

# **ACT 距離計算サービス**

Version 2.2

## **ACT 距離計算サービス プログラミング概要**

アドバンスド・コア・テクノロジー株式会社

## ACT 距離計算サービス プログラミング 概要

2005年	1月21日	初版発行
2010年	7月27日	第8版発行
2010年	8月30日	第9版発行
2010年	11月15日	第10版発行
2011年	2月1日	第11版発行
2011年	9月1日	第12版発行
2011年	11月12日	第13版発行

編著者・発行人 アドバンスド・コア・テクノロジー株式会社

〒105-0004 東京都港区新橋3-7-4 赤レンガ通りビル2F

電話 03-5512-9021 FAX 03-5512-9022

本書に記載されている事項は、予告なしに変更されることがあります。

アドバンスド・コア・テクノロジー株式会社は本書に記載されている事項に関して一切の責任を負いかねますのでご了承ください。

本書の一部または全部をアドバンスド・コア・テクノロジー株式会社の書面による承諾なしに複製することは禁じられています。

Copyright (C) 2004-2011 by Advanced Core Technologies, Inc.

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Windows, Visual Studio, Visual Basic, Visual C++, Visual C# は米国マイクロソフト社の登録商標です。

本書掲載の製品または製品名称は各社の商標または登録商標です。

# ACT 距離計算サービス プログラミング概要 目次

1. ACT 距離計算サービスの概要.....	1-1
2. ACT 距離計算サービスのサーバ構成.....	2-1
3. サービスの種類.....	3-1
4. 使用する住所データ .....	4-1
5. プログラミング方法 .....	5-1
6. WSDL ファイルを利用したプログラミング .....	6-1
6. 1    呼び出し手順.....	6-1
6. 2    WSDL ファイル .....	6-3
6. 3    管理サービス.....	6-4
6. 3. 1    計算用道路データ情報取得サービス.....	6-4
6. 3. 2    距離計算サーバ取得サービス .....	6-6
6. 3. 3    住所検索サーバ取得サービス .....	6-8
6. 3. 4    ユーザ情報取得サービス .....	6-10
6. 4    距離計算サービス.....	6-12
6. 4. 1    ルート計算サービス .....	6-12
6. 4. 2    最短ルート計算サービス .....	6-19
6. 4. 3    直線距離計算サービス.....	6-26
6. 4. 4    到達圏／流入圏計算サービス .....	6-30
6. 4. 5    片道一括計算サービス.....	6-38
6. 4. 6    最寄ノード取得サービス .....	6-44
6. 4. 7    近傍ノード列挙サービス .....	6-49
6. 4. 8    最寄リンク取得サービス .....	6-54
6. 4. 9    道路速度取得サービス.....	6-58
6. 4. 10    道路速度設定サービス.....	6-61
6. 4. 11    道路データ情報取得サービス .....	6-64
6. 4. 12    距離計算サービス用メッセージ文字列取得サービス .....	6-66
6. 5    住所検索サービス.....	6-67
6. 5. 1    住所検索サービス .....	6-67
6. 5. 2    住所コード検索サービス .....	6-71
6. 5. 3    下位住所数取得サービス .....	6-75
6. 5. 4    下位住所列挙サービス.....	6-78
6. 5. 5    全住所取得サービス .....	6-83
6. 5. 6    最寄住所取得サービス.....	6-87

6. 5. 7	郵便番号検索サービス.....	6-91
6. 5. 8	郵便番号対応住所数サービス.....	6-94
6. 5. 9	郵便番号対応住所列举サービス .....	6-97
6. 5. 10	住所検索サービス用メッセージ文字列取得サービス ...	6-100
6. 6	旧サービス .....	6-101
6. 6. 1	距離計算サービス .....	6-101
6. 6. 2	到達圏／流入圏計算サービス .....	6-115
6. 6. 3	到達圏／流入圏計算サービス（2） .....	6-120
6. 6. 4	片道一括計算サービス .....	6-128
6. 6. 5	片道一括計算サービス（2） .....	6-134
6. 6. 6	住所検索サービス .....	6-140
6. 7	結果コード一覧 .....	6-148
6. 8	Visual Studio .NET での定数定義 DLL の利用 .....	6-153
6. 9	サンプルコード .....	6-155
7.	ブリッジ DLL を利用したプログラミング .....	7-1
7. 1	概要 .....	7-1
7. 2	呼び出しシーケンス .....	7-2
7. 3	関数一覧 .....	7-6
7. 4	関数仕様 .....	7-10
7. 5	定数定義 .....	7-74
7. 6	構造体定義 .....	7-76
7. 7	サンプルコード .....	7-86
7. 8	エラーコード一覧 .....	7-92
Appendix A	: ポリゴン修正機能 .....	A-1
Appendix B	: ポリゴン始点包含機能 .....	B-1

## 改版履歴

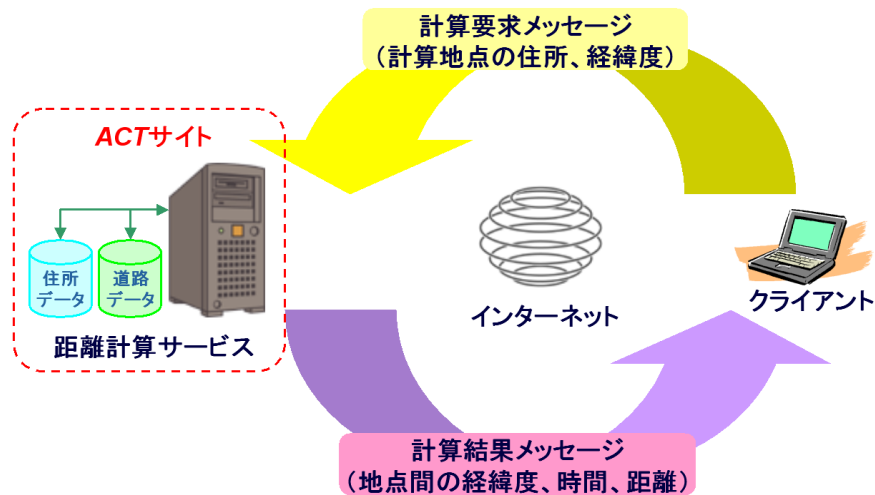
2009/02/28	Version2.0 到達圏／流入圏計算サービス、片道一括計算サービス追加
2010/04/30	ユーザ情報取得サービス追加
2010/07/23	Version2.1 CalcArea2、CalcOneWay2 関連の説明を追加
2010/07/27	郵便番号対応住所列挙機能のコマンド文字列を修正
2010/08/30	ブリッジ DLL を利用したプログラミングに CalcArea2、 CalcOneWay2 関連の説明を追加
2010/11/15	CalcArea サービスのポリゴン補完機能の説明を追加
2011/02/01	住所検索サービスに号レベルの設定を追加
2011/09/01	住所検索サービスの新サービスを追加
2011/11/12	Version2.2 ルート計算サービス、最短ルート計算サービス、直線距離計算サービス、 最寄ノード取得サービス、近傍ノード列挙サービス、最寄リンク 取得サービス、道路速度取得サービス、道路速度設定サービス、計算 用道路データ取得サービスについての説明を追加

## 1. ACT 距離計算サービスの概要

ACT 距離計算サービスは、ACT 距離計算シリーズの基本機能をインターネット環境で利用できる ASP サービスです。ACT 距離計算サービスは、次の機能を提供しています。

項番	サーバ	機能	解説
1	距離計算	ルート計算機能	指定経緯度間のルート計算を行う
2		最短ルート計算機能	指定経緯度間の最短ルート計算（巡回セールスマン問題）を行う
3		直線距離計算機能	指定経緯度間の直線距離計算を行う
4		到達圏／流入圏計算機能	指定経緯度からの到達圏／流入圏計算を行う
5		片道一括計算機能	指定経緯度からの片道一括計算を行う
6		最寄ノード取得機能	指定経緯度から最も近いノードを取得する
7		近傍ノード列挙機能	指定経緯度から最も近いノードを列挙する
8		最寄リンク取得機能	指定経緯度から最も近いリンクを取得する
9		道路速度取得機能	指定リンクの速度を取得する
10		道路速度設定機能	指定リンクの速度を設定する
11	住所検索	計算用道路データ取得機能	計算用道路データの情報を取得する
12		住所検索機能	指定住所文字列の経緯度を検索する
13		住所コード検索機能	指定住所コードの経緯度を検索する
14		下位住所列挙機能	指定住所コードの下位住所を列挙する
15		全住所取得機能	指定住所コードの都道府県からの住所を取得する
16		最寄住所取得機能	指定経緯度が一番近い大字・丁目レベルの住所を取得する
17		郵便番号検索機能	指定郵便番号の経緯度を取得する
18		郵便番号対応住所列挙機能	指定郵便番号に対応する住所を列挙する

**ACT** 距離計算サービスを利用する場合は、クライアントから計算要求メッセージを **ACT** 距離計算サービスのサーバに送信します。サーバでは、受信した計算要求メッセージを解析し、距離計算を行います。計算後、計算結果を計算結果メッセージとしてクライアントへ返信します。



**ACT** 距離計算サービスは、XML/Web サービスとして実装されています。クライアントと **ACT** 距離計算サービスのサーバの通信には SOAP を使用します。SOAP の下位プロトコルは、http (ポート番号 80) のみに対応しています。また、セキュリティとして SSL (ポート番号 443) にも対応しています (一般会員以上)。

本書でよく用いられる地図関連の用語説明を以下に記します。

カテゴリ	用語	説明
地図	ノード	交差点
	リンク	交差点と交差点を結ぶ道路
	ロケーション	計算対象となる交差点
	ジオアイテム	住所、郵便番号文字列から経緯度情報を取得するための構造体
	ジオポイント	住所、郵便番号文字列からノード情報を取得するための構造体
	ジオロケーション	住所、郵便番号文字列から経緯度情報、ノード情報を取得するための構造体
	住所レベル	ACT 距離計算サービスで使用する住所のレベル <ul style="list-style-type: none"> <li>・都道府県レベル</li> <li>・市区町村レベル</li> <li>・大字・丁目レベル</li> <li>・街区レベル</li> <li>・号レベル</li> </ul> 都道府県レベル側を上位、号レベル側を下位とします
	下位住所	住所レベルの 1 つ下のレベル
Web サービス	WSDL	XML 形式で記載された Web サービスを使用するための仕様書。WSDL ファイルを使用することで、Web サービスの開発を容易に行うことができます
	SOAP	ネットワーク上のオブジェクトにアクセスするための通信プロトコル
	リスナ	Web サービスのデータ受信を行うプログラム



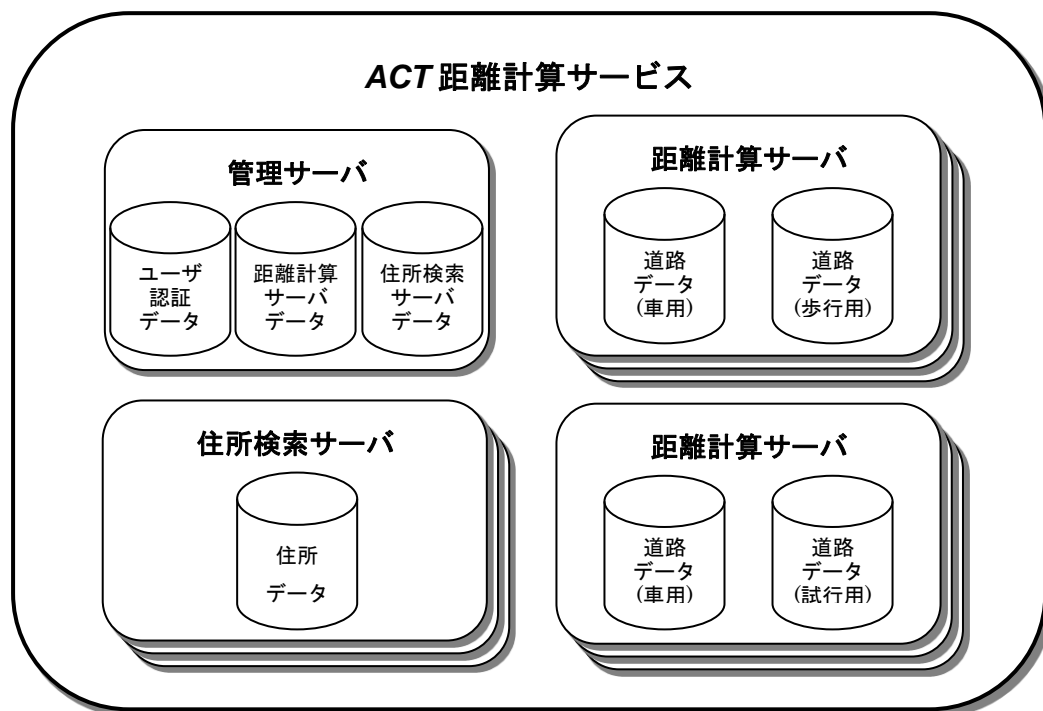
## 2. ACT 距離計算サービスのサーバ構成

ACT 距離計算サービスは、1 台の管理サーバと複数台の距離計算サーバと住所検索サーバから構成されています。

管理サーバは、ユーザ ID やパスワードが正しいかを判定するユーザ認証機能、距離計算サーバがどのような計算用道路データを保持しているかといった状態管理機能を有します。

