loqate エンジンはこの場合に、HouseNumber を特定できるはずです。Loqate エンジンが HouseNumber とその他のフィールドをパーシングできない場合は、確信レベルは 0 となります。
- **3** — 大ストリート、道路、またはストリート。入力レコードの Street、City、StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。City は正しいが StateProvince が正しくない場合、確信レベルは中程度になりますが、City は StateProvince に含まれるため、Loqate エンジンはこの場合に、StateProvince

を特定できるはずです。Loqate エンジンが City または両方のフィールド (City と StateProvince) をパーシングできない場合は、確信レベルは 0 となります。

- **2** — 地方 (都市または町)。入力レコードの City と StateProvince の両方が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。City は正しいが StateProvince が正しくない場合、確信レベルは中程度になりますが、City は StateProvince に含まれるため、Loqate エンジンはこの場合に、StateProvince を特定できるはずです。Loqate エンジンが City または両方のフィールド (City と StateProvince) をパーシングできない場合は、確信レベルは 0 となります。
- **1** — 行政区域 (州または地域)。入力レコードの StateProvince が、Loqate のリファレンス データセットと一致する場合、レコードは引き渡されるか、高い確信レベルを付与されます。
- **0** — なし。これは、最も制限の緩いマッチ オプションと同じです。

処理前検証マッチ レベル

使用可能な参照データに対する、入力データの処理前のマッチ レベル。

- **5** — 配達ポイント (建物または郵便受け)
- **4** — 敷地または建物。
- **3** — 大ストリート、道路、またはストリート。
- **2** — 地方 (都市または町)。
- **1** — 行政区域 (州または地域)。
- **0** — なし。

パーシング ステータス

住所のパーシング レベル。

- **I** — 識別およびパーシング済み。入力データは識別され、各コンポーネントに配置されました。例えば、"123 Kingston Av" に対して Validate Address Loqate は、"123" が敷地番号で、"Kingston" が大ストリート名で、"Av" または "Avenue" が大ストリートのタイプであると判断できます。
- **U** — パーシング不可。Validate Address Loqate は、入力データを識別およびパーシングできませんでした。"未検証" の検証ステータスと同様に、入力データは不完全またはあいまいでした。

辞書識別マッチ レベル

パターン マッチング (例えば、数値は敷地番号である可能性があります) や辞書マッチング (例えば、"rd" は大ストリートのタイプ "road" であったり、"London" は地方であったりする可能性があります) の適用による、入力データの認識済み形式のレベル。

- **5** — 配達ポイント (建物または郵便受け)
- **4** — 敷地または建物。
- **3** — 大ストリート、道路、またはストリート。
- **2** — 地方 (都市または町)。

- **1** — 行政区域 (州または地域)。
- **0** — なし。

コンテキスト識別マッチ レベル

出現のコンテキストに基づいて認識可能な、入力データのレベル。最も精度の低い形式のマッチングで、単語を特定の住所要素として識別することに基づいて行われます。例えば、前に敷地らしい要素があり、後に地方らしい要素が続き、後の項目は参照データまたは辞書とのマッチングによって識別されている場合、入力は大ストリートであると判断できる可能性があります。

- **5** — 配達ポイント (建物または郵便受け)
- **4** — 敷地または建物。
- **3** — 大ストリート、道路、またはストリート。
- **2** — 地方 (都市または町)。
- **1** — 行政区域 (州または地域)。
- **0** — なし。

郵便番号ステータス

郵便番号の検証レベル。

- **P8** — PostalCodePrimary と PostalCodeSecondary が検証済みです。
- **P7** — PostalCodePrimary は検証済みで、PostalCodeSecondary は追加または変更されています。
- **P6** — PostalCodePrimary が検証済みです。
- **P5** — PostalCodePrimary が、小さな変更を加えることによって検証済みです。
- **P4** — PostalCodePrimary が、大きな変更を加えることによって検証済みです。
- **P3** — PostalCodePrimary が追加されています。
- **P2** — PostalCodePrimary が辞書によって識別されています。
- **P1** — PostalCodePrimary がコンテキストによって識別されています。
- **P0** — PostalCodePrimary が空です。

マッチ スコア

識別済みの入力データと、レコードの出力データの間の類似性を表す 0 ～ 100 の間の数値。100 は、入力データに追加、エイリアス、大文字小文字、付加記号以外の変更を加えなかったことを表します。0 は、入力データと出力データの間にまったく類似性がないことを表します。

AMAS 出力

次の表に、ValidateAddressAUS が出力する標準フィールドを示します。

表 124 : 出力フィールド

フィールド名	説明
Barcode	DPID に基づく標準バーコード。 **F** 失敗 (バーコード不検出) **20 桁の数字** 成功
DPID	配達ポイント識別子。ストリート住所などの郵便物配達ポイントを一意に識別する 8 桁の数字。オーストラリア郵政公社郵便住所ファイルに規定されています。 **注 :** このフィールドは、AMAS で検証されていないオーストラリアの住所の場合は "00000000" となり、オーストラリア以外の住所の場合は空になります。
FloorNumber	フロア/レベル番号。例: 123 E Main St Apt 3, **4th** Floor
FloorType	フロア/レベル タイプ。例: 123 E Main St Apt 3, 4th **Floor**
PostalBoxNum	郵便配達番号。例: PO Box 42

誤検出

誤検出とは

住所リストが生成されるのを防ぐため、DPV と LACS^{Link} のデータベースに誤検出レコードが含まれています。誤検出レコードは人為的に作成され、誤検出テーブルに格納された住所です。DPV や LACS^{Link} のクエリで望ましくない応答が発生すると、誤検出テーブルに対してクエリが行われます。このテーブルにマッチすると (誤検出マッチと呼ばれます)、ユーザの DPV または LACS^{Link} キーが無効になります。バッチ処理の場合、この違反を含むジョブでも正しく完了しますが、違反をレポートして DPV や LACS^{Link} を再びアクティベートするためのキーを取得するまでは、DPV や LACS^{Link} を使用する後続のジョブを実行できません。

注 : 誤検出レコードにマッチすることを、"シード レコード違反" という言葉を使って表現する場合もあります。この 2 つの用語の意味は同じです。

DPV 誤検出違反の報告

Spectrum™ Technology Platform は、サーバー ログのメッセージを利用して誤検出マッチを報告します。

誤検出マッチが発生した場合、クライアント/サーバーの呼び出しが例外をスローします。DPV 誤検出レコード違反が発生すると、サーバー ログに以下が記録されます。

```
WARN [Log] Seed record violation for S<ZIP, ZIP+4, Address, Unit> ERROR
 [Log] Feature Disabled: DPU: DPV Seed Record Violation. Seed Code:
S<Address, ZIP, ZIP+4, Unit>
```

> **注 :** DPV の誤検出レコードが見つかった場合、process() メソッド (COM、C++、Java、.NET の場合) から、将来の DPU が無効化されたという例外がスローされます。C の場合、processMessage() 関数によってゼロ以外の値が返されます。

次の手順に従って、違反を報告し、再起動キーを取得することができます。

1. ブラウザで、http://<サーバー>:<ポート>/<製品コード>/dpv.jsp に移動します。例えば、Universal Addressing モジュールの場合は http://localhost:8080/unc/dpv.jsp、Enterprise Geocoding モジュールの場合は http://localhost:8080/geostan/dpv.jsp というアドレスになります。
2. 差出人の情報を各フィールドに入力します。各フィールド名に続く括弧内の数字は、フィールドの最大長を示します。
3. 終了したら **[送信]** をクリックします。**[ファイルのダウンロード]** ダイアログが表示されます。
4. **[保存]** をクリックして、ファイルをコンピュータに保存します。**[名前を付けて保存]** ダイアログが表示されます。
5. ローカルにあるハード ドライブの場所とファイル名 (`c:\DPVSeedFile.txt` など) を指定して、**[保存]** をクリックします。
6. **www.g1.com/support** に移動してログインします。
7. **[DPV & LACS**[Link] **誤検出]** をクリックします。
8. 画面上の指示に従って、シード ファイルを添付し、再起動キーを取得します。

DPV 誤検出ヘッダ ファイル レイアウト

USPS® は、DPV 誤検出ヘッダ ファイルで必須となるレイアウトを定義しています。これは現在、180 バイトのレコードを 2 つ以上含む固定長のファイルと決められています。最初のレコードは常にヘッダ レコードとする必要があります。ヘッダ レコードのレイアウトを以下に示します。

表 125 : DPV 誤検出ヘッダ レコードのレイアウト

位置	長さ	説明	書式
1-40	40	差出人の会社名	英数字
41-98	58	差出人の住所行	英数字
99-126	28	差出人の都市名	英数字
127-128	2	差出人の州の略語	英数字
129-137	9	差出人の 9 桁の ZIP Code	数値
138-146	9	処理されたレコードの合計数	数値
147-155	9	DPV マッチとなったレコードの合計数	数値
156-164	9	DSF に対するマッチ率 (%)	数値
165-173	9	ZIP +4® に対するマッチ率 (%)	数値
174-178	5	ファイル上の ZIP Code の数	数値
179-180	2	誤検出の数	数値

トレーラ レコードにはDPV誤検出マッチに関する情報が含まれます。誤検出ファイルには、DPV誤検出マッチ 1 件につき 1 つのトレーラ レコードを追加する必要があります。トレーラ レコードのレイアウトを以下に示します。

表 126 : DPV 誤検出トレーラ レコードのレイアウト

位置	長さ	説明	書式
1-2	2	ストリート名の直前で方向を示す文字列	英数字
3-30	28	ストリート名	英数字
31-34	4	ストリートの種類の省略形	英数字
35-36	2	ストリート名の直後で方向を示す文字列	英数字
37-46	10	住所のプライマリの番号	英数字
47-50	4	住所のセカンダリの省略形	英数字
51-58	8	住所のセカンダリの番号	数値
59-63	5	マッチした ZIP Code	数値
64-67	4	マッチした ZIP + 4®	数値
68-180	113	空白埋め	スペース

LACS/Link 誤検出違反のレポート

Spectrum™ Technology Platform は、サーバー ログのメッセージを利用して誤検出マッチを報告します。誤検出マッチが発生し、クライアント/サーバーの呼び出しが例外をスローすると、バッチ ジョブはエラーになります。