距離計算サーバは、計算用道路データという道路ネットワークデータを保持しており、クライアントから受信した計算要求をもとに、距離計算を実行し、計算結果をクライアントに返信します。  
1 台の距離計算サーバが複数の計算用道路データを保持している場合があります。

住所検索サーバは、住所データを保持しており、クライアントから受信した検索要求をもとに、漢字住所文字列、住所コード、郵便番号から当該地点の経緯度などを検索し、検索結果をクライアントに返信します。



### 3. サービスの種類

距離計算サービスのサーバには、次のような Web サービスが存在します。距離計算を行うには、少なくとも管理サーバと距離計算サーバの両方のサービス呼び出す必要があります。

サーバ	サービス	解 説
管理サーバ	計算用道路データ情報取得サービス	指定ユーザが計算可能な計算用道路データのリストを取得
	距離計算サーバ取得サービス	指定計算用道路データで計算可能な距離計算サーバのエンドポイント URL を取得
	住所検索サーバ取得サービス	検索可能な住所検索サーバのエンドポイント URL を取得
	ユーザ情報取得サービス	ユーザ情報を取得
距離計算サーバ	ルート計算サービス	ルート計算を実行
	最短ルート計算サービス	最短ルート計算を実行
	直線距離計算サービス	直線距離計算を実行
	到達圏/流入圏計算サービス	到達圏／流入圏計算を実行
	片道一括計算サービス	片道一括計算を実行
	最寄ノード取得サービス	指定経緯度から最も近いノードを取得
	近傍ノード列挙サービス	指定経緯度から最も近いノードを列挙
	最寄リンク取得サービス	指定経緯度から最も近いリンクを取得
	道路速度取得サービス	指定リンクの速度を取得
	道路速度設定サービス	指定リンクの速度を設定
	計算用道路データ取得サービス	現在使用している計算用道路データの情報を取得
	距離計算サービス用メッセージ文字列取得サービス	エラー番号からエラー内容を示す文字列を取得
	距離計算サービス（旧）	各種距離計算を実行（互換性のために残されています）
住所検索サーバ	住所検索サービス	指定漢字住所から経緯度を取得
	住所コード検索サービス	指定住所コードから経緯度を取得
	下位住所数取得サービス	指定住所コードの下位住所数を取得
	下位住所列挙サービス	指定住所コードの下位住所を列挙
	全住所取得サービス	指定住所コードの都道府県からの住所を取得

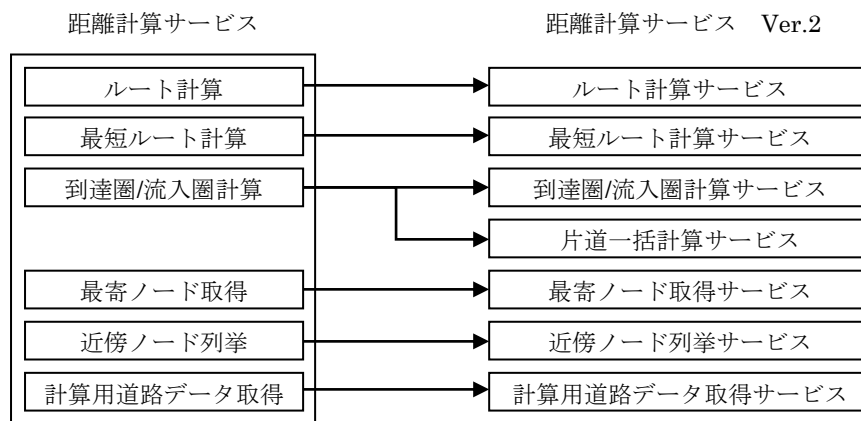
次ページに続く

前ページからの続き

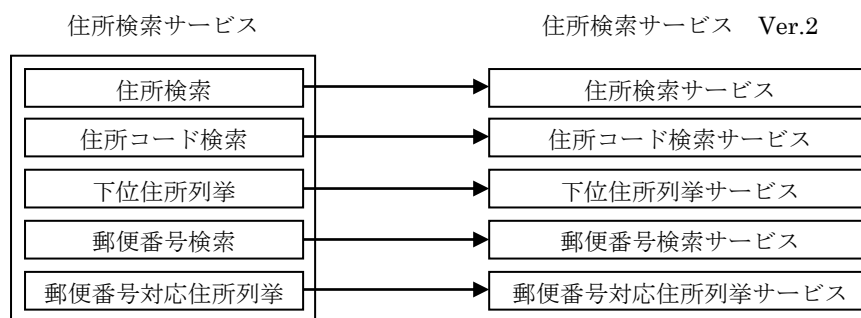
サーバ	サービス	解 説
住所検索サーバ	最寄住所取得サービス	指定経緯度の最寄住所（大字・丁目レベル）を取得
	郵便番号検索サービス	指定郵便番号から経緯度を検索
	郵便番号対応住所数取得サービス	指定郵便番号に対応する住所数を取得
	郵便番号対応住所列挙サービス	指定郵便番号に対応する住所を列挙
	住所検索サービス用メッセージ文字列取得サービス	エラー番号からエラー内容を示す文字列を取得
	住所検索サービス（旧）	各種住所検索を実行（互換性のために残されています）

## 【サービスの機能分割について】

距離計算サービス Ver.2 では、各機能を使いやすくするため、機能ごとにサービスを分割します。以前の距離計算サービスは、旧バージョンで作成されたプログラムとの互換性のために提供しているため、新規作成時は機能分割後のサービスをご利用ください。



住所検索サービスも同様に、機能ごとにサービスを分割します。以前の住所検索サービスは、旧バージョンで作成されたプログラムとの互換性のために提供しているため、新規作成時は機能分割後のサービスをご利用ください。



## 4. 使用する住所データ

**ACT** 距離計算サービスでは住所を下表の 5 段階に区切ってデータ化しています。

住所レベル名	住所レベル定数
都道府県レベル	1
市区町村レベル	2
大字・丁目レベル	8
街区レベル	16
号レベル	32

住所レベルごとに分割された住所データにはそれぞれ住所コードが割り当てられています。  
住所レベルごとの住所コード桁数は下表の通りです。

住所レベル名	住所コード桁数
都道府県レベル	2
市区町村レベル	5
大字・丁目レベル	11
街区レベル	16
号レベル	20

例えば、「東京都港区新橋 3 丁目 7 - 4」という住所は下表のように分割されてデータ化されています。

住所名	住所レベル	住所コード
東京都	1	13
港区	2	13103
新橋 3 丁目	8	13103016003
7	16	13103016003000007
4	32	131030160030000070003

この時「新橋 3 丁目」のひとつ上位の住所コード（市区町村レベル）を得るには、「新橋 3 丁目」の住所コード「13103016003」の先頭から 5 桁を取り出すと住所コード「13103」が得られます。

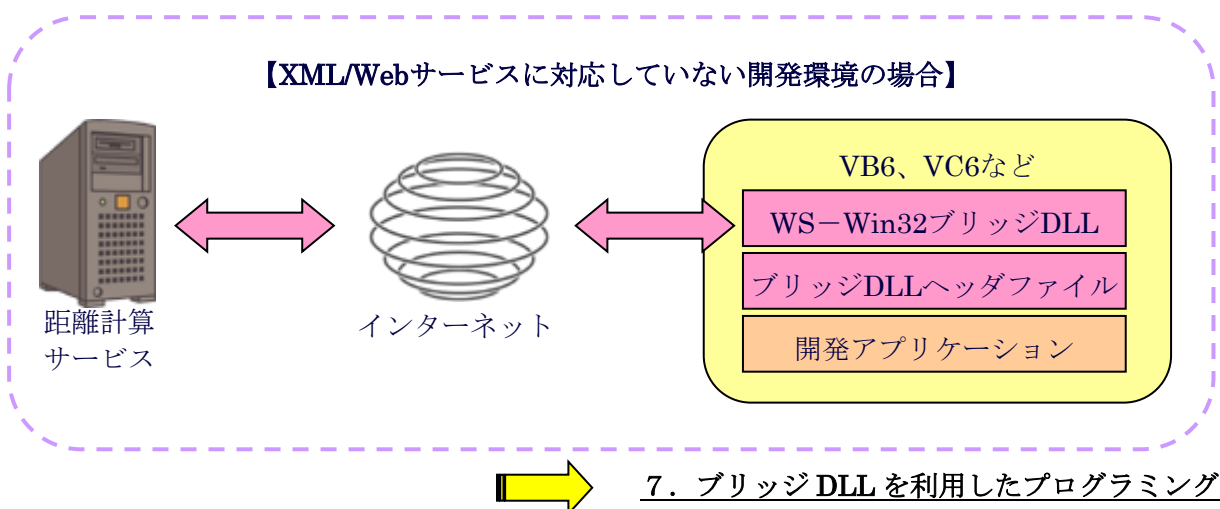
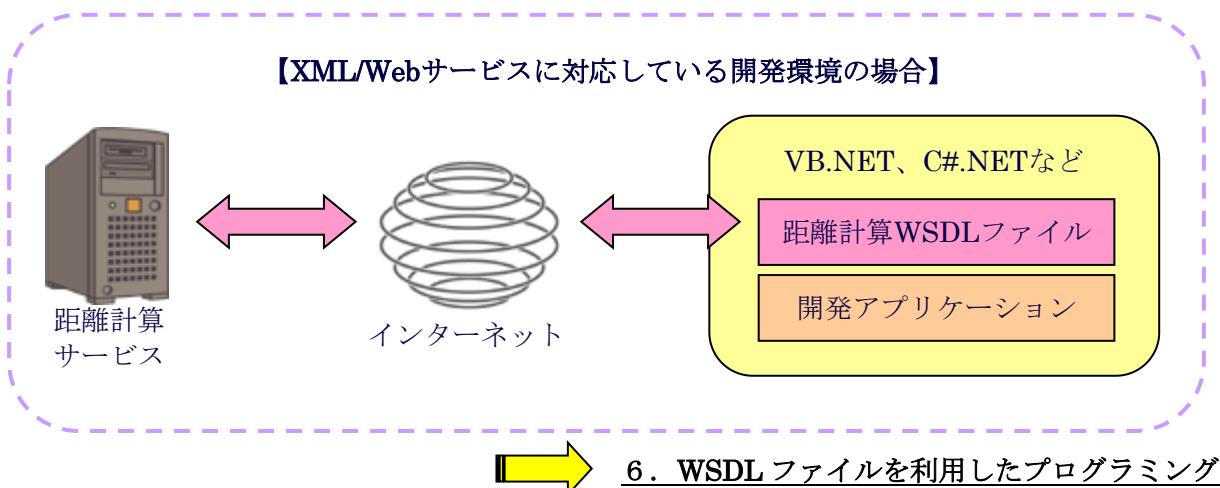
住所コードは本サービスの住所データ更新時に変更される場合がありますのでご注意ください。

## 5. プログラミング方法

**ACT** 距離計算サービスを呼び出すプログラミング方法として、WSDL ファイルを使用する方法と Win32DLL を使用する方法があります。

開発環境が XML/Web サービスに対応している場合、「**ACT** 距離計算サービス WSDL ファイル」を使用することにより、距離計算サービスを直接呼び出すことができます。

開発環境が XML/Web サービスに対応していない場合、直接距離計算サービスを呼び出すことができません。この場合、**ACT** 距離計算サービスと Win32 アプリケーションの橋渡しを行う「**ACT** 距離計算サービス - Win32 ブリッジ DLL」を使用します。この DLL を使用することにより、Win32 アプリケーションから XML/Web サービスを意識することなく **ACT** 距離計算サービスを利用できるようになります。



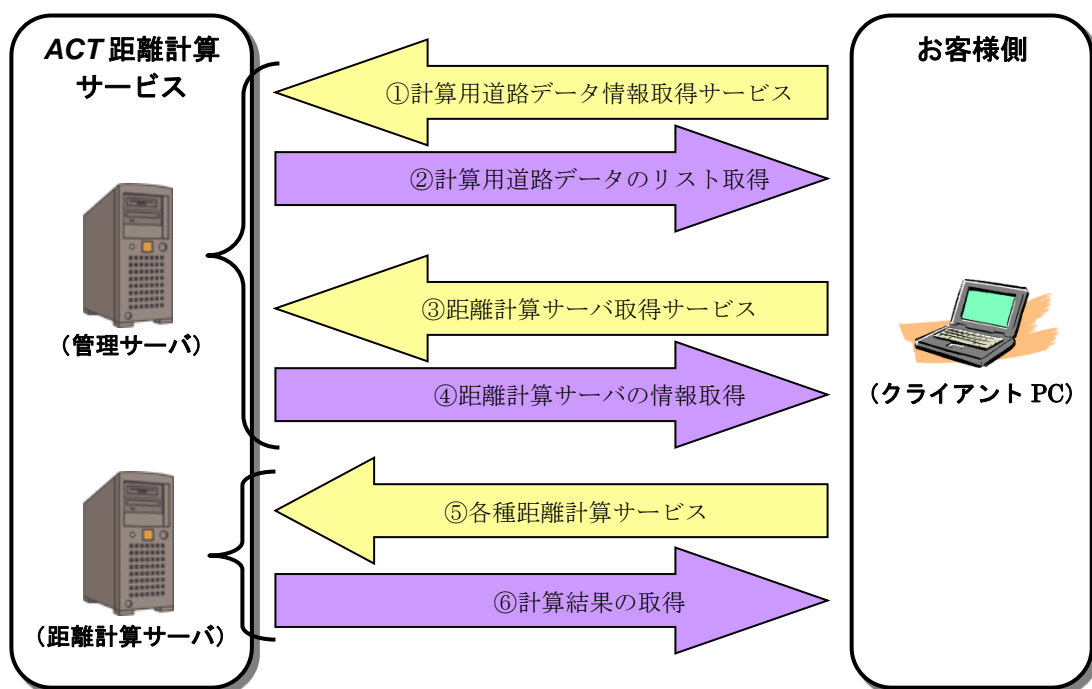
## 6. WSDL ファイルを利用したプログラミング

本章では、**ACT** 距離計算サービスの呼び出しに「**ACT** 距離計算サービス WSDL ファイル」を利用したプログラミングについて解説します。WSDL ファイルを開発環境へ取り込む方法や、SOAP のエンドポイント URL の設定については、各開発環境のマニュアルを参照してください。

### 6. 1 呼び出し手順

#### (1) 距離計算を行う場合

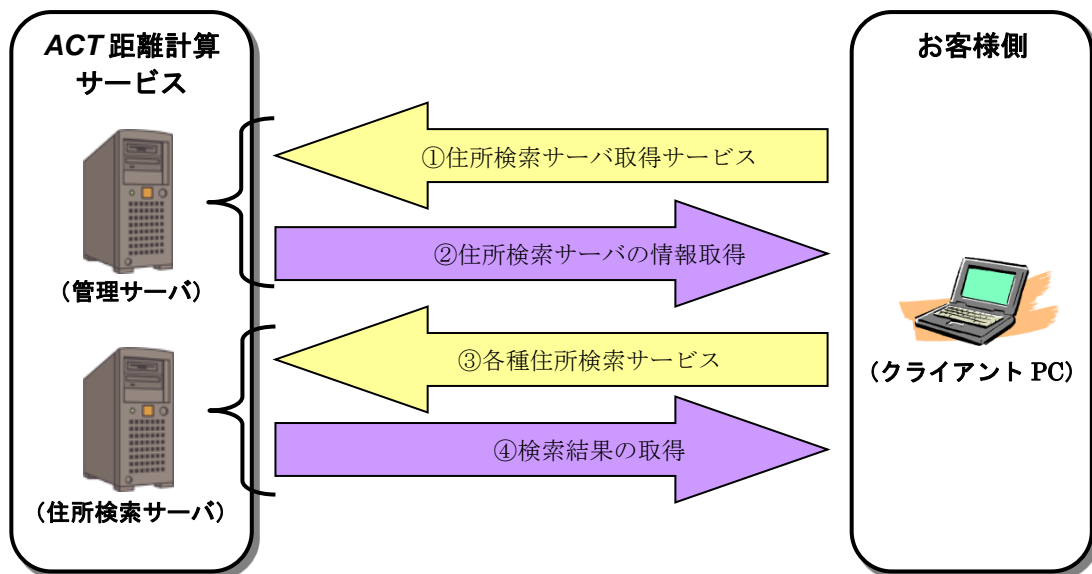
距離計算は、次のような手順で処理を行います。



- ① 計算用道路データ情報取得サービス呼び出し  
ユーザ毎に与えられるユーザ ID を管理サーバへ送信し、計算が可能な計算用道路データのリストを取得するサービス呼び出します。
- ② 計算用道路データのリスト取得  
管理サーバから計算用道路データのリストを取得します。クライアントではこのリストを画面に表示し、ユーザ操作により選択、計算したい計算用道路データを決定します。
- ③ 距離計算サーバ取得サービス呼び出し  
計算したい計算用道路データで距離計算可能な距離計算サーバの情報を取得するサービス呼び出します。
- ④ 距離計算サーバの情報取得  
距離計算サーバのエンドポイント URL、IP アドレス、ポート番号を取得します。
- ⑤ 各種距離計算サービス呼び出し  
④で取得した距離計算サーバのエンドポイント URL、IP アドレス、ポート番号を用いて、各計算サービスを呼び出します。
- ⑥ 計算結果の取得  
計算結果を取得し、所要時間、道のり、走行ルートなどの情報を表示します。

(2) 住所検索を行う場合

住所検索は、次のような手順で処理を行います。



①住所検索サーバ取得サービス呼び出し

住所検索可能な住所検索サーバの情報を取得するサービスを呼び出します。

②住所検索サーバの情報取得

住所検索サーバのエンドポイント URL、IP アドレス、ポート番号を取得します。

③各種住所検索サービス呼び出し

②で取得した住所検索サーバのエンドポイント URL、IP アドレス、ポート番号を用いて、各種住所検索サービスを呼び出します。

④検索結果の取得

検索結果である経緯度を取得し、地図上に表示します。

## 6. 2 WSDL ファイル

**ACT** 距離計算サービスの WSDL ファイルは、各サーバ毎に分かれています。

サーバ	WSDL ファイル	サービス
管理サーバ	DCAdmin.wsdl	<a href="#">計算用道路データ情報取得サービス</a>
		<a href="#">距離計算サーバ取得サービス</a>
		<a href="#">住所検索サーバ取得サービス</a>
		<a href="#">ユーザ情報取得サービス</a>
距離計算サーバ	DCService.wsdl	<a href="#">ルート計算サービス</a>
		<a href="#">最短ルート計算サービス</a>
		<a href="#">直線距離計算サービス</a>
		<a href="#">到達圏／流入圏計算サービス</a>
		<a href="#">片道一括計算サービス</a>
		<a href="#">最寄ノード取得サービス</a>
		<a href="#">近傍ノード列挙サービス</a>
		<a href="#">最寄リンク取得サービス</a>
		<a href="#">道路速度取得サービス</a>
		<a href="#">道路速度設定サービス</a>
		<a href="#">計算用道路データ情報取得サービス</a>
		<a href="#">距離計算サービス用メッセージ文字列取得サービス</a>
		<a href="#">距離計算サービス (旧)</a>
住所検索サーバ	GCService.wsdl	<a href="#">住所検索サービス</a>
		<a href="#">住所コード検索サービス</a>
		<a href="#">下位住所数取得サービス</a>
		<a href="#">下位住所列挙サービス</a>
		<a href="#">全住所取得サービス</a>
		<a href="#">最寄住所取得サービス</a>
		<a href="#">郵便番号検索サービス</a>
		<a href="#">郵便番号対応住所数取得サービス</a>
		<a href="#">郵便番号対応住所列挙サービス</a>
		<a href="#">住所検索サービス用メッセージ文字列取得サービス</a>
		<a href="#">住所検索サービス (旧)</a>



### 6. 3 管理サービス

#### 6. 3. 1 計算用道路データ情報取得サービス

#### GetUsableNWInfos(Environ As Environ, NWInfos As ArrayOfNWInfo) As boolean

環境設定構造体の UserID メンバに設定されているユーザが計算可能な計算用道路データの情報を取得します。

#### エンドポイント URL

「サービス開始通知書」で指定された URL

(2011 年 11 月 12 日現在 <http://distcalc.act-inc.co.jp/dcadmin/admin.asmx>)

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
NWInfos	計算可能な計算用道路データの情報を取得する計算用道路データ構造体配列

#### 戻り値

正常に取得できた場合 True、取得出来なかった場合 False。

#### 備考

本サービスを呼び出す際には、環境設定構造体 Environ のメンバに各データを設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列)
Host	string		(空文字列)
Port	int		(0)

本サービスの呼び出しが成功すると、計算用道路データ構造体配列 NWInfos の各要素に次のようなデータが設定されます。

メンバ	データ型	説 明
Folder	string	計算用道路データの格納フォルダ
Note	string	計算用道路データの名称
IsDetailExist	boolean	詳細ルート出力可能フラグ
IsTollExist	boolean	通行料金出力可能フラグ
NWID	guid	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

## 6. 3. 2 距離計算サーバ取得サービス

**GetUsableDCServer(Environ As Environ, NWInfo As NWInfo) As boolean**

計算用道路データ情報構造体に設定されているネットワーク ID の計算用道路データを使用して、距離計算が可能な距離計算サーバの情報を環境設定構造体に設定します。

**エンドポイント URL**

「サービス開始通知書」で指定された URL

(2011 年 11 月 12 日現在 <http://distcalc.act-inc.co.jp/dcadmin/admin.asmx>)

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
NWInfo	計算したい計算用道路データのネットワーク ID が設定された計算用道路データ構造体

**戻り値**

正常に取得できた場合 True、取得出来なかった場合 False。

**備考**

本サービスを呼び出す際には、環境設定構造体 Environ の各メンバにデータを設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列)
Host	string		(空文字列)
Port	int		(0)

また、計算用道路データ情報構造体 NWInfo の各メンバにデータを設定してください。

メンバ	データ型	設定	説 明
Folder	string		(空文字列) (注)
Note	string		(空文字列) (注)
IsDetailExist	boolean		(true) (注)
IsTollExist	boolean		(true) (注)
NWID	guid	必要	計算用道路データを識別するためのネットワーク ID
NWID_Without Speed	guid		(空文字列) (注)

(注) 計算用道路データ情報取得サービスで取得したデータが設定されていても、動作に支障はありません。

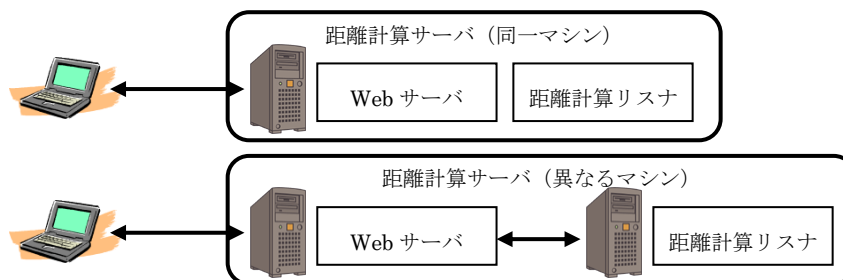
本サービスの呼び出しが成功すると、環境設定構造体 Environ の各要素に次のようなデータが設定されます。環境設定構造体 Environ は、入力データであるユーザ ID やパスワードの設定、および、出力データである距離計算サーバ情報の取得、いずれにも使用します。

メンバ	データ型	説 明
UserID	string	(ユーザ ID)
UUID	string	(パスワード)
HWUID	string	(空文字列)
WSURL	string	距離計算サーバのエンドポイント URL
Host	string	距離計算リスナ(注)の IP アドレス
Port	int	距離計算リスナ(注)のポート番号