> **注 :** 誤検出レコードにマッチすることを、"シード レコード違反" という言葉を使って表現する場合もあります。この 2 つの用語の意味は同じです。

誤検出レコードが見つかった場合、サーバ ログに以下が記録されます。

```
2005-05-06 17:05:38,978 WARN [com.g1.component.ValidateAddress] Seed
record violation for RR 2 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.component.ValidateAddress] Feature
 Disabled: LLU: LACS Seed Record Violation. Seed Code: 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.dcg.gateway.Gateway] Gateway
exception: com.g1.dcg.stage.StageException:
com.g1.dcg.component.ComponentException: Feature Disabled: LLU
2005-05-06 17:06:30,291 ERROR
[com.pb.spectrum.platform.server.runtime.core.license.impl.policy.Policy]
 Feature LACSLink Real- time is disabled.
```

> **注：** LACS^{Link} の誤検出レコードが見つかった場合、process() メソッド (COM、C++、Java、.NET の場合) から、将来の LLU が無効化されたという例外がスローされます。C の場合、processMessage() 関数によってゼロ以外の値が返されます。

1. ブラウザで、http://<サーバ名>:<ポート>/<製品コード>/lacslink.jsp に移動します。例えば、Universal Addressing モジュールの場合は http://localhost:8080/unc/lacslink.jsp、Enterprise Geocoding モジュールの場合は http://localhost:8080/geostan/lacslink.jsp というアドレスになります。
2. 差出人の情報を各フィールドに入力します。フィールド名に続く括弧内の数字は、フィールドの最大長を示します。終了したら **[送信]** をクリックします。**[ファイルのダウンロード]** ダイアログが表示されます。
3. **[保存]** をクリックして、ファイルをコンピュータに保存します。**[名前を付けて保存]** ダイアログが表示されます。
4. ローカルにあるハード ドライブの場所とファイル名 (`c:\lacslink.txt` など) を指定して、**[保存]** をクリックします。
5. **www.g1.com/support** に移動してログインします。
6. **[DPV & LACS^{Link} 誤検出]** をクリックします。
7. 画面上の指示に従って、シード ファイルを添付し、再起動キーを取得します。

ValidateAddress 確信アルゴリズム

Validate Address 確信アルゴリズムの概要

ValidateAddress は、妥当性を確認した各住所に対し、確信スコアを計算します。このスコアは、妥当性を確認した住所がどの程度正しいかを表します。確信コードの値は 0 ～ 100 で、0 はまったく確信がなく、100 はマッチ結果が正しいことに対する確信レベルが非常に高いことを表しま

す。確信コードは、個々の出力フィールドに対するマッチ結果を考慮に入れたアルゴリズムに基づいて計算されます。この計算に関連する出力フィールドは、以下のとおりです。

- Country
- City
- State
- PostalCode
- StreetName
- HouseNumber
- LeadingDirectional
- TrailingDirectional
- StreetSuffix
- ApartmentNumber

アルゴリズムにおいて、各フィールドにはそれぞれ独自の重みがあります。また、各フィールドに対し、"Success"、"Failure"、または "Changed" というマッチ結果が存在します("Changed" は、マッチさせるためにフィールドの内容を修正した場合に該当します)。マッチ結果 ("Success"、"Failure"、または "Changed") によって、そのフィールドの係数が決まります。以上より、確信コードは、重みと係数の積を用いて、次のように算出します。

```
Confidence = (Weight * Factor) for City
  + (Weight * Factor) for Country
  + (Weight * Factor) for State
  + (Weight * Factor) for PostalCode
  + (Weight * Factor) for StreetName
  + (Weight * Factor) for HouseNumber
  + (Weight * Factor) for Directionals
  + (Weight * Factor) for Street Suffix
  + (Weight * Factor) for ApartmentNumber
```

米国とカナダの住所に対する確信アルゴリズム住所の検証

以下の表で、米国およびカナダの住所に対する ValidateAddress の確信アルゴリズムのスコアリングとロジックについて説明します。およびカナダの住所のみに適用されます。

表 127 : 米国とカナダの住所に対する確信アルゴリズム住所の検証

フィールド	重み付け/マッチ スコア	変更の場合の係数[1]	埋めた場合の係数[2]
Country	10	100%	0%
City	10	50%	75%
State	15	50%	75%
PostalCode	15	25%	25%
StreetName	15	50%	75%
HouseNumber	15	50%	75%
Directionals	10	50%	75%
StreetSuffix	5	50%	75%
ApartmentNumber	5	50%	75%

国際住所用の確信アルゴリズム

米国およびカナダ以外の住所用に 2 つの確信アルゴリズムがあります。1 つは郵便番号を使用する国の住所用で、もう 1 つは郵便番号を使用しない国の住所用です。

以下の表に、郵便番号を使用している国の米国およびカナダ以外の住所用の確信アルゴリズムの詳細を示します。

[2] このフィールドに入力データが存在しないが、マッチを得るためにデータが埋められた場合のこと。

[1] このフィールドの入力データが、マッチを得るために変更された場合のこと。

表 128 : 郵便番号がある国用の確信アルゴリズム

フィールド	重み付け/マッチ スコア	変更の場合の係数[3]	埋めた場合の係数[4]	郵便データが使用できない場合の係数
Country	11.1111111111111	100%	0%	0%
City	11.1111111111111	50%	75%[5]	0%
State	16.6666666666667	100%	100	80%
PostalCode	16.6666666666667	100%	100%	80%
StreetName	16.6666666666667	50%	75%	50%
HouseNumber	16.6666666666667	50%	75%	50%
Directionals	0	50%	75%	0%
StreetSuffix	5.55555555555556	50%	75%	50%
ApartmentNumber	5.55555555555556	50%	75%	50%

[4] このフィールドに入力データが存在しないが、マッチを得るためにデータが埋められた場合のこと。

[3] このフィールドの入力データが、マッチを得るために変更された場合のこと。

[5] カテゴリ C の国の場合、この値は 50% です。各国は、次のいずれかのカテゴリに分類されます。

- **カテゴリ A** — 住所の郵便番号、都市名、州/郡名、ストリートの住所要素、および国名の検証と修正が可能です。
- **カテゴリ B** — 住所の郵便番号、都市名、州/郡名、および国名の検証と修正が可能です。ストリートの住所要素の検証または修正はサポートしません。
- **カテゴリ C** — 国名の検証および修正と、郵便番号の書式の検証が可能です。

郵便番号を使用しない国用の確信アルゴリズムの詳細を次の表に示します。

表 129 : 郵便番号がない国用の確信アルゴリズム

フィールド	重み付け/マッチ スコア	変更の場合の係数[6]	埋めた場合の係数[7]	郵便データが使用できない場合の係数
Country	13.3333333333333	100%	0%	0%
City	13.3333333333333	50%	75%[8]	0%
State	20	100%	100	80%
StreetName	20	50%	75%	50%
HouseNumber	20	50%	75%	50%
Directionals	0	50%	75%	0%
StreetSuffix	6.66666666666667	50%	75%	50%
ApartmentNumber	6.66666666666667	50%	75%	50%

[7] このフィールドに入力データが存在しないが、マッチを得るためにデータが埋められた場合のこと。

[6] このフィールドの入力データが、マッチを得るために変更された場合のこと。

[8] カテゴリ C の国の場合、この値は 50% です。各国は、次のいずれかのカテゴリに分類されます。

- **カテゴリ A** — 住所の郵便番号、都市名、州/郡名、ストリートの住所要素、および国名の検証と修正が可能です。
- **カテゴリ B** — 住所の郵便番号、都市名、州/郡名、および国名の検証と修正が可能です。ストリートの住所要素の検証または修正はサポートしません。
- **カテゴリ C** — 国名の検証および修正と、郵便番号の書式の検証が可能です。

郵便番号がない国の一覧を以下の表に示します。

表 130：郵便番号がない国

Afghanistan	Albania	Angola
Anguilla	バハマ	Barbados
Belize	Benin	Bhutan
Botswana	Burkina Faso	Burundi
Cameroon	Cayman Islands	Central African Rep.
Chad	Cocos Islands	Colombia
Comoros	Congo (Dem.民主主義人民共和国	Congo (Rep.)
Cote d'Ivoire	Korea (North)	Djibouti
Dominica	Equatorial Guinea	Eritrea
Fiji	Gabon	Gambia
Ghana	Grenada	Guyana
アイルランド	Jamaica	Kiribati
リビア	Malawi	Mali
Mauritania	Namibia	Nauru

Palau	Panama	Peru
Qatar	Rwanda	Saint Lucia
セントビンセントおよびグレナディーン諸島	Samoa	サントメ・プリンシペ
Seychelles	Sierra Leone	Suriname
Tanzania	Timor	Togo
Tonga	Trinidad & Tobago	Tuvalu
Uganda	United Arab Emirates	Vanuatu
Yemen	Zimbabwe	

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。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。例えば、結合された名前は、 "Mr.and Mrs.John Smith" や "Baltimore Gas & Electric dba Constellation Energy"です。
- 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。

入力

表 131 : Open Name Parser の入力

フィールド名	説明
CultureCode	入力された名前データのカルチャー。オプションは次のとおりです。 **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。 **注 :** Open Parser ドメインエディタを使用して独自のドメインを追加した場合、そのドメインのカルチャーとカルチャー コードも有効になります。
Name	パースしたい名前。このフィールドは必須です。

オプション

OpenNameParser のオプションは、Spectrum™ Technology Platform の任意のクライアントによってステージ レベルで設定するか、データフロー オプションを使用して実行時に設定できます。