(注) 下記【距離計算サーバについて】を参照。

#### 【距離計算サーバについて】

Web サービスは、エンドポイント URL に対して呼び出しを行いますので、通常 IP アドレスやポート番号は必要ありません。距離計算サーバでは、SOAP メッセージに応答する Web サーバと、実際に距離計算を行う距離計算リスナという 2つのプログラムが動作しています。クライアントからの計算要求が多くなると距離計算リスナの負荷が高くなり、Web サーバの応答が遅くなる可能性があります。Web サーバと距離計算リスナを別マシンで稼働させることで、この問題に対応します。このため、距離計算サーバ取得サービスでは、Web サーバのエンドポイント URL や距離計算リスナの IP アドレスやポート番号を取得する必要があります。



### 6. 3. 3 住所検索サーバ取得サービス

#### GetUsableGeoServer(Environ As Environ) As boolean

住所検索が可能な住所検索サーバの情報を環境設定構造体に設定します。

#### エンドポイント URL

「サービス開始通知書」で指定された URL

(2011 年 11 月 12 日現在 <http://distcalc.act-inc.co.jp/dcadmin/admin.asmx>)

#### パラメータ

#### 解説

---

Environ	ユーザ ID およびパスワードが設定された環境設定構造体
---------	------------------------------

#### 戻り値

正常に取得できた場合 True、取得出来なかった場合 False。

#### 備考

本サービスを呼び出す際には、環境設定構造体 Environ の各メンバにデータを設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列)
Host	string		(空文字列)
Port	int		(0)

本サービスの呼び出しが成功すると、環境設定構造体 **Environ** の各要素に次のようなデータが設定されます。環境設定構造体 **Environ** は、入力データであるユーザ ID やパスワードの設定、および、出力データである距離計算サーバ情報の取得、いずれにも使用します。

メンバ	データ型	説 明
UserID	string	(ユーザ ID)
UUID	string	(パスワード)
HWUID	string	(空文字列)
WSURL	string	住所検索サーバのエンドポイント URL
Host	string	住所検索リスナの IP アドレス
Port	int	住所検索リスナのポート番号

## 6. 3. 4 ユーザ情報取得サービス

**GetUserInfo2 (Environ As Environ, UserInformation As UserInformation2) As boolean**

ユーザ情報をユーザ情報構造体に設定します。

**エンドポイント URL**

「サービス開始通知書」で指定された URL

(2011 年 11 月 12 日現在 <http://distcalc.act-inc.co.jp/dcadmin/admin.asmx>)

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
UserInformation	ユーザ情報構造体

**戻り値**

正常に取得できた場合 True、取得出来なかった場合 False。

**備考**

本サービスを呼び出す際には、環境設定構造体 Environ の各メンバにデータを設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列)
Host	string		(空文字列)
Port	int		(0)

本サービスの呼び出しが成功すると、ユーザ情報構造体 `UserInfomation` の各要素に次のようなデータが設定されます。

メンバ	データ型	説 明
<code>MemberKind</code>	string	会員種別
<code>ExpiryDate</code>	string	会員期限 (YYYY/MM/DD)
<code>CalcInterval</code>	int	距離計算間隔(単位：秒)
<code>CalcRouteMaxLocs</code>	int	ルート計算可能最大地点数
<code>CalcOptRouteMaxLocs</code>	int	最短ルート計算可能最大地点数
<code>CalcAreaMaxTime</code>	int	到達圏／流入圏計算可能最長時間 (-1 は無制限)
<code>CalcAreaMaxDist</code>	int	到達圏／流入圏計算可能最長距離 (-1 は無制限)
<code>CalcAreaMaxLocs</code>	int	到達圏／流入圏計算可能最大地点数
<code>GeoMaxAddressLevel</code>	int	住所検索レベル (1:都道府県、2:市区町村、4:大字丁目、16:街区、32:号)
<code>GeoInterval</code>	int	住所検索計算間隔 (単位：秒、0 は無制限)
<code>GeoEnumInterval</code>	int	下位住所列举計算間隔 (単位：秒、0 は無制限)
<code>GeoMaxLocs</code>	int	住所検索可能最大地点数
<code>CalcCount</code>	int	計算回数
<code>MaxCalcCount</code>	int	最大計算回数 (-1 は無制限)
<code>GeoCount</code>	int	住所検索回数
<code>MaxGeoCount</code>	int	最大住所検索回数 (-1 は無制限)
<code>Expand_I32_1</code>	int	(0)
<code>Expand_I32_2</code>	int	(0)
<code>Expand_I32_3</code>	int	(0)
<code>Expand_I32_4</code>	int	(0)
<code>Expand_Str_1</code>	string	(空文字列)
<code>Expand_Str_2</code>	string	(空文字列)
<code>Expand_Str_3</code>	string	(空文字列)
<code>Expand_Str_4</code>	string	(空文字列)



## 6. 4 距離計算サービス

### 6. 4. 1 ルート計算サービス

**CalcRoute(Environ As Environ, CRParam As CRParam, Locs As ArrayOfLoc, Pnts As ArrayOfPnt, Rns As ArrayOfRN, Trds As ArrayOfTrd, NWInfo As NWInfo) As int**

ロケーション構造体配列に設定されている要素順にルート計算を行います

#### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
CRParam	IN	ルート計算用パラメータ構造体
Locs	IN / OUT	ロケーション構造体配列
Pnts	OUT	ポイント構造体配列
Rns	OUT	経由道路名構造体配列
Trds	OUT	経由有料道路構造体配列
NWInfo	OUT	計算用道路データ情報構造体

#### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	CRParam	CRParam
(3)	Locs	ArrayOfLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) ルート計算用パラメータ構造体 CRParam

ルート計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Transport	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用しない</p> <p>1 : 高速道路を使用する</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※ 1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空機を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
PntsOutput	int	R	<p>ポイント構造体配列出力フラグ</p> <p>走行ルート of の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
RnsOutput	int	R	<p>経由道路名構造体配列出力フラグ</p> <p>経由道路名の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する。</p>
TrdsOutput	int	R	<p>経由有料道路構造体配列出力フラグ</p> <p>経由有料道路の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
HighwayOutput	int	R	<p>高速道路区間出力フラグ</p> <p>ジオロケーション構造体の高速道路区間の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
TollOutput	int	R	<p>通行料金出力フラグ</p> <p>ロケーション構造体の通行料金の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
RouteKind	int	R	<p>ルート種別指定</p> <p>ポイント構造体 to 出力する走行ルートの種別を指定します。</p> <p>0 : 簡易ルート、1 : 詳細ルート</p>

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

## (3) ロケーション構造体配列 Locs

ルート計算を行うロケーションを設定します。

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R/W	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。 計算後は最寄ノードコードが出力されます。
Lon	string	R/W	経度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの経度が出力されます。
Lat	string	R/W	緯度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの緯度が出力されます。
Time	int	W	所要時間（分単位）
DSec	int	W	所要時間（0.1 秒単位）
Dist	int	W	道のり（m 単位）
HighwayTime	int	W	高速道路区間の所要時間（分単位）
HighwayDSec	int	W	高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int	W	高速道路区間の道のり（m 単位）
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) ポイント構造体配列 Pnts

計算後にルート情報の経緯度が出力されます。

ポイント構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (5) 経由道路名構造体配列 Rns

計算後に走行ルートの道路名／交差点名が出力されます。

経由道路名構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
LocFlag	int	W	ロケーションフラグ 1 : ジオロケーション構造体で指定された地点 0 : 上記以外
LinkFlag	int	W	リンクフラグ 0 : リンク以外、1 : リンク
NLCode	int	W	ノードリンク番号 リンクフラグが 0 の場合、ノードコード リンクフラグが 1 の場合、リンク番号
Name	string	W	名称 ロケーションフラグが 1 の場合、ロケーションの Tag 情報。 リンクフラグが 0 の場合、交差点名称。 リンクフラグが 1 かつインターチェンジタイプが 0 の場合、道路名称。 リンクフラグが 1 かつインターチェンジタイプが 0 でない場合、インターチェンジ名称
Time	int	W	所要時間（分単位）
DSec	int	W	所要時間（1/10 秒単位）
Dist	int	W	道のり（m 単位）

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
ICType	int	W	インターチェンジタイプ 0 : インターチェンジではない 0 以外 : インターチェンジ
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

#### (6) 経由有料道路構造体配列 Trds

計算後に走行ルートの有料道路の情報が設定されます。

経由有料道路構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
InRdName	string	W	入口または課金の有料道路名称 入口または課金の料金所名称、出口の有料道路名称、出口の料金所名称が空白の場合、ロケーションのタグ情報が格納されます。
InICName	string	W	入口または課金の料金所名称
OutRdName	string	W	出口の有料道路名称
OutICName	string	W	出口の料金所名称
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Dist	int	W	距離（km 単位） 有料道路のキロポスト。 0 または有効な値でない場合があります。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (7) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 4. 2 最短ルート計算サービス

**CalcOptRoute(Environ As Environ, CRParam As CRParam, Locs As ArrayOfLoc, Pnts As ArrayOfPnt, Rns As ArrayOfRN, Trds As ArrayOfTrd, NWInfo As NWInfo) As int**

ロケーション構造体配列の最初の要素を出発地、最後の要素を到着地として、その間の要素の最適巡回ルート（巡回セールスマン問題）を計算します。要素数が約 20 を超えると最適解（最適巡回ルート）を得られない場合があります。

### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
CRParam	IN	ルート計算用パラメータ構造体
Locs	IN / OUT	ロケーション構造体配列
Pnts	OUT	ポイント構造体配列
Rns	OUT	経由道路名構造体配列
Trds	OUT	経由有料道路構造体配列
NWInfo	OUT	計算用道路データ情報構造体

### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	CRParam	CRParam
(3)	Locs	ArrayOfLoc



## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) ルート計算用パラメータ構造体 CRParam

ルート計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Transport	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用しない</p> <p>1 : 高速道路を使用する</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※ 1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空機を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
PntsOutput	int	R	<p>ポイント構造体配列出力フラグ</p> <p>走行ルート of の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
RnsOutput	int	R	<p>経由道路名構造体配列出力フラグ</p> <p>経由道路名の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する。</p>
TrdsOutput	int	R	<p>経由有料道路構造体配列出力フラグ</p> <p>経由有料道路の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
HighwayOutput	int	R	<p>高速道路区間出力フラグ</p> <p>ジオロケーション構造体の高速道路区間の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
TollOutput	int	R	<p>通行料金出力フラグ</p> <p>ロケーション構造体の通行料金の出力を指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
RouteKind	int	R	<p>ルート種別指定</p> <p>ポイント構造体 to 出力する走行ルートの種別を指定します。</p> <p>0 : 簡易ルート、1 : 詳細ルート</p>

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

## (3) ロケーション構造体配列 Locs

ルート計算を行うロケーションを設定します。

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R/W	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。 計算後は最寄ノードコードが出力されます。
Lon	string	R/W	経度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの経度が出力されます。
Lat	string	R/W	緯度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの緯度が出力されます。
Time	int	W	所要時間（分単位）
DSec	int	W	所要時間（0.1 秒単位）
Dist	int	W	道のり（m 単位）
HighwayTime	int	W	高速道路区間の所要時間（分単位）
HighwayDSec	int	W	高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int	W	高速道路区間の道のり（m 単位）
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) ポイント構造体配列 Pnts

計算後にルート情報の経緯度が出力されます。

ポイント構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (5) 経由道路名構造体配列 Rns

計算後に走行ルートの道路名／交差点名が出力されます。

経由道路名構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
LocFlag	int	W	ロケーションフラグ 1：ジオロケーション構造体で指定された地点 0：上記以外
LinkFlag	int	W	リンクフラグ 0：リンク以外、1：リンク
NLCode	int	W	ノードリンク番号 リンクフラグが 0 の場合、ノードコード リンクフラグが 1 の場合、リンク番号
Name	string	W	名称 ロケーションフラグが 1 の場合、ロケーションの Tag 情報。 リンクフラグが 0 の場合、交差点名称。 リンクフラグが 1 かつインターチェンジタイプが 0 の場合、道路名称。 リンクフラグが 1 かつインターチェンジタイプが 0 でない場合、インターチェンジ名称
Time	int	W	所要時間（分単位）
DSec	int	W	所要時間（1/10 秒単位）
Dist	int	W	道のり（m 単位）

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
ICType	int	W	インターチェンジタイプ 0 : インターチェンジではない 0 以外 : インターチェンジ
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

## (6) 経由有料道路構造体配列 Trds

計算後に走行ルートの有料道路の情報が設定されます。

経由有料道路構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
InRdName	string	W	入口または課金の有料道路名称 入口または課金の料金所名称、出口の有料道路名称、出口の料金所名称が空白の場合、ロケーションのタグ情報が格納されます。
InICName	string	W	入口または課金の料金所名称
OutRdName	string	W	出口の有料道路名称
OutICName	string	W	出口の料金所名称
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Dist	int	W	距離（km 単位） 有料道路のキロポスト。 0 または有効な値でない場合があります。

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

## (7) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

### 6. 4. 3 直線距離計算サービス

**CalcStraightLine(Environ As Environ, CSLParam As CSLParam, Locs As ArrayOfLoc, Pnts As ArrayOfPnt, NWInfo As NWInfo) As int**

ロケーション構造体配列に設定されている要素順に直線距離を計算します。

この計算で求める直線距離は、地球を回転楕円体と仮定した場合の最短測地線長です。

実際の長さとは異なる可能性がありますので注意してください。

また 1 地点からその対蹠点（回転楕円体の裏側）付近への距離計算はできません。

#### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
CSLParam	IN	直線距離計算用パラメータ構造体
Locs	IN / OUT	ロケーション構造体配列
Pnts	OUT	ポイント構造体配列
NWInfo	OUT	計算用道路データ情報構造体

#### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	CSLParam	CSLParam
(3)	Locs	ArrayOfLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 直線距離計算用パラメータ構造体 CSLParam

直線距離計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## (3) ロケーション構造体配列 Locs

直線距離計算を行うロケーションを設定します。

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string		ノードコード
Lon	string	R	経度（度単位小数点以下 6 桁）
Lat	string	R	緯度（度単位小数点以下 6 桁）
Time	int		所要時間（分単位）
DSec	int		所要時間（0.1 秒単位）
Dist	int	W	累積直線距離（m 単位）
HighwayTime	int		高速道路区間の所要時間（分単位）
HighwayDSec	int		高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int		高速道路区間の道のり（m 単位）
Toll_SS	int		二輪・軽の通行料金（円単位）
Toll_S	int		普通車の通行料金（円単位）
Toll_M	int		中型車の通行料金（円単位）
Toll_L	int		大型車の通行料金（円単位）
Toll_LL	int		特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) ポイント構造体配列 Pnts

計算後にロケーションを結ぶ直線の経緯度が出力されます。

ポイント構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (5) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

#### 6. 4. 4 到達圏／流入圏計算サービス

**CalcArea3(Environ As Environ, CA3Param As CA3Param, Pnts As ArrayOfPnt, NWInfo As NWInfo) As int**

CA3Param 到達圏／流入圏計算用パラメータ構造体に設定されているスタート地点からの到達圏／流入圏計算を行います。Pnts ポイント構造体配列には、到達／流入範囲ポリゴンの各頂点の座標が格納されます。

#### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
CA3Param	IN / OUT	到達圏／流入圏計算用パラメータ構造体
Pnts	OUT	ポイント構造体配列
NWInfo	OUT	計算用道路データ情報構造体

#### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	CA3Param	CA3Param

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 到達圏／流入圏計算用パラメータ構造体 CA3Param

到達圏／流入圏計算を行う際に必要なパラメータを設定します。

計算後に使用されたポリゴン詳細設定が出力されます。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短
PolygonKind	int	R	ポリゴン種別 1 : 単一ポリゴン、2 : 複数ポリゴン
StartPnt	NodePnt	R	到達圏／流入圏計算の中心地点

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Transport	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用しない</p> <p>1 : 高速道路を使用する</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空機を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
AreaKind	int	R	<p>到達圏／流入圏計算種別</p> <p>計算範囲の種別を指定します。</p> <p>1 : 時間圏、2 : 距離圏</p>
AreaRange	int	R	<p>到達圏／流入圏範囲</p> <p>到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。</p>
PolygonLevel	int	R	<p>到達圏／流入圏ポリゴンレベル</p> <p>到達ポリゴンのポリゴンレベルを指定します。</p> <p>-1 : 凸型(簡易)ポリゴン</p> <p>0～20 : 凹凸型(通常)ポリゴン (大きいほど細かくなります)</p>
DonutPolygonLevel	int	R	<p>到達圏／流入圏ドーナツポリゴンレベル</p> <p>到達ポリゴン内で到達できない範囲のポリゴンレベルを指定します。</p> <p>-2 : 表示しない</p> <p>-1 : 凸型(簡易)ポリゴン</p> <p>0～20 : 凹凸型(通常)ポリゴン (大きいほど細かくなります)</p>

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
AreaDirection	int	R	範囲方向 到達圏または流入圏を指定します。 0：到達圏、4096：流入圏
PolygonDetail	int	R/W	ポリゴン詳細設定 下記値の論理和 計算時に使用する機能を指定してください。計算後は使用された機能が設定されています。 0：詳細機能を使用しない 1：中間リンク補間 2：末端リンク補間

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

\* R：リード、W：ライト、R/W：リード&amp;ライトを表しています。

到達圏／流入圏計算用パラメータ構造体のメンバの StartPnt の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。
Lon	string	R	経度（度単位小数点以下6桁） Node を指定した場合、使用されません。
Lat	string	R	緯度（度単位小数点以下6桁） Node を指定した場合、使用されません。

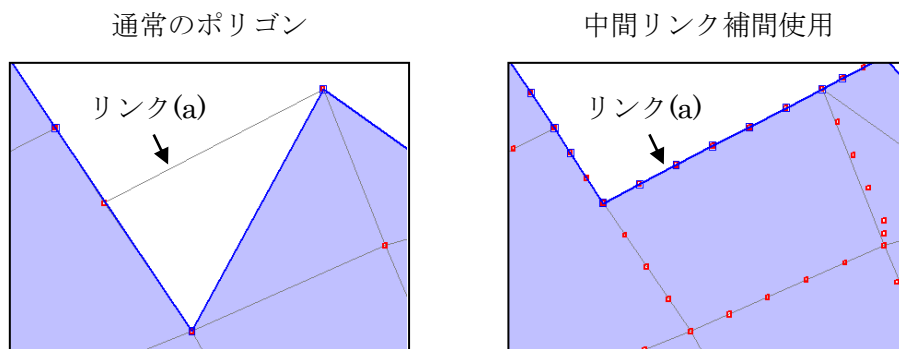
\* R：リード、W：ライト、R/W：リード&amp;ライトを表しています。

### ※ ポリゴン補間機能について

通常、ポリゴンは到達圏内のノードだけを結んで作成しますが、ポリゴン補間機能を使用すると補助となるポイントを追加してポリゴンを作成することができます。

#### ・ 中間リンク補間

到達圏内ノード間のリンク上にポイントを追加してポリゴンを作成します。

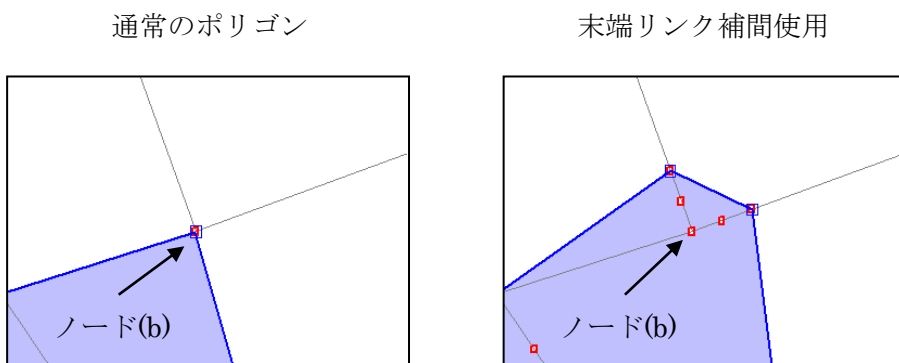


上左図は通常のポリゴン作成機能を使用しています。到達圏内ノードのみを考慮するのでリンク(a)を通行することが可能でも到達圏外になっています。

上右図は中間リンク補間機能を使用しています。到達圏内ノード間のリンク上にポイントを追加しているのでリンク(a)に沿ってポリゴンが作成されています。

#### ・ 末端リンク補間

到達圏の末端リンク上にポイントを追加してポリゴンを作成します。



例えば10分到達圏の計算をしてノード(b)に9分で到達できますが、その先のノードには1分で到達できないとします。

上左図は通常のポリゴン作成機能を使用しています。到達圏内ノードのみを考慮してポリゴンを作成しているので、端数の1分はポリゴンに反映されません。

上右図は末端リンク補間機能を使用しています。ノード(b)の先のリンク上に端数の1分で到達可能なポイントを追加してポリゴンを作成しています。

- ・ 制限

中間リンク補間機能と末端リンク補間機能が有効になるのは、単一ポリゴンかつ到達圏内ノード数が 10 万未満の場合です。ポリゴン補間機能が使用されたかどうか確認したい場合は、計算後に **CA3Param** の **PolygonDetail** を確認してください。ポリゴン補間機能が使用されなかった場合は設定したフラグが消去されています。



## (3) ポイント構造体配列 Pnts

計算後に到達／流入範囲ポリゴンの各頂点の座標が設定されます。

ポイント構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



ポリゴンデータの格納順は左図のようになっています。

各ポリゴンの区切りとして、Lon と Lat に数値として 0 が格納されているポイント構造体を使用されます。Lon と Lat は数値に変換して 0 かどうか判定してください。

まず到達範囲ポリゴンが格納されています。

次に区切りが一つある場合は、直前の到達範囲ポリゴン内のドーナツポリゴンが格納されています。

区切りが二つ続く場合は、次の到達範囲ポリゴンが格納されています。

区切りが三つ続く場合はポリゴンデータの終了を意味します。

## (4) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 4. 5 片道一括計算サービス

**CalcOneWay3(Environ As Environ, COW3Param As COW3Param, Locs As ArrayOfLoc, NWInfo As NWInfo) As int**

中心地点から複数の地点までの距離計算、または複数の地点から中心地点までの距離計算を行います。

## エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
COW3Param	IN	片道一括計算用パラメータ構造体
Locs	IN / OUT	ロケーション構造体配列
NWInfo	OUT	計算用道路データ情報構造体

## 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

## 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	COW3Param	COW3Param
(3)	Locs	ArrayOfLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 片道一括計算用パラメータ構造体 COW3Param

片道一括計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短
StartPnt	NodePnt	R	片道一括計算の中心地点情報
AreaKind	int	R	到達圏／流入圏計算種別 計算範囲の種別を指定します。 1 : 時間圏、2 : 距離圏

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Transport	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用しない</p> <p>1 : 高速道路を使用する</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※ 1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空機を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
AreaRange	int	R	<p>到達圏／流入圏範囲</p> <p>到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。</p>
AreaDirection	int	R	<p>範囲方向</p> <p>到達圏または流入圏を指定します。</p> <p>0 : 到達圏、4096 : 流入圏</p>
HighwayOutput	int	R	<p>高速道路区間出力フラグ</p> <p>ジオロケーション構造体の高速道路区間の出力を行うか指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
TollOutput	int	R	<p>通行料金出力フラグ</p> <p>ジオロケーション構造体の通行料金の出力を行うか指定します。</p> <p>0 : 出力しない、1 : 出力する</p>

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

片道一括計算用パラメータ構造体のメンバの **StartPnt** の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。
Lon	string	R	経度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。
Lat	string	R	緯度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) ロケーション構造体配列 Locs

片道一括計算を行うロケーションを設定します。

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R/W	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。 計算後は最寄ノードコードが出力されます。
Lon	string	R/W	経度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの経度が出力されます。
Lat	string	R/W	緯度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの緯度が出力されます。
Time	int	W	所要時間（分単位）
DSec	int	W	所要時間（0.1 秒単位）
Dist	int	W	道のり（m 単位）
HighwayTime	int	W	高速道路区間の所要時間（分単位）
HighwayDSec	int	W	高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int	W	高速道路区間の道のり（m 単位）
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## 6. 4. 6 最寄ノード取得サービス

**GetNearestNode(Environ As Environ, GNNParam As GNNParam, Locs As ArrayOfLoc, NWInfo As NWInfo) As int**

ロケーション構造体配列に設定されている経緯度から最も近傍のノードを探索し、そのノードコードをロケーション構造体に設定します。

## エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
GNNParam	IN	最寄ノード取得用パラメータ構造体
Locs	IN / OUT	ロケーション構造体配列
NWInfo	OUT	計算用道路データ情報構造体

## 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

## 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	GNNParam	GNNParam
(3)	Locs	ArrayOfLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 最寄ノード取得用パラメータ構造体 GNNParam