パーシング オプション

次の表に、名前のパーシングを制御するオプションを示します。

表 132 : Open Name Parser パーシング オプション

オプション名	説明
ParseNaturalOrderPersonalNames	敬称、名、ミドル ネーム、姓、および接尾語の順序で名前をパースするかどうかを指定します。 **true** 正順序の個人名をパースします。 **false** 正順序の名前をパースしません。
ParseReverseOrderPersonalNames	姓が最初に指定されている名前をパースするかどうかを指定します。 **true** 逆順序の個人名をパースします。 **false** 逆順序の名前をパースしません。
ParseConjoinedNames	結合名をパースするかどうかを指定します。 **true** 結合名をパースします。 **false** 結合名をパースしません。
SplitConjoinedNames	`Bill & Sally Smith` など、複数の人物を含む結合名を複数のレコードに分割するかどうかを指定します。 **true** 結合名を分割します。 **false** 結合名を分割しません。
ParseBusinessNames	企業名をパースするかどうかを指定します。 **true** 企業名をパースします。 **false** 企業名をパースしません。
OutputAsList	パース済み名前要素をリスト形式で返すかどうかを指定します。 **true** パース済み要素をリスト形式で返します。 **false** パース済み要素をリスト形式で返しません。

オプション名	説明
ShortcutThreshold	パフォーマンスと品質のバランスをとる方法を指定します。パフォーマンスを上げると、品質出力が下がります。同様に、品質を上げると、パフォーマンスが下がります。このしきい値を満たすと、レコードに対して他の処理は実行されません。 0 ～ 100 の値を指定します。デフォルト値は 100 です。

カルチャー オプション

次の表に、名前カルチャーを制御するオプションを示します。

表 133 : Open Name Parser カルチャー オプション

オプション名	説明
DefaultCulture	パーシング グラマーに含めるカルチャーを指定します。デフォルトでは、グローバル カルチャーが選択されます。 カンマ区切りリストに 2 文字のカルチャー コードを優先する順に指定することで、カルチャーを指定します。例えば、まずはスペインのカルチャー、次に日本のカルチャーを使用して名前のパースを試みるには、次のように指定します。 `es,ja,,`

詳細設定オプション

次の表に、名前パーシング用の詳細オプションを示します。

表 134 : Open Name Parser の詳細オプション

オプション	説明
NaturalOrderPersonalNamesDomain	正順序個人名のパース時に使用するドメインを指定します。有効な値は、Enterprise Designer の Open Parser ドメイン エディタ ツールで定義されたドメイン名です。
NaturalOrderPersonalNamesPriority	使用する他のドメインに対する正順序個人名ドメインの優先度を示す 1 ～ 5 の数値を指定します。これにより、実行するパーサーの順序が決定されます。 ショートカットしきい値のオプションに設定された数字よりもスコアの高い最初のドメインに対して結果が返されます。そのしきい値に達しているドメインがない場合は、スコアの最も高いドメインに対する結果が返されます。複数のドメインが同時にしきい値に達している場合は、最初に実行された (ここで設定された順序によって決まる) ドメインが優先され、その結果が返されます。
ReverseOrderPersonalNamesDomain	逆順序個人名のパース時に使用するドメインを指定します。有効な値は、Enterprise Designer の Open Parser ドメイン エディタ ツールで定義されたドメイン名です。
ReverseOrderPersonalNamesPriority	使用する他のドメインに対する逆順序個人名ドメインの優先度を示す 1 ～ 5 の数値を指定します。これにより、実行するパーサーの順序が決定されます。 ショートカットしきい値のオプションに設定された数字よりもスコアの高い最初のドメインに対して結果が返されます。そのしきい値に達しているドメインがない場合は、スコアの最も高いドメインに対する結果が返されます。複数のドメインが同時にしきい値に達している場合は、最初に実行された (ここで設定された順序によって決まる) ドメインが優先され、その結果が返されます。
NaturalOrderConjoinedPersonalNamesDomain	正順序結合個人名のパース時に使用するドメインを指定します。有効な値は、Enterprise Designer の Open Parser ドメイン エディタ ツールで定義されたドメイン名です。

オプション	説明
NaturalOrderConjoinedPersonalNamesPriority	使用する他のドメインに対する正順序結合個人名ドメインの優先度を示す 1 ～ 5 の数値を指定します。これにより、実行するパーサーの順序が決定されます。 ショートカットしきい値のオプションに設定された数字よりもスコアの高い最初のドメインに対して結果が返されます。そのしきい値に達しているドメインがない場合は、スコアの最も高いドメインに対する結果が返されます。複数のドメインが同時にしきい値に達している場合は、最初に実行された (ここで設定された順序によって決まる) ドメインが優先され、その結果が返されます。
ReverseOrderConjoinedPersonalNamesDomain	逆順序結合個人名のパース時に使用するドメインを指定します。有効な値は、Enterprise Designer の Open Parser ドメイン エディタ ツールで定義されたドメイン名です。
ReverseOrderConjoinedPersonalNamesPriority	使用する他のドメインに対する逆順序結合個人名ドメインの優先度を示す 1 ～ 5 の数値を指定します。これにより、実行するパーサーの順序が決定されます。 ショートカットしきい値のオプションに設定された数字よりもスコアの高い最初のドメインに対して結果が返されます。そのしきい値に達しているドメインがない場合は、スコアの最も高いドメインに対する結果が返されます。複数のドメインが同時にしきい値に達している場合は、最初に実行された (ここで設定された順序によって決まる) ドメインが優先され、その結果が返されます。
BusinessNamesDomain	企業名のパース時に使用するドメインを指定します。有効な値は、Enterprise Designer の Open Parser ドメイン エディタ ツールで定義されたドメイン名です。

オプション	説明
BusinessNamesPriority	使用する他のドメインに対する企業名ドメインの優先度を示す 1 ～ 5 の数値を指定します。これにより、実行するパーサーの順序が決定されます。 ショートカットしきい値のオプションに設定された数字よりもスコアの高い最初のドメインに対して結果が返されます。そのしきい値に達しているドメインがない場合は、スコアの最も高いドメインに対する結果が返されます。複数のドメインが同時にしきい値に達している場合は、最初に実行された (ここで設定された順序によって決まる) ドメインが優先され、その結果が返されます。

実行時におけるオプションの設定

OpenNameParser のオプションは、データフロー オプションとしてエクスポーズされている場合は実行時に設定して引き渡すことができます。これにより、既存の設定をJSON 形式の名前パーシング文字列でオーバーライドできます。また、プロセス フローまたは Job Executor コマンドライン ツールからジョブを呼び出すときに、ステージ オプションを設定することもできます。

OpenNameParser のオプションを実行時に定義するには

1. Enterprise Designer で、Open Name Parser ステージを使用するデータフローを開きます。
2. そのデータフローを保存してエクスポーズします。
3. 編集 > データフローオプション に移動します。
4. **[データフロー オプションをステージにマッピングします]** テーブルで、Open Name Parser を展開し、必要に応じてオプションを編集します。編集するオプションのチェック ボックスをオンにしてから、**[デフォルト値]** ドロップダウンの値を変更します。
5. オプション: **[オプション ラベル]** フィールドで、オプションの名前を変更します。
6. **[OK]** を 2 回クリックします。

出力

表 135 : Open Name Parser の出力

columnName	書式	説明
AccountDescription	String	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"。
Names	String	パース済み要素のリストを含む階層フィールド。このフィールドは、[パーシング オプション] の **[結果をリストに出力]** ボックスをチェックしている場合に返されます。
会社名関係のフィールド		
FirmConjunction	String	"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	String	会社名。例えば、"Pitney Bowes"。
FirmSuffix	String	会社名の接尾語。例えば、"Co."、"Inc."
IsFirm	String	名前が、個人名ではなく、企業名であることを示します。
個人名に関係するフィールド		
Conjunction	String	名前に、"and"、"or"、"&" などの接続詞が含まれることを示します。
CultureCode	String	入力データに含まれるカルチャー コード。

columnName	書式	説明
CultureCodeUsedToParse	String	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。 **注：** Open Parser ドメインエディタを使用して独自のドメインを追加した場合、そのドメインのカルチャーとカルチャーコードもこのフィールドに表示されます。
FirstName	String	個人のファースト ネーム。
GeneralSuffix	String	個人名の一般/職業接尾語。例えば、 MD PhD 。
IsParsed	String	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	String	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	String	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	String	個人名のラスト ネーム。父方の姓が含まれます。
LeadingData	String	名前の前に付けられる、名前以外の情報。
MaturitySuffix	String	個人の世代/家族接尾語。例えば、Jr.または Sr.。
MiddleName	String	個人のミドル ネーム。

columnName	書式	説明
Name.	String	入力に指定された個人名または企業名。
NameScore	String	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	String	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	String	"Mr."、"Mrs."、"Dr." など、名前の前に付けられる情報。
TrailingData	String	名前の後に付けられる、名前以外の情報。
結合名関係のフィールド		
Conjunction2	String	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	String	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	String	結合されている 2 番目の企業名。例えば、Baltimore Gas & Electric dba Constellation Energy。
FirmSuffix2	String	結合されている 2 番目の企業の接尾語。
FirstName2	String	結合されている FirstName の 2 番目の名
FirstName3	String	結合されている FirstName の 3 番目の名。

columnName	書式	説明
GeneralSuffix2	String	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	String	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	String	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"。
LastName2	String	結合されている LastName の 2 番目の姓。
LastName3	String	結合されている LastName の 3 番目の姓。
MaturitySuffix2	String	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	String	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	String	結合されている MiddleName の 2 番目の名。
MiddleName3	String	結合されている MiddleName の 3 番目の名。
TitleOfRespect2	String	"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	String	"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。

9 - Spectrum™ Technology Platform について

このセクションの構成

Spectrum™ Technology Platform とは 580
エンタープライズ データ管理アーキテクチャ 581
Spectrum™ Technology Platformのアーキテクチャ 585
モジュールとコンポーネント 590

Spectrum™ Technology Platform とは

Spectrum™ Technology Platform は、データの正規化、検証、拡張 (価値向上) の 3 つの側面からデータの完全性、妥当性、一貫性、適時性、および正確性を高めるシステムです。データを正確かつ包括的に、最新の状態に維持することで、顧客への理解を深め、顧客とより良好な関連性を構築できます。

Spectrum™ Technology Platform は、以下の機能を実行して、データの品質を高めるビジネス ルールの設計と実装を支援します。

パーシング、名前の正規化、名前のバリデーション

正規化をきわめて正確に実行するには、一連のデータ列を複数のフィールドに分割する必要がある場合があります。Spectrum™ Technology Platform は、個人名、企業名、およびその他多くの語や略語をパースする高度なパーシング機能を備えています。また、スキャン/抽出操作のベースとして使用するカスタム表現のリストを独自に作成することもできます。Universal Name モジュールは、この機能を備えています。