最寄ノード取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0：日本測地系、1：世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
Transport	int	R	使用交通機関フラグ 計算用道路データによって使用できる 交通機関は異なります。 下記値の論理和です。 0：高速道路を使用しない 1：高速道路を使用する 2：北海道～青森間、沖縄～鹿児島間の みフェリーを使用する※ 1 4：フェリーを使用する 8：鉄道を使用する 16：航空機を使用する 64：内航船を使用する、または旅客鉄 道を使用する場合特急を使用する

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

※ 1 本州—北海道間、本州—沖縄間の道路距離を簡易的に計算するための設定

## (3) ロケーション構造体配列 Locs

最寄ノードを取得するロケーションを設定します。

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R/W	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。 計算後は最寄ノードコードが出力されます。
Lon	string	R/W	経度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの経度が出力されます。
Lat	string	R/W	緯度（度単位小数点以下6桁） Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの緯度が出力されます。
Time	int		所要時間（分単位）
DSec	int		所要時間（0.1 秒単位）
Dist	int		道のり（m 単位）
HighwayTime	int		高速道路区間の所要時間（分単位）
HighwayDSec	int		高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int		高速道路区間の道のり（m 単位）
Toll_SS	int		二輪・軽の通行料金（円単位）
Toll_S	int		普通車の通行料金（円単位）
Toll_M	int		中型車の通行料金（円単位）
Toll_L	int		大型車の通行料金（円単位）
Toll_LL	int		特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 4. 7 近傍ノード列挙サービス

**EnumNearNode(Environ As Environ, ENNParam As ENNParam, Locs As ArrayOfLoc, NWInfo As NWInfo) As int**

パラメータ構造体に設定されているスタート地点から近傍のノードを最大 100 地点列挙し、ロケーション構造体配列に設定します。

**エンドポイント URL**

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
ENNParam	IN	近傍ノード列挙用パラメータ構造体
Locs	OUT	ロケーション構造体配列
NWInfo	OUT	計算用道路データ情報構造体

**戻り値**

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	ENNParam	ENNParam
(3)	Locs	ArrayOfLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 近傍ノード列挙用パラメータ構造体 ENNParam

近傍ノード列挙を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0：日本測地系、1：世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
Transport	int	R	使用交通機関フラグ 計算用道路データによって使用できる 交通機関は異なります。 下記値の論理和です。 0：高速道路を使用しない 1：高速道路を使用する 2：北海道～青森間、沖縄～鹿児島間の みフェリーを使用する※ 1 4：フェリーを使用する 8：鉄道を使用する 16：航空機を使用する 64：内航船を使用する、または旅客鉄 道を使用する場合特急を使用する
StartPnt	NodePnt	R	近傍ノード列挙の中心地点情報

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

近傍ノード列挙用パラメータ構造体のメンバの StartPnt の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	R	ノードコード 地点をノードコードで指定する場合に使用し ます。通常は"0"を設定してください。
Lon	string	R	経度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。
Lat	string	R	緯度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。



## (3) ロケーション構造体配列 Locs

計算終了後結果が格納されます。

ロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Node	string	W	ノードコード ノードコードが出力されます。
Lon	string	W	経度（度単位小数点以下 6 桁） ノードの経度が出力されます。
Lat	string	W	緯度（度単位小数点以下 6 桁） ノードの緯度が出力されます。
Time	int		所要時間（分単位）
DSec	int		所要時間（0.1 秒単位）
Dist	int		道のり（m 単位）
HighwayTime	int		高速道路区間の所要時間（分単位）
HighwayDSec	int		高速道路区間の所要時間（0.1 秒単位）
HighwayDist	int		高速道路区間の道のり（m 単位）
Toll_SS	int		二輪・軽の通行料金（円単位）
Toll_S	int		普通車の通行料金（円単位）
Toll_M	int		中型車の通行料金（円単位）
Toll_L	int		大型車の通行料金（円単位）
Toll_LL	int		特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称 などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (4) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

#### 6. 4. 8 最寄リンク取得サービス

**GetNearestLink2(Environ As Environ, GNLPParam As GNLPParam, LinkCode As string, NWInfo As NWInfo) As int**

指定経緯度から一番近いリンクを取得します。

##### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
GNLPParam	IN	最寄リンク取得用パラメータ構造体
LinkCode	OUT	結果リンクコード
NWInfo	OUT	計算用道路データ情報構造体

##### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

##### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	GNLPParam	GNLPParam

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 最寄リンク取得用パラメータ構造体 GNLParam

最寄リンク取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0：日本測地系、1：世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
Transport	int	R	使用交通機関フラグ 計算用道路データによって使用できる 交通機関は異なります。 下記値の論理和です。 0：高速道路を使用しない 1：高速道路を使用する 2：北海道～青森間、沖縄～鹿児島間の みフェリーを使用する※ 1 4：フェリーを使用する 8：鉄道を使用する 16：航空機を使用する 64：内航船を使用する、または旅客鉄 道を使用する場合特急を使用する
Lon	String	R	経度（度単位小数点以下 6 桁）
Lat	String	R	緯度（度単位小数点以下 6 桁）
Range	int	R	検索範囲（m 単位）

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

(3) 結果リンクコード **LinkCode**

計算終了後、リンクコードが格納されます。

見つからなかった場合は 0 が格納されます。

(4) 計算用道路データ情報構造体 **NWinfo**

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
<b>Folder</b>	<b>string</b>		未使用
<b>Note</b>	<b>string</b>	<b>W</b>	計算用道路データの説明
<b>IsDetailExist</b>	<b>boolean</b>	<b>W</b>	詳細ルート出力可能フラグ
<b>IsTollExist</b>	<b>boolean</b>	<b>W</b>	通行料金出力可能フラグ
<b>NWID</b>	<b>guid</b>	<b>W</b>	計算用道路データを識別するためのネットワーク ID
<b>NWID_WithoutSpeed</b>	<b>guid</b>	<b>W</b>	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 4. 9 道路速度取得サービス

**GetSpeed2(Environ As Environ, GSParam As GSParam, SpeedAB As int, SpeedBA As int, NodeA As NodePnt, NodeB As NodePnt, NWInfo As NWInfo) As int**

指定リンクの速度を取得します。

**エンドポイント URL**

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
GSParam	IN	道路速度取得用パラメータ構造体
SpeedAB	OUT	順方向の速度
SpeedBA	OUT	逆方向の速度
NodeA	OUT	ノードポイント構造体
NodeB	OUT	ノードポイント構造体
NWInfo	OUT	計算用道路データ情報構造体

**戻り値**

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	GSParam	GSParam

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 道路速度取得用パラメータ構造体 GSParam

道路速度取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
LinkCode	String	R	速度を取得するリンクのリンクコード
SpeedFlag	int	R	速度フラグ 取得する速度を指定します。 両方指定した場合はメモリ上の速度が優先されます 16 : メモリ上の速度 32 : ファイル上の速度

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## (3) ノード A から B への速度 SpeedAB

計算終了後、ノード A から B への速度が格納されます。

単位は 0.1km/h です。

## (4) ノード B から A への速度 SpeedAB

計算終了後、ノード B から A への速度が格納されます。

単位は 0.1km/h です。

## (5) ノードポイント構造体 NodeA

計算終了後、リンクの始点情報が格納されます。

メンバ	データ型	アクセス	説 明
Node	string	W	ノードコード
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (6) ノードポイント構造体 NodeB

計算終了後、リンクの終点情報が格納されます。

構造体は（5）NodeA と同じです。

## (7) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## 6. 4. 10 道路速度設定サービス

**SetSpeed2(Environ As Environ, SSParam As SSParam, OldSpeedAB As int, OldSpeedBA As int, NWInfo As NWInfo) As int**

指定リンクの速度を設定します。

## エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
SSParam	IN	道路速度設定用パラメータ構造体
OldSpeedAB	OUT	設定前の順方向の速度
OldSpeedBA	OUT	設定前の逆方向の速度
NWInfo	OUT	計算用道路データ情報構造体

## 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

## 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	SSParam	SSParam

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 道路速度設定用パラメータ構造体 SSParam

道路速度設定を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
LinkCode	String	R	速度を取得するリンクのリンクコード
SpeedFlag	int	R	速度フラグ 変更方法と対象を指定します。 16 : メモリ上の速度のみ変更 (指定しない場合は、メモリ・ファイル両方の速度を変更します) 48 : 速度が 0 の方向は変更しない (通行止め変更防止)
SpeedAB	int	R	設定する順方向の速度 (0.1km/h 単位)
SpeedBA	int	R	設定する逆方向の速度 (0.1km/h 単位)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 設定前の順方向の速度 SpeedAB

計算終了後、設定前の順方向の速度が格納されます。

単位は 0.1km/h です。

## (4) 設定前の逆方向の速度 SpeedAB

計算終了後、設定前の逆方向の速度が格納されます。

単位は 0.1km/h です。

## (5) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

6. 4. 1.1 道路データ情報取得サービス**GetNetworkInfo(Environ As Environ, NWInfo As NWInfo) As int**

計算に使用する道路データの情報を取得します。

**エンドポイント URL**

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
NWInfo	OUT	計算用道路データ情報構造体

**戻り値**

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から GetDCWSMessage で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 計算用道路データ情報構造体 NWinfo

計算に使用する計算用道路データの情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 4. 1 2 距離計算サービス用メッセージ文字列取得サービス

**GetDCWSMessage(Environ As Environ, MsgNumber As int) As String**

エラー番号からエラーメッセージを取得します。

**エンドポイント URL**

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
MsgNumber	エラー番号

**戻り値**

指定されたエラー番号の内容を示す文字列。

**備考**

本サービスを呼び出す際には、以下構造体にデータを設定してください。

項番	メンバ	データ型	説 明
(1)	Environ	Environ	環境設定構造体

**(1) 環境設定構造体 Environ**

ユーザ認証に必要なユーザ ID、パスワードを設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	(空文字列)
Port	int	R	(0)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## 6. 5 住所検索サービス

### 6. 5. 1 住所検索サービス

**SearchAddress(Environ As Environ, Param As CommonParam, SearchAddressLevel As int, Items As ArrayOfSearchAddressItem, GeoInfo As GeoInfo ) As int**

漢字住所を解析して経緯度を取得できるサービスです。

#### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
SearchAddressLevel	IN	検索する住所レベル
Items	IN / OUT	住所検索アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

#### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
( 1 )	Environ	Environ
( 2 )	Param	CommonParam
( 3 )	SearchAddressLevel	int
( 4 )	Items	ArrayOfSearchAddressItem



## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

住所検索を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 0 : なし 1 : 全住所も取得する

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 検索住所レベル SearchAddressLevel

住所検索時にどの住所レベルまで検索するかを設定します。

1 : 都道府県、2 : 市区町村、8 : 大字・丁目、16 : 街区 (番地・地番)、32 : 号 (枝番)

## (4) 住所検索アイテム構造体配列 Items

検索する住所を設定します。

検索が終了すると結果が出力されます。

住所検索アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
SearchAddress	string	R	検索住所 例 東京都港区新橋 3-A
AnalyzedAddress	string	W	解析住所 SearchAddress の解析できた部分文字列が出力されます。 例 東京都港区新橋 3-
RestAddress	string	W	残住所 SearchAddress の解析できなかった部分文字列が出力されます。 例 A
AddressLevel	int	W	結果住所レベル 0:失敗、1:都道府県、2:市区町村、8:大字・丁目、16:街区、32:号 例 8
AddressCode	string	W	結果住所コード 例 13103016003
Lon	string	W	結果経度 (単位:度) 例 139.758484
Lat	string	W	結果緯度 (単位:度) 例 35.66276
FullAddress	string	W	結果全住所 検索フラグで設定されていた場合、都道府県からの住所が出力されます。 例 東京都港区新橋 3 丁目
Tag	string		タグ情報 ユーザが自由に使用できる文字列。メッシュ番号や届先名称などの識別情報を設定できます。

\* R: リード、W: ライト、R/W: リード&ライトを表しています。

## (5) 住所データ情報構造体 GeoInfo

使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ～ 8 桁：住所データリリース年月日 9 ～ 1 0 桁：住所データの住所レベル 1 1 ～ 1 2 桁：ユーザの最大住所レベル 1 3 ～ 3 2 桁：未使用

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## 6. 5. 2 住所コード検索サービス

**SearchAddressCode(Environ As Environ, Param As CommonParam, Items As  
ArrayOfSearchAddressCodeItem, GeoInfo As GeoInfo ) As int**

住所コードを検索して経緯度を取得できるサービスです。

### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
Items	IN / OUT	住所コード検索アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
( 1 )	Environ	Environ
( 2 )	Param	CommonParam
( 3 )	Items	ArrayOfSearchAddressCodeItem

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

住所コード検索を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 住所検索アイテム構造体配列 Items

検索する住所コードを設定します。

検索が終了すると結果が出力されます。

メンバ	データ型	アクセス	説 明
SearchAddressCode	string	R	検索住所コード 例 13103016003
AddressItem	AddressItem	W	住所アイテム構造体 検索結果が出力されます。
Tag	string		タグ情報 ユーザが自由に使用できる文字列。 メッシュ番号や届先名称などの識別情報を設定できます。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

AddressItem 住所アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
AddressCode	string	W	住所コード 例 13103016003
AddressLevel	int	W	住所レベル 0 : 失敗、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 8
UpperAddressLevel	int	W	上位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 2
LowerAddressLevel	int	W	下位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 16
Address	string	W	住所 例 新橋3丁目

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Kana	string	W	ひらがな住所 住所コードによっては登録されていない場合があります。 例 しんばし3ちょうめ
KataKana	string	W	カタカナ住所 住所コードによっては登録されていない場合があります。 例 シンバシ3チョウメ
Zip	string	W	郵便番号 住所コードによっては登録されていない場合があります。 例 1050004
Lon	string	W	経度 (単位: 度) 例 139.758484
Lat	string	W	緯度 (単位: 度) 例 35.66276

\* R: リード、W: ライト、R/W: リード&amp;ライトを表しています。

## (4) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ~ 8 桁: 住所データリリース年月日 9 ~ 10 桁: 住所データの住所レベル 11 ~ 12 桁: ユーザの最大住所レベル 13 ~ 32 桁: 未使用

\* R: リード、W: ライト、R/W: リード&amp;ライトを表しています。

### 6. 5. 3 下位住所数取得サービス

**CountLowerAddressByCode(Environ As Environ, Param As CommonParam, SearchAddressCode As string, Count As int, GeoInfo As GeoInfo) As int**

指定された住所コードの下位住所の数を取得できるサービスです。

#### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
SearchAddressCode	IN	検索住所コード
Count	OUT	下位住所数
GeoInfo	OUT	住所データ情報構造体

#### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	Param	CommonParam
(3)	SearchAddressCode	string



## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

下位住所数取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 検索住所コード SearchAddressCode

検索する住所コードを設定します。

空文字列を設定すると都道府県を列挙します。2 桁都道府県コードを設定すると市区町村を列挙します。5 桁市区町村コードを指定すると大字・丁目を列挙します。11 桁大字・丁目コードを指定すると街区を列挙します。16 桁街区コードを指定すると号を列挙します。

## (4) 下位住所数 Count

下位住所数が出力されます。

## (5) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ～ 8 桁：住所データリリース年月日 9 ～ 1 0 桁：住所データの住所レベル 1 1 ～ 1 2 桁：ユーザの最大住所レベル 1 3 ～ 3 2 桁：未使用

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## 6. 5. 4 下位住所列挙サービス

**EnumLowerAddressByCode(Environ As Environ, Param As CommonParam, SearchAddressCode As string, Items As ArrayOfAddressItem, GeoInfo As GeoInfo) As int**

指定された住所コードの下位住所を取得できるサービスです。

## エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
SearchAddressCode	IN	検索住所コード
Items	OUT	住所アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

## 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

## 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	Param	CommonParam
(3)	SearchAddressCode	string

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

下位住所列挙を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 検索住所コード SearchAddressCode

検索する住所コードを設定します。

空文字列を設定すると都道府県を列挙します。2 桁都道府県コードを設定すると市区町村を列挙します。5 桁市区町村コードを指定すると大字・丁目を列挙します。11 桁大字・丁目コードを指定すると街区を列挙します。16 桁街区コードを指定すると号を列挙します。

## (4) 住所アイテム構造体配列 Items

下位住所が出力されます。

住所アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
AddressCode	string	W	住所コード 例 13103016003
AddressLevel	int	W	住所レベル 1 : 都道府県、2 : 市区町村、8 : 大字・丁目、16 : 街区、32 : 号 例 8
UpperAddressLevel	int	W	上位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、8 : 大字・丁目、16 : 街区、32 : 号 例 2
LowerAddressLevel	int	W	下位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、8 : 大字・丁目、16 : 街区、32 : 号 例 16
Address	string	W	住所 例 新橋3丁目
Kana	string	W	ひらがな住所 住所コードによっては登録されていない場合があります。 例 しんばし3ちょうめ
KataKana	string	W	カタカナ住所 住所コードによっては登録されていない場合があります。 例 シンバシ3チョウメ
Zip	string	W	郵便番号 住所コードによっては登録されていない場合があります。 例 1050004
Lon	string	W	経度 (単位 : 度) 例 139.758484
Lat	string	W	緯度 (単位 : 度) 例 35.66276

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## (5) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ～ 8 桁：住所データリリース年月日 9 ～ 1 0 桁：住所データの住所レベル 1 1 ～ 1 2 桁：ユーザの最大住所レベル 1 3 ～ 3 2 桁：未使用

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

### 6. 5. 5 全住所取得サービス

**CreateFullAddressByCode(Environ As Environ, Param As CommonParam, Items As  
ArrayOfCreateFullAddressByCodeItem, GeoInfo As GeoInfo) As int**

指定住所コードの都道府県からの住所を取得できるサービスです。

#### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
Items	IN / OUT	全住所取得アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

#### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
( 1 )	Environ	Environ
( 2 )	Param	CommonParam
( 3 )	Items	ArrayOfCreateFullAddressByCodeItem



## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

全住所取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 全住所取得アイテム構造体配列 Items

検索する住所コードを入力します。

検索後、結果が出力されます。

全住所取得アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
SearchAddressCode	string	R	検索住所コード 例 13103016003000070004
AddressLevel	int	W	結果住所レベル 0:失敗、1:都道府県、2:市区町村、8:大字・丁目、16:街区、32:号 例 8
AddressCode	string	W	結果住所コード 例 13103016003000070004
FullAddress	string	W	結果全住所 都道府県からの住所が出力されます。 例 東京都港区新橋3丁目7-4
PrefAddress	string	W	都道府県住所 例 東京都
CityAddress	string	W	市区町村住所 例 港区
StreetAddress	string	W	大字・丁目住所 例 新橋3丁目
BlockAddress	string	W	番地・地番住所 例 7
HouseAddress	string	W	号・枝番住所 例 4
OtherAddress	string	W	その他住所 号・枝番以下の住所が出力されます。 現在未使用です。
Tag	string		タグ情報 ユーザが自由に使用できる文字列。 メッシュ番号や届先名称などの識別情報を設定できます。

\* R: リード、W: ライト、R/W: リード&ライトを表しています。

## (4) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ～ 8 桁：住所データリリース年月日 9 ～ 1 0 桁：住所データの住所レベル 1 1 ～ 1 2 桁：ユーザの最大住所レベル 1 3 ～ 3 2 桁：未使用

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

### 6. 5. 6 最寄住所取得サービス

**GetNearestAddress(Environ As Environ, Param As CommonParam, Items As  
ArrayOfGetNearestAddressItem, GeoInfo As GeoInfo ) As int**

指定経緯度が一番近い大字・丁目レベル住所を取得できるサービスです。

#### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
Items	IN / OUT	最寄住所取得アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

#### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
( 1 )	Environ	Environ
( 2 )	Param	CommonParam
( 3 )	Items	ArrayOfCreateFullAddressByCodeItem

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

最寄住所取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 検索経緯度と結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 最寄住所取得アイテム構造体配列 Items

検索する経緯度を入力します。

検索後、結果が出力されます。

最寄住所取得アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
SearchLon	string	R	検索経度 例 139.754570
SearchLat	string	R	検索緯度 例 35.654868
FullAddress	string	W	結果全住所 都道府県からの住所が出力されます。 例 東京都港区芝公園 1 丁目
AddressItem	AddressItem	W	結果住所アイテム 検索結果が出力されます。
Distance	double	W	結果距離 (メートル単位) 検索経緯度と結果住所の直線距離 例 48.380828103031156
Tag	string		タグ情報 ユーザが自由に使用できる文字列。 メッシュ番号や届先名称などの識別情報を設定できます。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

AddressItem 住所アイテム構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
AddressCode	string	W	住所コード 例 13103012001
AddressLevel	int	W	住所レベル 0 : 失敗、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 8
UpperAddressLevel	int	W	上位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 2

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
LowerAddressLevel	int	W	下位住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 16
Address	string	W	住所 例 新橋3丁目
Kana	string	W	ひらがな住所 住所コードによっては登録されていない場合があります。 例 しんばし3ちょうめ
KataKana	string	W	カタカナ住所 住所コードによっては登録されていない場合があります。 例 シンバシ3チョウメ
Zip	string	W	郵便番号 住所コードによっては登録されていない場合があります。 例 1050004
Lon	string	W	経度 (単位: 度) 例 139.758484
Lat	string	W	緯度 (単位: 度) 例 35.66276

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

## (4) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ~ 8 桁 : 住所データリリース年月日 9 ~ 10 桁 : 住所データの住所レベル 11 ~ 12 桁 : ユーザの最大住所レベル 13 ~ 32 桁 : 未使用

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

### 6. 5. 7 郵便番号検索サービス

**SearchZip(Environ As Environ, Param As CommonParam, Items As  
ArrayOfSearchZipItem, GeoInfo As GeoInfo) As int**

郵便番号を検索して経緯度を取得できるサービスです。

#### エンドポイント URL

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
Items	IN / OUT	郵便番号検索アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

#### 戻り値

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

#### 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	Param	CommonParam
(3)	Items	ArrayOfSearchZipItem



## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

郵便番号検索を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 1 : 先頭データ。郵便番号に該当する住所データの先頭データを検索します。 8 : 中心位置。郵便番号に該当する住所の平均経緯度位置を検索します。 16 : 中心住所。郵便番号に該当する住所の平均経緯度が一番近い住所データを検索します。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 郵便番号検索アイテム構造体配列 Items

検索する郵便番号を設定します。

検索が終了すると結果が出力されます。

メンバ	データ型	アクセス	説 明
SearchZip	string	R	検索郵便番号 例 105-0004
AddressCode	string	W	結果住所コード 例 13103016000
AddressLevel	int	W	住所レベル 0 : なし、1 : 都道府県、2 : 市区町村、 8 : 大字・丁目、16 : 街区、32 : 号 例 8
FullAddress	string	W	結果全住所 都道府県からの住所が出力されます。 例 東京都港区新橋
Lon	string	W	結果経度 例 139.759094
Lat	string	W	結果緯度 例 35.661605
Tag	string		タグ情報 ユーザが自由に使用できる文字列。 メッシュ番号や届先名称などの識別 情報を設定できます。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (4) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ~ 8 桁 : 住所データリリース年月日 9 ~ 10 桁 : 住所データの住所レベル 11 ~ 12 桁 : ユーザの最大住所レベル 13 ~ 32 桁 : 未使用

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 5. 8 郵便番号対応住所数取得サービス

**CountAddressByZip(Environ As Environ, Param As CommonParam, SearchZip As string, Count As int, GeoInfo As GeoInfo) As int**

指定された郵便番号に対応する住所の数を取得できるサービスです。

**エンドポイント URL**

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
SearchZip	IN	検索郵便番号
Count	OUT	郵便番号対応住所数
GeoInfo	OUT	住所データ情報構造体

**戻り値**

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	Param	CommonParam
(3)	SearchAddressCode	string

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

郵便番号対応住所数取得を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 検索郵便番号 SearchZip

検索する郵便番号を設定します。

## (4) 郵便番号対応住所数 Count

郵便番号対応住所数が出力されます。

## (5) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ～ 8 桁：住所データリリース年月日 9 ～ 1 0 桁：住所データの住所レベル 1 1 ～ 1 2 桁：ユーザの最大住所レベル 1 3 ～ 3 2 桁：未使用

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## 6. 5. 9 郵便番号対応住所列挙サービス

**EnumAddressByZip(Environ As Environ, Param As CommonParam, SearchZip As string, Items As ArrayOfZipItem, GeoInfo As GeoInfo) As int**

指定された郵便番号に対応する住所を列挙するサービスです。

**エンドポイント URL**

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
Param	IN	共通パラメータ構造体
SearchZip	IN	検索郵便番号
Items	OUT	郵便番号アイテム構造体配列
GeoInfo	OUT	住所データ情報構造体