重複除外統合

一意のエンティティを識別することで、レコードを統合する、重複レコードを排除する、および "最良の組み合わせ" レコードを作成できます。"最良の組み合わせ" レコードとは、別のレコードのデータを使用して作成する複合的なレコードです。Advanced Matching モジュールと Data Normalization モジュールは、この機能を備えています。

住所検証

住所検証では、管轄の郵便当局のルールを適用して、住所を標準形式に変換し、その住所が配達可能な住所であるかどうかを確認します。住所検証により、郵便料金の割引を受けやすくなり、郵便物の配達品質を高めることができます。Universal Addressing モジュールと Address Now モジュールは、この機能を備えています。

ジオコーディング

ジオコーディングとは、住所を地図上のポイント（緯度と経度）に変換する処理です。ジオコーディングは、地図製作に使用されますが、それは 1 つの使用例にすぎません。基盤を成すロケーション データがあると、ビジネス上の意思決定を行いやすくなります。処理を逆にすることで、ジオコード (緯度と経度で表現される地図上のポイント) を入力し、そのジオコードに関する住所情報を取得できます。Enterprise Geocoding モジュールは、この機能を備えています。

ロケーションインテリジェンス

ロケーション インテリジェンスは、地理関係を調査、評価、分析、およびモデル化して、データに関する新しい情報を作成します。ロケーション インテリジェンス処理を使用すると、ロケーションを検証し、情報を有益なビジネス インテリジェンスに変換できます。Location Intelligence モジュールは、この機能を備えています。

マスターデータ管理

マスター データ管理では、重要なデータ アセットの関連性を中心に捉えたマスター データ ビューを作成できます。Data Hub モジュールは、インフルエンサーと明白でない関連性の特定、詐欺行為の検出、情報の品質、統合、およびアクセシビリティを高めるのに役立ちます。

税務管轄区域の割り当て

税務管轄区域の割り当てでは、住所の地域に適用される税務管轄区域を判断します。税務管轄区域を最も正確に割り当てると、経済上のリスクや、法的義務を軽減できます。

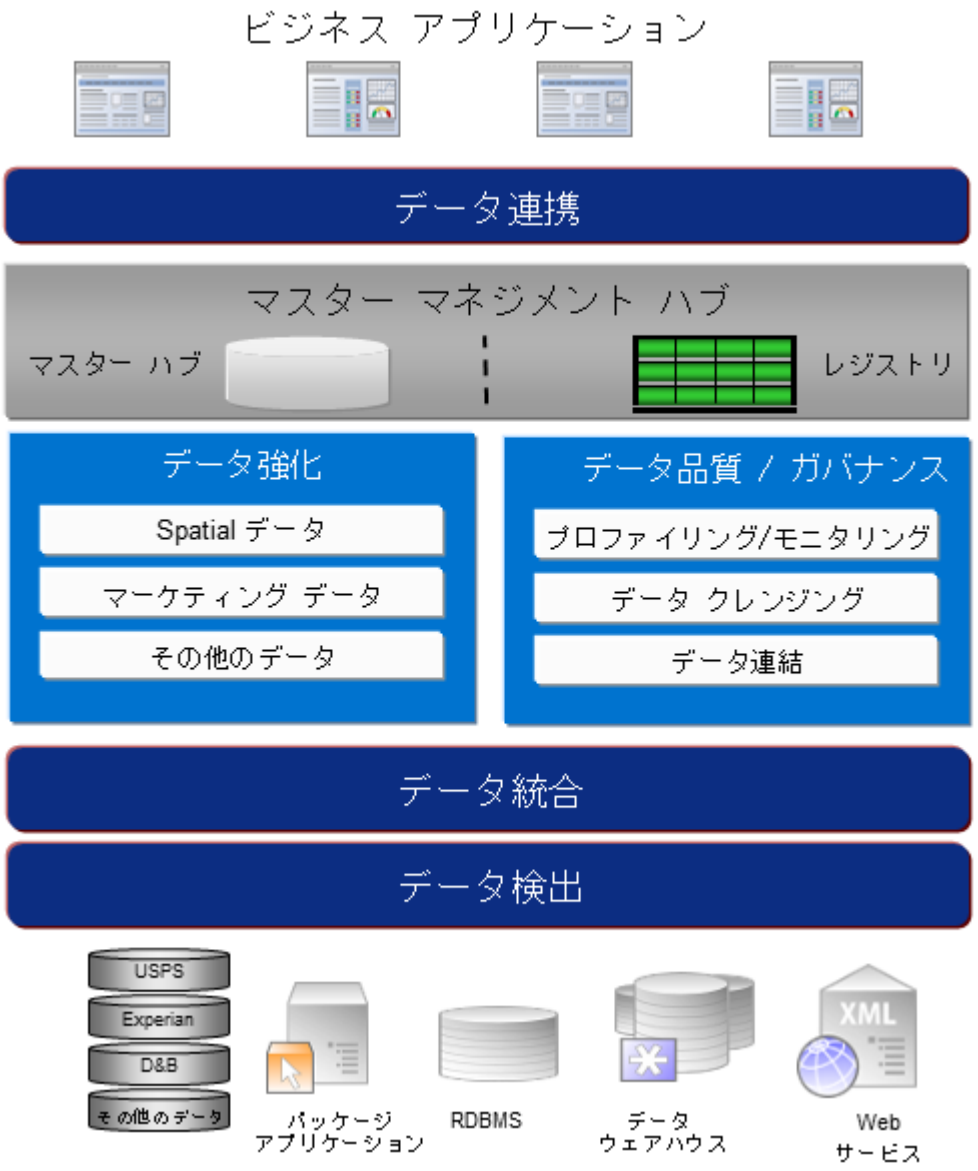
Spectrum™ Technology Platform が提供する Pitney Bowes ソフトウェアでは、最新の税務管轄区域と顧客レコードの正確な住所を統合して正確な州、郡、郡区、市、および特殊な税務管轄区域の情報をレコードに追加できます。税務管轄区域の割り当ての使用例を次に示します。

- 売上税と使用税
- 動産税
- 保険料税

Enterprise Tax モジュールは、この機能を備えています。

エンタープライズ データ管理アーキテクチャ

Spectrum™ Technology Platform では、包括的なエンタープライズ データ管理処理を構築できます。あるいは、対象をさらに絞り込んだソリューションとしてこれを活用することも可能です。次の図は、ソースからデータを取得し、データ強化およびデータ品質処理を経て、マスター データ管理ハブに引き渡す、包括的なソリューションを示したものです。MDM ハブは、データの単一のビューを作成して複数のビジネス アプリケーションに提供します。



データ検出

データ検出は、データ リソースをスキャンしてデータの状況を詳細に把握するプロセスです。Spectrum™ Technology Platform は、さまざまなデータプロファイリング手法を使用して、構造化されたデータ、構造化されていないデータ、および一部分のみ構造化されたデータをスキャンできます。スキャン結果は、会社のデータ アセットを記述するドキュメントのライブラリの生成とメタデータ リポジトリの作成に自動的に使用されます。このドキュメントと付属のメタデータ リポジトリから提供される情報を十分に吟味したうえで、データ統合、データ品質、データ制御、またはマスター データ管理プロジェクトを始めてください。

Spectrum™ Technology Platform のデータ検出モジュールの詳細については、営業担当者にお問い合わせください。

データ統合

データの状況を把握したら、次は、管理する必要があるデータへのアクセス方法を検討する必要があります。Spectrum™ Technology Platform は、複数のソースのデータに直接接続できます。また、既存のデータ アクセス手法を統合した方法で接続することもできます。データ ウェアハウス、データ品質、システム統合、移行といった多様なビジネス ニーズに対応するバッチおよびリアル タイム データ統合機能をサポートします。Spectrum™ Technology Platform は RDBMS データベース、データ ウェアハウス、XML ファイル、フラット ファイルなどのデータにアクセスできます。Spectrum™ Technology Platform は、複雑な結合や集計を含むSQL クエリをサポートし、視覚的なクエリ開発ツールを提供します。また、Spectrum™ Technology Platform は REST および SOAP Web サービスを介してデータにアクセスできます。

Spectrum™ Technology Platform は、指定されたフォルダ内の1つ以上のソースファイルの存在チェック結果に基づいてバッチ処理をトリガできます。この "ホット フォルダ" トリガは、FTP アップロードの監視と、アップロード直後の処理に役立ちます。

これらのデータ統合機能の一部には、Enterprise Data Integration モジュールのライセンスが必要です。詳細については、営業担当者に問い合わせてください。

最後に、Spectrum™ Technology Platform は SAP や Siebel などのパッケージ アプリケーションと統合可能です。

データ品質/ガバナンス

データ品質およびデータ ガバナンス処理では、重複レコード、矛盾した情報、不正確な情報がないか、データを確認します。

重複マッチングは、重複レコードの可能性や、レコード間の関連性を特定します。データが実際の名前や住所であるか、または他の種類の顧客情報であるかは関係ありません。Spectrum™ Technology Platform では、boolean 型マッチング方式、スコアリング方式、しきい値、アルゴリズム、および重みを使用する一貫したビジネスマッチルールを指定して、レコードのグループに重複が含まれているかどうかを調べることができます。Spectrum™ Technology Platform は、多種多様なカスタマイズをサポートしているため、ビジネス固有のニーズに適合するようにルールを調整できます。

重複レコードを特定したら、それらのレコードを統合することもできます。Spectrum™ Technology Platform は、重複レコードをリンクまたは結合して、収集した顧客情報から最も正確かつ包括的なレコードを作成する方法を指定できます。例えば、ある世帯のすべてのレコードに基づいて、1 つの Best-of-Breed (最良の組み合わせ) レコードを作成できます。重複の特定とその排除には、Advanced Matching モジュールを使用します。

データ品質処理では、データの正規化も行われます。正規化は、きわめて重要な処理です。レコードの照合とレコード間の関連性の識別において、最も可能性の高い結果を得るために、正規化データ要素が必要であるためです。モジュールによっては、複数のタイプの正規化を実行するものもありますが、Spectrum™ Technology Platform の Data Normalization モジュールは最も包括的な正規化機能セットを備えています。また、Universal Name モジュールは、個人名や企業名データを処理するための特定のデータ品質機能を提供します。

正規化データは、必ずしも正確なデータではありません。Spectrum™ Technology Platform は、データを既知の最新の参照データと比較して、その妥当性を確認できます。この処理に用いられるソースとしては、米国郵政公社などの規制機関、Experian やD&B などのサードパーティのデータプロバイダ、会計データなどの企業内の参照ソースがあります。Spectrum™ Technology Platform は、住所データの検証に特に優れています。世界中の 250 の国および地域の住所の検証または正規化が可能です。住所の検証を実行するモジュールには、Address Now モジュールと Universal Addressing モジュールの 2 つがあります。

どちらのモジュールがニーズに適しているかは、営業担当者と相談して判断してください。

Spectrum™ Technology Platform は、幅広いデータ品質問題を自動的に処理できますが、データスチュワードによる手動確認が適切である場合が存在します。これをサポートするために、Business Steward モジュールでは、手動確認をトリガするルールを指定するための方法と、例外レコードを確認するための Web ベースのツールが提供されています。確認および解決処理においてデータスチュワードを支援するための、Bing マップや Experian データといったサードパーティ ツールへの統合アクセスも含まれています。

データ強化(データ・エンリッチメント)

データ強化処理は、追加情報によってデータを増補します。強化は、データに詳細情報を追加するためにユーザが使用したいと考える、空間データ、マーケティング データ、または他のソースからのデータに基づいて行うことができます。例えば、顧客住所のデータベースが存在する場合、住所のジオコーディングを行って、住所の緯度/経度座標を特定し、その座標をレコードの一部として保存することができます。これによって顧客データは、顧客に最も近い銀行支店の検索など、多様な空間分析に使用できるようになります。Spectrum™ Technology Platform では、データをさまざまな情報で強化できます。例えば、ジオコーディング (Enterprise Geocoding モジュールを使用)、税務管轄区域の割り当て (Enterprise Tax モジュール)、地理空間分析 (Location Intelligence モジュール)、2 点間の車移動または徒歩経路 (Enterprise Routing モジュール) の情報を利用できます。

マスター データ管理ハブ

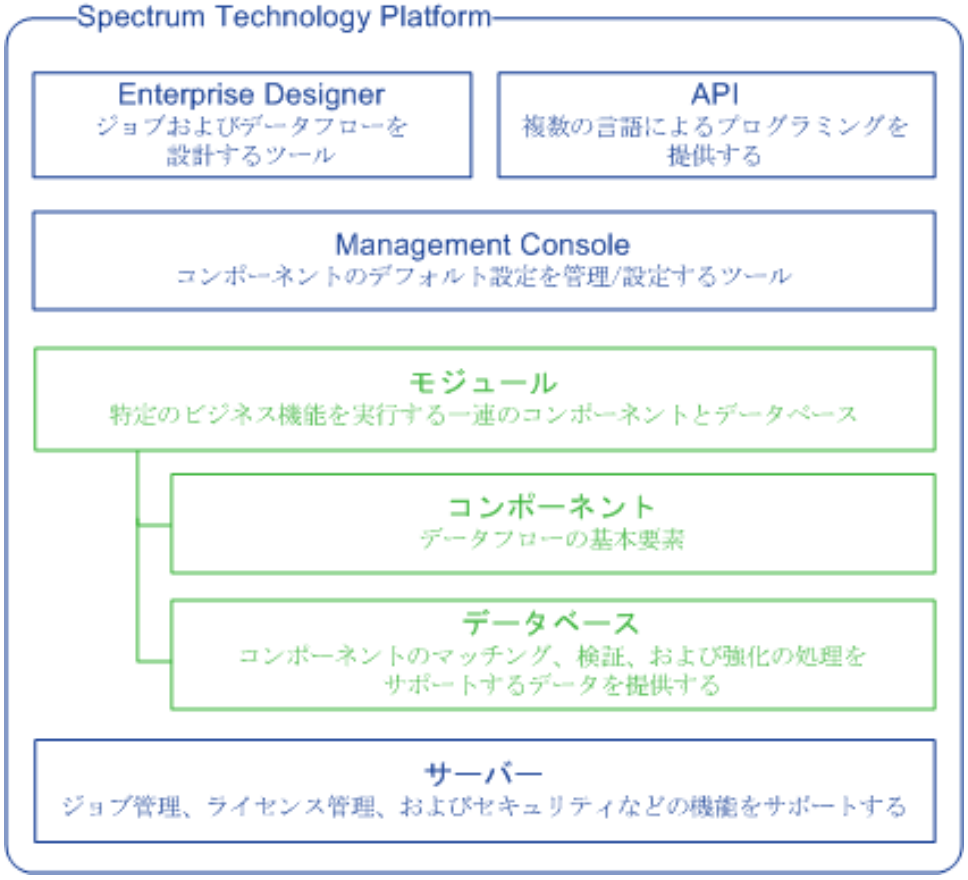
マスター データ管理 (MDM) ハブは、エンティティと、その役割、処理、やり取りの間の複雑な関連性の迅速なモデリングを可能にします。ソーシャル ネットワーク分析機能が組み込まれており、インフルエンサー (influencer) の理解、チャーンの予測、明白でない関係や不正パターンの検出、レコメンデーションを支援します。

Spectrum™ Technology Platform は、MDM ハブに対する 2 つのアプローチをサポートします。マスター ハブのアプローチでは、データは単一の MDM データベースに維持され、アプリケーションは MDM データベースからデータにアクセスします。レジストリのアプローチでは、データは各ビジネス アプリケーションに維持され、MDM ハブ レジストリに、関連レコードの検索に用いられるキーが含まれます。例えば、顧客のレコードが、注文入力データベースと顧客サポートデータベースに存在する場合があります。この場合MDM レジストリには、両方の場所の顧客データへのアクセスに使用できる単一のキーが含まれます。

Data Hub モジュールが、MDM 機能を提供します。

Spectrum™ Technology Platformのアーキテクチャ

Spectrum™ Technology Platformが提供するPitney Bowes は、いくつかのモジュールを実行するサーバーで構成されます。これらのモジュールは、住所のバリデーション、ジオコーディング、高度なパーシング等、さまざまな機能を備えています。次の図に、Spectrum™ Technology Platform のアーキテクチャを示します。



サーバー

このサーバーがSpectrum™ Technology Platformの基盤となります。サーバーは、データを処理し、リポジトリ データを同期し、通信を管理します。また、ジョブ管理およびセキュリティ機能も提供します。

モジュール

モジュールは、特定の機能を実行する機能群です。例えば、Universal Addressing モジュールは、郵便規格に準拠するように住所を正規化します。Enterprise Tax モジュールは、その住所に該当する税務管轄区域を判定します。共通のビジネス問題を解決する各種モジュールがまとめられて、バンドルとしてライセンス供与されています。

コンポーネント

モジュールは、特定の機能をフロー内で実行するか、サービスとして実行するコンポーネントで構成されます。例えば、Enterprise Geocoding モジュールの Geocode US Address コンポーネントは、住所を地図上のポイント (緯度と経度) に変換して返します。Universal Addressing モジュールの Get City State Province は、郵便番号が示す都市および州/省を返します。

システムで利用できるコンポーネントは、Spectrum™ Technology Platformのどのバンドルのライセンスを取得したかによって異なります。

データベース

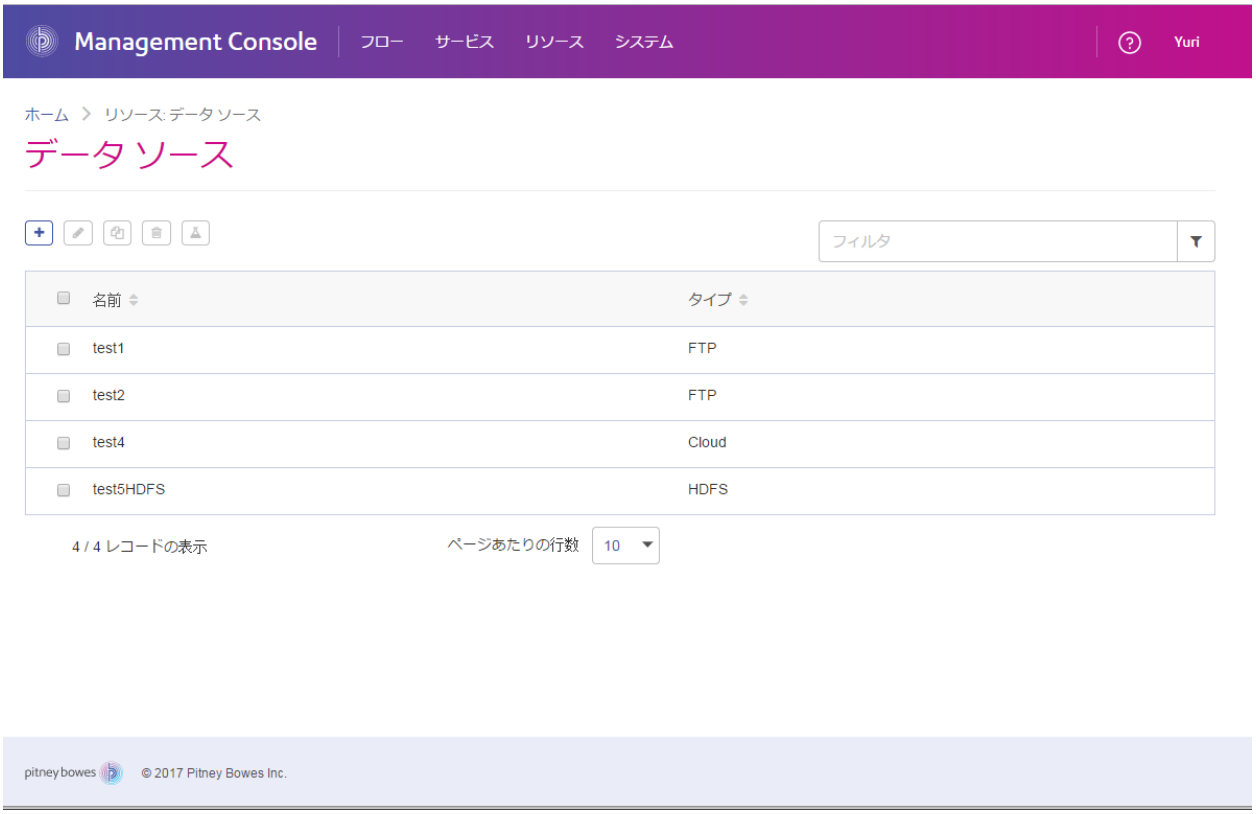
一部のモジュールは、参照データを含むデータベースに依存します。例えば、Universal Addressing モジュールは、米国の住所を検証して正規化するために米国郵政公社 (USPS) のデータにアクセスする必要があります。データベースは個別にインストールされ、一部のデータベースは最新データを提供するために定期的に更新されます。

モジュールには、必須データベースとオプションのデータベースがあります。オプションのデータベースは、Spectrum™ Technology Platformの処理を強化することのできる特定の機能に必要なデータを提供します。

Management Console

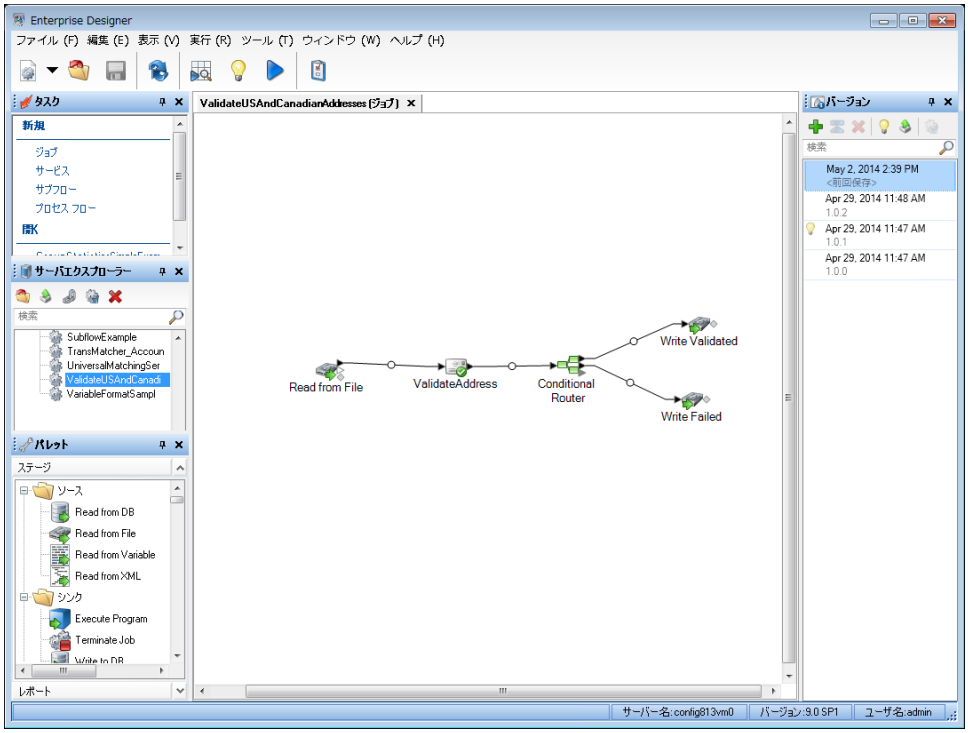
Management Console は、Spectrum™ Technology Platformを管理するためのツールです。Management Console で実行できる操作は、次のとおりです。

- Spectrum™ Technology Platformとデータの間の接続を定義する。
- サービスやフローのデフォルト設定を指定する。
- 権限やパスワード等、ユーザ アカウントを管理する。
- ログを表示する。
- ライセンス有効期限情報を含めて、ライセンスを表示する。



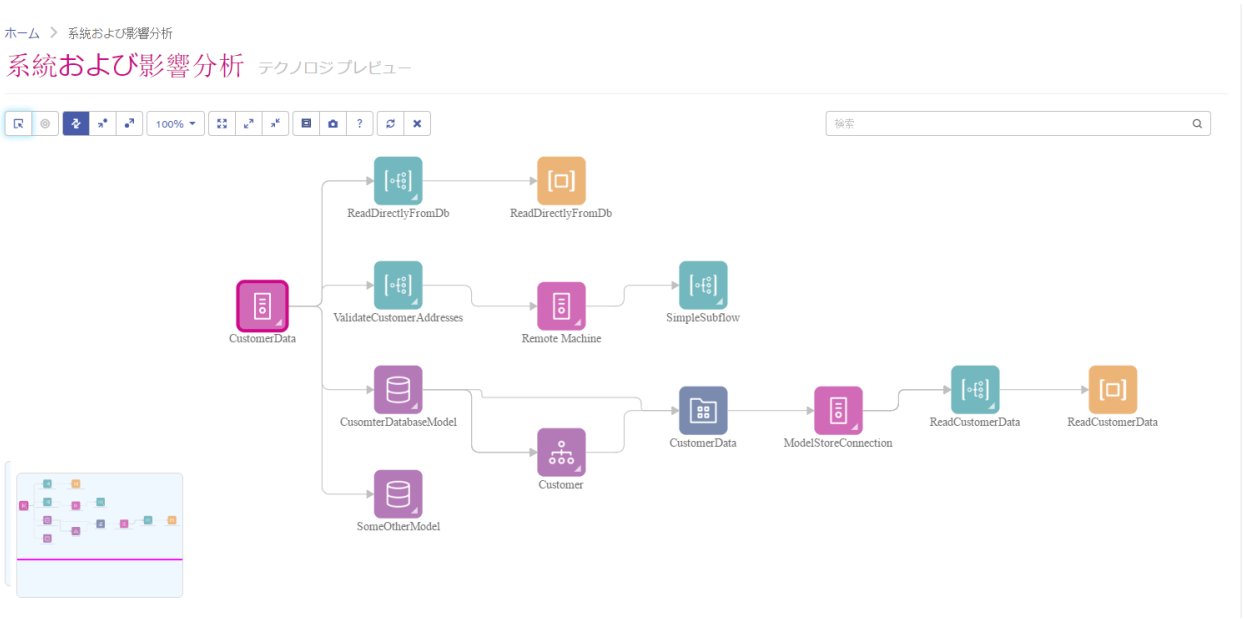
Enterprise Designer

Enterprise Designer は、Spectrum™ Technology Platformのジョブ、サービス、サブフロー、およびプロセス フローを作成するためのツールです。ドラッグアンドドロップ インターフェイスを利用して、複雑なデータフローをグラフィカルに、容易に作成できます。



Metadata Insights

Metadata Insights を使用すると、適切な時間に収集された正確なデータに基づくビジネス分析を得るために必要な制御が可能になります。Metadata Insights を使用して、データ モデルを開発し、ソースからビジネス アプリケーションまでのデータの流れを表示し、プロファイリングによってデータの品質を評価します。この分析を活用すれば、特定のビジネスの課題を解決できるデータ リソースの特定、ビジネス全体でデータの有益性と一貫性を向上するプロセスの適合と最適化、およびデータの問題のトラブルシューティングを行うことができます。



Web サービスと API

Web サービスとプログラミング API を使用して、Spectrum™ Technology Platformの機能を独自のアプリケーションに統合することができます。これらのインターフェイスは、シンプルな統合とレコード処理の効率化を可能とし、将来のバージョンの下位互換性をサポートします。

Spectrum™ Technology PlatformAPI は、以下の言語で使用可能です。

- C
- C++
- COM
- Java
- .NET

Web サービスは、SOAP および REST を介して提供されています。

モジュールとコンポーネント

表 136 : モジュールとコンポーネント

モジュール	説明	コンポーネント
Address Now モジュール	米国以外の住所についての高度なバリデーションと正規化を提供します。また、別の住所処理も提供します。	Build Global Address Get Global Candidate Addresses Validate Global Address
Advanced Matching モジュール	入力ファイル内や入力ファイル間でレコードを照合します。	Best Of Breed Candidate Finder Duplicate Synchronization Filter Interflow Match Intraflow Match Match Key Generator Transactional Match
Analytics Scoring モジュール	入力ファイル内や入力ファイル間でレコードを照合します。	PMML Model Scoring Miner データセットからの読み込み Miner データセットへの書き出し
Business Steward モジュール	例外レコードを特定し、例外レコードを手動で確認するためのブラウザベースのツールを提供します。	Exception Monitor Read Exceptions Write Exceptions
Country Identifier	国名を、または郵便番号と州/省の組み合わせを受け取って、2 文字の ISO 国コード、3 文字の Universal Postal Union (UPU) コード、および国名を英語で返します。	Country Identifier

モジュール	説明	コンポーネント
Data Hub モジュール	データをリンクおよび分析して、関係と傾向を識別します。	Write to Hub Read From Hub Query Hub Graph Visualization
Data Integration モジュール	データ ウェアハウジング、データ品質、システム統合、および移行に便利な機能を備えています。	Field Selector Generate Time Dimension Query Cache Write to Cache
Data Normalization モジュール	データの不整合を取り除きます。	Advanced Transformer Open Parser Table Lookup Transliterator
Enterprise Data Integration	データ ウェアハウジングや、データ品質、システム統合、移行などのさまざまなビジネス ニーズに対応するために複数のソースのデータに接続します。	Call Stored Procedure Field Selector Generate Time Dimension Query Cache Write to Cache
Enterprise Geocoding モジュール	住所を表す地図上のポイント（緯度と経度）に変換します。また、指定された緯度と経度を住所に変換します。	Geocode Address AUS Geocode Address GBR Geocode Address Global Geocode Address World Geocode US Address GNAF PID Location Search Reverse APN Lookup Reverse Geocode Address Global Reverse Geocode US Location

モジュール	説明	コンポーネント
Enterprise Routing モジュール	自動車移動または徒歩の経路を取得し、移動時間および移動距離を計算し、始点から一定の時間または距離内に含まれる領域を特定します。	Get Route Data Get Travel Boundary Get Travel Cost Matrix Get Travel Directions Persistent Update
Enterprise Tax モジュール	特定の地域に適用する税務管轄区域を決定します。	Assign GeoTAX Info Calculate Distance
GeoConfidence モジュール	指定された領域に住所または交差点が含まれる可能性を判定します。	Geo Confidence Surface CreatePointsConvexHull
Global Addressing モジュール	米国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。また、入力の途中で住所を自動的に予測し、入力に基づく候補を直ちに返します。機械学習の手法を使用して、郵便住所文字列を個々の住所要素に分割します。	Global Address Validation Global Type Ahead グローバル住所パーサー
Global Geocoding モジュール	住所を表す地図上のポイント（緯度と経度）に変換します。また、指定された緯度と経度を住所に変換します。	Global Geocode Global Reverse Geocode
Global Sentry	政府から提供される、各国のデータを含む警戒リストとトランザクションとの照合を試みます。	Global Sentry Global Sentry Address Check Global Sentry ID Number Check Global Sentry Name Check Global Sentry Other Data Check

モジュール	説明	コンポーネント
Location Intelligence モジュール	さまざまな地理空間データベースと照合して Point In Polygon と半径分析を実行します。	Closest Site Find Nearest Point In Polygon Query Spatial Data Read Spatial Data Spatial Calculator Spatial Union
Machine Learning モジュール	数値データをビニングし、教師ありと教師なしの機械学習モデルを適合し、これらのモデルでデータをスコアリングするための機能を提供します。	Binning Binning Lookup Java Model Scoring K-Means Clustering Linear Regression Logistic Regression 主成分分析 Random Forest Classification Random Forest Regression
Metadata Insights	適切な時間に収集された正確なデータに基づくビジネス分析を得るために必要な制御を可能にします。データ モデルを開発し、ソースからビジネス アプリケーションまでのデータの流れを表示し、プロファイリングによってデータの品質を評価できます。この分析を活用すれば、特定のビジネスの課題の解決に使用すべきデータ リソースの特定や、ビジネス全体でデータの有益性と一貫性を向上するプロセスの最適化を行うことができます。	モデル (論理および物理) Model Store プロファイル 系統および影響分析

モジュール	説明	コンポーネント
SAP モジュール	Spectrum™ Technology Platform と SAP Customer Relationship Management モジュール アプリケーションとの連携を有効にします。	SAP Generate Match Key SAP Generate Match Score SAP Generate Search Key SAP Generate Search Key Constant SAP Generate Search Key Metaphone SAP Generate Search Key Substring SAP Validate Address With Candidates
Siebel モジュール	Spectrum™ Technology Platform と Siebel アプリケーションとの連携を有効にします。	Siebel Generate Match Key Siebel Generate Match Score Siebel Generate Search Key Siebel Business Name Standardization Siebel Standardize Name. Siebel Geocode US Address With Candidates Siebel Geocode US Address With No Candidates Siebel Get Global Candidate Addresses Siebel Validate Address With Candidates Siebel Validate Address With No Candidates
Universal Addressing モジュール	郵便当局の規格に従って、住所を正規化してバリデーションを実行します。	Get Candidate Addresses Get City State Province Get Postal Codes Validate Address Validate Address AUS Validate Address Global

モジュール	説明	コンポーネント
Universal Name モジュール	個人名、企業名、住所、およびその他多くの語や略語をパースします。	Name Parser (非推奨) Name Variant Finder Open Name Parser

付録

このセクションの構成

ISO 国コードとモジュール サポート 597

A - 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 国コードとモジュール サポート 598

ISO 国コードとモジュール サポート

この表に、各国の ISO コードと、各国の住所作成、ジオコーディング、およびルーティングをサポートするモジュールを示します。

Enterprise Geocoding モジュールにアフリカ (30 か国)、中東 (8 か国)、ラテン アメリカ (20 か国) のデータベースが含まれていることに注意してください。これらのデータベースは、国別のジオコーディング データベースがない、各地域の比較的小さな国をカバーします。[サポートされるモジュール] 列は、これらのアフリカ、中東、ラテン アメリカ データベースに含まれる国を示しています。

また、Geocode Address World データベースは、すべての国について地図上の限定的な郵便ジオコーディング (ストリート レベルではない) を提供します。

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Afghanistan	AF	AFG	Universal Addressing モジュール
Aland Islands	AX	ALA	Universal Addressing モジュール
Albania	AL または SQ (Routing)	ALB	Universal Addressing モジュール Enterprise Geocoding モジュール Enterprise Routing モジュール
Algeria	DZ	DZA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
American Samoa	AS	ASM	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Andorra	AD	AND	Enterprise Geocoding モジュール(アンドラは、スペインのジオコーダで扱われています)。 Universal Addressing モジュール GeoComplete モジュール
Angola	AO	AGO	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Anguilla	AI	AIA	Universal Addressing モジュール
Antarctica	AQ	ATA	Universal Addressing モジュール
Antigua And Barbuda	AG	ATG	Universal Addressing モジュール
アルゼンチン	AR	ARG	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール
Armenia	AM	ARM	Universal Addressing モジュール
Aruba	AW	ABW	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
オーストラリア	AU	AUS	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール
オーストリア	AT	AUT	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Azerbaijan	AZ	AZE	Universal Addressing モジュール
バハマ	BS	BHS	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール
Bahrain	BH	BHR	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール
Bangladesh	BD	BGD	Universal Addressing モジュール
Barbados	BB	BRB	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Belarus	BY	BLR	Universal Addressing モジュール Enterprise Routing モジュール
ベルギー	BE	BEL	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Belize	BZ	BLZ	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Benin	BJ	BEN	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Bermuda	BM	BMU	Universal Addressing モジュール Enterprise Routing モジュール
Bhutan	BT	BTN	Universal Addressing モジュール
Bolivia, Plurinational State Of	BO	BOL	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Bonaire, Saint Eustatius And Saba	BQ	BES	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Bosnia And Herzegovina	BA	BIH	Universal Addressing モジュール Enterprise Routing モジュール Enterprise Geocoding モジュール
Botswana	BW	BWA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
ブーベ島	BV	BVT	Universal Addressing モジュール
ブラジル	BR	BRA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
British Indian Ocean Territory	IO	IOT	Universal Addressing モジュール
Brunei Darussalam	BN	BRN	Enterprise Geocoding モジュール Universal Addressing モジュール
Bulgaria	BG	BGR	Universal Addressing モジュール
Burkina Faso	BF	BFA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Burundi	BI	BDI	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Cambodia	KH	KHM	Universal Addressing モジュール
Cameroon	CM	CMR	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
カナダ	CA	CAN	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Cape Verde	CV	CPV	Universal Addressing モジュール
Cayman Islands	KY	CYM	Universal Addressing モジュール
Central African Republic	CF	CAF	Universal Addressing モジュール
Chad	TD	TCD	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
チリ	CL	CHL	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール GeoComplete モジュール
中国	CN または zh_CN (Routing)	CHN	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール
Christmas Island	CX	CXR	Universal Addressing モジュール
Cocos (Keeling) Islands	CC	CCK	Universal Addressing モジュール
Colombia	CO	COL	Universal Addressing モジュール
Comoros	KM	COM	Universal Addressing モジュール
Congo	CG	COG	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Congo, The Democratic Republic Of The	CD	COD	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Cook Islands	CK	COK	Universal Addressing モジュール
Costa Rica	CR	CRI	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Côte d'Ivoire	CI	CIV	Universal Addressing モジュール
クロアチア	HR	HRV	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール
Cuba	CU	CUB	Enterprise Geocoding モジュール (ラテン アメリカ) Enterprise Routing モジュール Universal Addressing モジュール
Curacao	CW	CUW	Universal Addressing モジュール
Cyprus	CY	CYP	Enterprise Geocoding モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
チェコ共和国	CZ または CS (Routing)	CZE	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール GeoComplete モジュール
デンマーク	DK	DNK	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Djibouti	DJ	DJI	Universal Addressing モジュール
Dominica	DM	DMA	Universal Addressing モジュール
Dominican Republic	DO	DOM	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Ecuador	EC	ECU	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Egypt	EG	EGY	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
El Salvador	SV	SLV	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Equatorial Guinea	GQ	GNQ	Universal Addressing モジュール
Eritrea	ER	ERI	Universal Addressing モジュール
エストニア	EE	EST	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール
Ethiopia	ET	ETH	Universal Addressing モジュール
フォークランド諸島 (マルビナス)	FK	FLK	Universal Addressing モジュール
Faroe Islands	FO	FRO	Universal Addressing モジュール
Fiji	FJ	FJI	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
フィンランド	FI	FIN	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
フランス	FR	FRA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
French Guiana	GF	GUF	Enterprise Geocoding モジュール (*フランス領ギアナは、フランスのジオコーダで扱われています*)。 Universal Addressing モジュール
French Polynesia	PF	PYF	Universal Addressing モジュール
French Southern Territories	TF	ATF	Universal Addressing モジュール
Gabon	GA	GAB	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Gambia	GM	GMB	Universal Addressing モジュール
Georgia	GE	GEO	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
ドイツ	DE	DEU	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Ghana	GH	GHA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール
Gibraltar	GI	GIB	Enterprise Geocoding モジュール (*ジブラルタルは、スペインのジオコーダで扱われています*)。 Universal Addressing モジュール
ギリシャ	GR	GRC	Enterprise Geocoding モジュール Universal Addressing モジュール
Greenland	GL	GRL	Universal Addressing モジュール
Grenada	GD	GRD	Universal Addressing モジュール
Guadeloupe	GP	GLP	Enterprise Geocoding モジュール (*グアドループは、フランスのジオコーダで扱われています*)。 Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Guam	GU	GUM	Universal Addressing モジュール
Guatemala	GT	GTM	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Guernsey	GG	GGY	Universal Addressing モジュール
Guinea	GN	GIN	Universal Addressing モジュール
Guinea-Bissau	GW	GNB	Universal Addressing モジュール
Guyana	GY	GUY	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Haiti	HT	HTI	Universal Addressing モジュール
Heard Island and McDonald Islands	HM	HMD	Universal Addressing モジュール
法王聖座 (バチカン市国)	VA	VAT	Enterprise Geocoding モジュール (*バチカンは、イタリアのジオコーダで扱われています*)。 Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Honduras	HN	HND	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
香港	HK	HKG	Enterprise Geocoding モジュール Universal Addressing モジュール
ハンガリー	HU	HUN	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Iceland	IS	ISL	Enterprise Geocoding モジュール Universal Addressing モジュール
インド	IN	IND	Enterprise Geocoding モジュール Universal Addressing モジュール
インドネシア	ID	IDN	Enterprise Geocoding モジュール Universal Addressing モジュール
イラン・イスラム共和国	IR	IRN	Universal Addressing モジュール
Iraq	IQ	IRQ	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
アイルランド	IE	IRL	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Isle Of Man	IM	IMN	Universal Addressing モジュール
Israel	IL	ISR	Universal Addressing モジュール Enterprise Routing モジュール
イタリア	IT	ITA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Jamaica	JM	JAM	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
日本	JP	JPN	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Jersey	JE	JEY	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Jordan	JO	JOR	Universal Addressing モジュール Enterprise Geocoding モジュール (中東) Enterprise Routing モジュール
Kazakhstan	KZ	KAZ	Universal Addressing モジュール
Kenya	KE	KEN	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール
Kiribati	KI	KIR	Universal Addressing モジュール
Korea, Democratic People's Republic Of	KP	PRK	Universal Addressing モジュール
Korea, Republic Of	KR	KOR	Universal Addressing モジュール
Kosovo	KS	KOS	Enterprise Geocoding モジュール Universal Addressing モジュール GeoComplete モジュール
Kuwait	KW	KWT	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Kyrgyzstan	KG	KGZ	Universal Addressing モジュール
Lao People's Democratic Republic	LA	LAO	Universal Addressing モジュール
ラトビア	LV	LVA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール
Lebanon	LB	LBN	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール
Lesotho	LS	LSO	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール
Liberia	LR	LBR	Universal Addressing モジュール
Libyan Arab Jamahiriya	LY	LBY	Universal Addressing モジュール
Liechtenstein	LI	LIE	Enterprise Geocoding モジュール (リヒテンシュタインは、スイスのジオコーダで扱われています)。 Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
リトアニア	LT	LTU	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール
Luxembourg	LU	LUX	Enterprise Geocoding モジュール (*ルクセンブルクは、ベルギーのジオコーダで扱われています*)。 Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Macao	MO	MAC	Enterprise Geocoding モジュール Universal Addressing モジュール
Macedonia, Former Yugoslav Republic Of	MK	MKD	Enterprise Geocoding モジュール Universal Addressing モジュール
Madagascar	MG	MDG	Universal Addressing モジュール
Malawi	MW	MWI	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
マレーシア	MY	MYS	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Maldives	MV	MDV	Universal Addressing モジュール
Mali	ML	MLI	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Malta	ML	MLT	Enterprise Geocoding モジュール Universal Addressing モジュール
Marshall Islands	MH	MHL	Universal Addressing モジュール
Martinique	MQ	MTQ	Enterprise Geocoding モジュール (マルティニークは、フランスのジオコーダで扱われています)。 Universal Addressing モジュール
Mauritania	MR	MRT	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Mauritius	MU	MUS	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Mayotte	YT	MYT	Enterprise Geocoding モジュール (マヨットは、フランスのジオコーダで扱われています)。 Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
メキシコ	MX	MEX	Enterprise Geocoding モジュール Universal Addressing モジュール
Micronesia, Federated States Of	FM	FSM	Universal Addressing モジュール
Moldova, Republic Of	MD	MDA	Universal Addressing モジュール Enterprise Routing モジュール
Monaco	MC	MCO	Enterprise Geocoding モジュール (*モナコはフランスのジオコーダで扱われています*)。 Universal Addressing モジュール
Mongolia	MN	MNG	Universal Addressing モジュール
Montenegro	ME	MNE	Enterprise Geocoding モジュール Universal Addressing モジュール
Montserrat	MS	MSR	Universal Addressing モジュール
Morocco	MA	MAR	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Mozambique	MZ	MOZ	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール
Myanmar	MM	MMR	Universal Addressing モジュール
Namibia	NA	NAM	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Nauru	NR	NRU	Universal Addressing モジュール
Nepal	NP	NPL	Universal Addressing モジュール
Netherlands	NL	NLD	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
New Caledonia	NC	NCL	Universal Addressing モジュール
ニュージーランド	NZ	NZL	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Nicaragua	NI	NIC	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Niger	NE	NER	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Nigeria	NG	NGA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Niue	NU	NIU	Universal Addressing モジュール
Norfolk Island	NF	NFK	Universal Addressing モジュール
Northern Mariana Islands	MP	MNP	Universal Addressing モジュール
ノルウェー	NO	NOR	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Oman	OM	OMN	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Pakistan	PK	PAK	Universal Addressing モジュール
Palau	PW	PLW	Universal Addressing モジュール
Palestinian Territory, Occupied	PS	PSE	Universal Addressing モジュール
Panama	PA	PAN	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Papua New Guinea	PG	PNG	Universal Addressing モジュール
Paraguay	PY	PRY	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Peru	PE	PER	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Philippines	PH	PHL	Enterprise Geocoding モジュール Universal Addressing モジュール Enterprise Routing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Pitcairn	PN	PCN	Universal Addressing モジュール
ポーランド	PL	POL	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
ポルトガル	PT	PRT	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Puerto Rico	PR	PRI	Universal Addressing モジュール
Qatar	QA	QAT	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール
Reunion	RE	REU	Enterprise Geocoding モジュール (*レユニオンはフランスのジオコーダで扱われています*)。 Universal Addressing モジュール
Romania	RO	ROU	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Russian Federation	RU	RUS	Enterprise Geocoding モジュール Universal Addressing モジュール GeoComplete モジュール
Rwanda	RW	RWA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
サン・バルテルミー島	BL	BLM	Universal Addressing モジュール
セントヘレナ、アセンションおよびトリスタン・ダ・クーニャ	SH	SHE	Universal Addressing モジュール
Saint Kitts and Nevis	KN	KNA	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Saint Lucia	LC	LCA	Universal Addressing モジュール
Saint Martin (French Part)	MF	MAF	Universal Addressing モジュール
Saint Pierre and Miquelon	PM	SPM	Universal Addressing モジュール
セントビンセントおよびグレナディーン諸島	VC	VCT	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Samoa	WS	WSM	Universal Addressing モジュール
San Marino	SM	SMR	Enterprise Geocoding モジュール (サンマリノは、イタリアのジオコーダで扱われています)。 Universal Addressing モジュール
サントメ・プリンシペ	ST	STP	Universal Addressing モジュール
Saudi Arabia	SA	SAU	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール
Senegal	SN	SEN	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Serbia	RS	SRB	Enterprise Geocoding モジュール Universal Addressing モジュール
Seychelles	SC	SYC	Universal Addressing モジュール
Sierra Leone	SL	SLE	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
シンガポール	SG	SGP	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Sint Maarten (Dutch Part)	SX	SXM	Universal Addressing モジュール
スロバキア	SK	SVK	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
スロベニア	SI	SVN	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Solomon Islands	SB	SLB	Universal Addressing モジュール
Somalia	SO	SOM	Universal Addressing モジュール
南アフリカ	ZA	ZAF	Enterprise Geocoding モジュール Universal Addressing モジュール GeoComplete モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
South Georgia And The South Sandwich Islands	GS	SGS	Enterprise Geocoding モジュール Universal Addressing モジュール
南スーダン	SS	SSD	Universal Addressing モジュール
スペイン	ES	ESP	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Sri Lanka	LK	LKA	Universal Addressing モジュール
Sudan	SD	SDN	Universal Addressing モジュール
Suriname	SR	SUR	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Svalbard And Jan Mayen	SJ	SJM	Universal Addressing モジュール
Swaziland	SZ	SWZ	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
スウェーデン	SE	SWE	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
スイス	CH	CHE	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
Syrian Arab Republic	SY	SYR	Universal Addressing モジュール
Taiwan, Province of China	TW または zh_TW (Routing)	TWN	Universal Addressing モジュール Enterprise Routing モジュール
Tajikistan	TJ	TJK	Universal Addressing モジュール
Tanzania, United Republic Of	TZ	TZA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール Enterprise Routing モジュール
タイ	TH	THA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Timor-Leste	TL	TLS	Universal Addressing モジュール
Togo	TG	TGO	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Tokelau	TK	TKL	Universal Addressing モジュール
Tonga	TO	TON	Universal Addressing モジュール
Trinidad and Tobago	TT	TTO	Enterprise Geocoding モジュール (ラテン アメリカ) Universal Addressing モジュール
Tunisia	TN	TUN	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
トルコ	TR	TUR	Enterprise Geocoding モジュール Universal Addressing モジュール GeoComplete モジュール
Turkmenistan	TM	TKM	Universal Addressing モジュール
Turks And Caicos Islands	TC	TCA	Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Tuvalu	TV	TUV	Universal Addressing モジュール
Uganda	UG	UGA	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
ウクライナ	UA	UKR	Enterprise Geocoding モジュール Universal Addressing モジュール
United Arab Emirates	AE	ARE	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール
United Kingdom	GB	GBR	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
米国	US	USA	Enterprise Geocoding モジュール Enterprise Routing モジュール Universal Addressing モジュール GeoComplete モジュール
合衆国領有小離島	UM	UMI	Universal Addressing モジュール
ウルグアイ	UY	URY	Enterprise Geocoding モジュール Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Uzbekistan	UZ	UZB	Universal Addressing モジュール
Vanuatu	VU	VUT	Universal Addressing モジュール
Venezuela, Bolivarian Republic Of	VE	VEN	Enterprise Geocoding モジュール Universal Addressing モジュール
Viet Nam	VN	VNM	Universal Addressing モジュール
Virgin Islands, British	VG	VGB	Universal Addressing モジュール
Virgin Islands, U.S.	VI	VIR	Universal Addressing モジュール
Wallis and Futuna	WF	WLF	Universal Addressing モジュール
Western Sahara	EH	ESH	Universal Addressing モジュール
Yemen	YE	YEM	Enterprise Geocoding モジュール (中東) Universal Addressing モジュール

ISO 国名	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	サポートされるモジュール
Zambia	ZM	ZMB	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール
Zimbabwe	ZW	ZWE	Enterprise Geocoding モジュール (アフリカ) Universal Addressing モジュール

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