**戻り値**

正常に検索できた場合 0、検索出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetGCWSMessage で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	Param	CommonParam
(3)	SearchAddressCode	String

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	住所検索リスナの IP アドレス
Port	int	R	住所検索リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 共通パラメータ構造体 Param

郵便番号対応住所列挙を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください。
SearchFlag	int	R	検索フラグ 本サービスでは未使用です。

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (3) 検索郵便番号 SearchZip

検索する郵便番号を設定します。

## (4) 郵便番号アイテム構造体配列 Items

郵便番号対応住所が出力されます。

メンバ	データ型	アクセス	説 明
AddressLevel	int	W	住所レベル 1 : 都道府県、2 : 市区町村、8 : 大字・丁目、16 : 街区、32 : 号 例 8
AddressCode	string	W	住所コード 例 13103016000
FullAddress	string	W	全住所 都道府県からの住所が出力されます。 例 東京都港区新橋
Zip	string	W	郵便番号 例 1050004
Lon	string	W	経度 例 139.759094
Lat	string	W	緯度 例 35.661605

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (5) 住所データ情報構造体 GeoInfo

検索に使用した住所データの情報が出力されます。

メンバ	データ型	アクセス	説 明
Folder	string		未使用
Note	string	W	住所データの説明文
GeoID	guid	W	住所データ ID 1 ~ 8 桁 : 住所データリリース年月日 9 ~ 1 0 桁 : 住所データの住所レベル 1 1 ~ 1 2 桁 : ユーザの最大住所レベル 1 3 ~ 3 2 桁 : 未使用

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## 6. 5. 10 住所検索サービス用メッセージ文字列取得サービス

**GetGCWSMessage(Environ As Environ, MsgNumber As int) As string**

エラー番号からエラーメッセージを取得します。

**エンドポイント URL**

住所検索サーバ取得サービスで取得した住所検索サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
MsgNumber	IN	エラー番号

**戻り値**

指定されたエラー番号の内容を示す文字列。

**備考**

本サービスを呼び出す際には、以下構造体にデータを設定してください。

項番	メンバ	データ型	説明
(1)	Environ	Environ	環境設定構造体

**(1) 環境設定構造体 Environ**

ユーザ認証に必要なユーザ ID、パスワードを設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。

メンバ	データ型	設定	説明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string		(空文字列)
Port	int		(0)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## 6. 6 旧サービス

### 6. 6. 1 距離計算サービス

#### InvokeService(DCCCommand As DCCCommand) As boolean

距離計算サービスでは次の 6 処理を行うことができます。

各処理に応じたコマンド文字列を DCCCommand 距離計算コマンド構造体に設定し、距離計算を行います。計算結果は DCCCommand 距離計算コマンド構造体の各メンバ構造体に設定されます。

処 理	コマンド文字列	解 説
ルート計算	CALCROUTE	ロケーション構造体配列に設定されている要素順にルート計算を行い、各要素に所要時間、道のり、高速区間所要時間、高速区間道のり、通行料金を設定します。
最短ルート計算	CALCOPTROUTE	ロケーション構造体配列の最初の要素を出発地、最後の要素を到着地として、その間の要素の最適巡回ルート（巡回セールスマン問題）を求め、各要素に所要時間、道のり、高速区間所要時間、高速区間道のり、通行料金を設定します。(注)
到達圏／流入圏計算	CALCAREA	パラメータ構造体に設定されているスタート地点からの到達圏／流入圏計算を行います。到達圏／流入圏の到達範囲のポリゴンを作成し、その頂点の経緯度をポイント構造体配列に設定するとともに、ロケーション構造体配列に設定されている地点までの所要時間、道のりを求めます。
最寄ノード取得	GETNEARESTNODE	ロケーション構造体配列に設定されている経緯度から最も近傍のノードを探索し、そのノードコードをロケーション構造体に設定します。
近傍ノード列挙	ENUMNEARNODE	パラメータ構造体に設定されているスタート点から近傍のノードを最大 100 点列挙し、ロケーション構造体配列に設定します。
計算用道路データ取得	GETNETWORKINFO	計算用道路データの情報を取得します。

(注) 要素数が約 20 を超えると最適解（最適巡回ルート）を得られない場合があります。

#### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	解説
-------	----

DCCCommand	距離計算コマンド構造体
------------	-------------

## 戻り値

正常に計算できた場合 **True**、計算出来なかった場合 **False**。

## 備考

本サービスを呼び出す際には、**DCCCommand** 距離計算コマンド構造体の各メンバにデータを設定してください。

到達圏／流入圏計算を行う場合、ポリゴン修正機能（**Appendix A**）、ポリゴン始点包含機能（**Appendix B**）も必要に応じてご利用ください。

### ○ 距離計算コマンド構造体 **DCCCommand**

距離計算に必要な全ての情報を設定、または取得します。

項番	メンバ	データ型	説 明
(1)	<b>Command</b>	<b>string</b>	距離計算コマンド文字列
(2)	<b>ResultMessage</b>	<b>string</b>	距離計算結果文字列
(3)	<b>Environ</b>	<b>Environ</b>	環境設定構造体
(4)	<b>Param</b>	<b>Param</b>	パラメータ設定構造体
(5)	<b>Locs</b>	<b>ArrayOfLoc</b>	ロケーション構造体配列
(6)	<b>Pnts</b>	<b>ArrayOfPnt</b>	ポイント構造体配列
(7)	<b>Rns</b>	<b>ArrayOfRn</b>	経由道路名構造体配列
(8)	<b>Trds</b>	<b>ArrayOfTrd</b>	経由有料道路構造体配列
(9)	<b>NWInfo</b>	<b>NWInfo</b>	計算用道路データ情報構造体

### (1) 距離計算コマンド文字列 **Command**

距離計算サーバへどのような処理要求を行うかを設定します。各処理に応じたコマンド文字列を設定してください。

処 理	コマンド文字列
ルート計算	<b>CALCROUTE</b>
最短ルート計算	<b>CALCOPTROUTE</b>
到達圏／流入圏計算	<b>CALCAREA</b>
最寄ノード取得	<b>GETNEARESTNODE</b>
近傍ノード列挙	<b>ENUMNEARNODE</b>
計算用道路データ取得	<b>GETNETWORKINFO</b>

## (2) 距離計算結果文字列 ResultMessage

距離計算を行い、計算結果に何らかの問題が発生した場合に、その状況が文字列として設定されます。

## (3) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	必要	距離計算リスナの IP アドレス
Port	int	必要	距離計算リスナのポート番号

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (4) パラメータ構造体 Param

距離計算を行う際に必要なパラメータを設定します。また、後述するパラメータ構造体アクセス表の”R”の箇所を各距離計算コマンドごとに設定する必要があります。

メンバ	データ型	説 明
Datum	int	経緯度の座標系識別子（現状では 0 を指定してください）
CoordinateFormat	int	経緯度の座標フォーマット識別子（現状では 0 を指定してください）
CalcKind	int	計算方法（1:時間最短、2:距離最短）
NotUseHiway	int	下記値の論理和 高速道路（0:使用する、1:使用しない）、 北海道～青森間、沖縄～鹿児島間のみフェリーを使用（2:使用する）※1、フェリー（4:使用する） 鉄道（8:使用する）、航空（16:使用する） 内航船（鉄道使用の場合、特急使用）（64:使用する）
LocsOutput	int	ロケーション構造体配列出力フラグ （到達圏／流入圏計算で、ロケーション地点までの所要時間、道のりを出力するか指定します。0:出力しない、1:出力する）
PntsOutput	int	ポイント構造体配列出力フラグ （ルート計算／最短ルート計算で、ポイント構造体に走行ルートの出力を行うか指定します。また、到達圏／流入圏計算で、到達ポリゴンの出力を行うか指定します。0:出力しない、1:出力する）
RnsOutput	int	経由道路名構造体配列出力フラグ （ルート計算／最短ルート計算で、経由道路名構造体の出力を行うか指定します。0:出力しない、1:出力する。また、到達圏／流入圏計算で、ポリゴン修正を行うか指定します。0:修正しない、1:修正する ポリゴン修正については <b>Appendix A</b> を参照）
TrdsOutput	int	経由有料道路構造体配列出力フラグ （ルート計算／最短ルート計算で、経由有料道路構造体の出力を行うか指定します。0:出力しない、1:出力する）
CalcRoute_HighwayOutput	int	高速道路区間出力フラグ （ルート計算／最短ルート計算で、ロケーション構造体の高速道路区間の出力を行うか指定します。0:出力しない、1:出力する）
CalcRoute_TollOutput	int	通行料金出力フラグ （ルート計算／最短ルート計算で、ロケーション構造体の通行料金の出力を行うか指定します。0:出力しない、1:出力する）

（次ページへ続く）

メンバ	データ型	説 明
CalcRoute_RouteKind	int	ルート種別指定 (ルート計算／最短ルート計算で、ポイント構造体 に出力する走行ルートの種別を指定します。0:簡易 ルート、1:詳細ルート)
CalcArea_StartNode	string	スタートノードコード (到達圏／流入圏計算で中心となる地点のノード コード。中心地点を経緯度ではなく、直接ノードコ ードで指定する場合に使用します。通常は"0"を設定 してください。また、近傍ノード列挙での中心地点 としても使用します。)
CalcArea_StartLongitude	string	スタート経度 (到達圏／流入圏計算で中心となる地点の経度。上 記スタートノードコードが指定されている場合、本 パラメータは無視されます。また、近傍ノード列挙 での中心経度としても使用します。経度 は、"135.123456"のように度単位小数点以下 6 桁の 文字列で指定してください。)
CalcArea_StartLatitude	string	スタート緯度 (到達圏／流入圏計算で中心となる地点の緯度。上 記スタートノードコードが指定されている場合、本 パラメータは無視されます。また、近傍ノード列挙 での中心緯度としても使用します。経度 は、"35.123456"のように度単位小数点以下 6 桁の文 字列で指定してください。)
CalcArea_AreaKind	int	到達圏／流入圏計算種別 (到達圏／流入圏計算での範囲を時間または距離の どちらで指定するかを指定します。 1:時間圏 2:距離圏)
CalcArea_Range	int	到達圏／流入圏範囲 (到達圏／流入圏計算での範囲を指定します。時間 圏の場合、0.1 秒単位、距離圏の場合、m 単位で指 定します。)
CalcArea_PolygonLevel	int	到達圏／流入圏ポリゴンレベル (到達圏／流入圏計算での到達ポリゴンのポリゴン レベルを指定します。ポリゴンレベルが-1 の場合、 凸型(簡易)ポリゴンを作成します。ポリゴンレベル が 0～20 の場合、凹凸型(通常)ポリゴンを作成しま す。 ポリゴンレベルを高くすると距離計算のレスポンス が遅くなります。また、ポリゴンレベルを高くする とポリゴンの作成に失敗する場合があります。ポリ ゴンの作成に失敗した場合は、凸型(簡易)ポリゴン が出力されます。)
CalcArea_AreaDirection	int	範囲方向指定 (到達圏または流入圏のどちらの計算を行うかを指 定します。0:到達圏 4096:流入圏)

※ 1 本州—北海道間、本州—沖縄間の道路距離を簡易的に計算するための設定

各距離計算コマンドでのパラメータ構造体メンバへのアクセスは次のとおりです。

＜パラメータ構造体アクセス表＞

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列挙	計算用道路データ取得	備考
Datum	R	R	R	R	R		
CoordinateFormat	R	R	R	R	R		
CalcKind	R	R	R				
NotUseHiway	R	R	R				
LocsOutput			R				出力を行うと計算時間が長くなります
PntsOutput	R	R	R				
RnsOutput	R	R	R				
TrdsOutput	R	R					
CalcRoute_HighwayOutput	R	R					
CalcRoute_TollOutput	R	R					
CalcRoute_RouteKind	R	R					詳細ルートは、簡易ルートより計算時間が長くなります
CalcArea_StartNode			R		R		
CalcArea_StartLon			R		R		
CalcArea_StartLat			R		R		
CalcArea_AreaKind			R				
CalcArea_Range			R				範囲を大きくすると計算時間が長くなります
CalcArea_PolygonLevel			R				ポリゴンレベルを大きくすると計算時間が長くなります
CalcArea_AreaDirection			R				

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (5) ロケーション構造体配列 Locs

特定の場所（ノードコード、経緯度）を設定すると同時に、その場所への所要時間、道のり、通行料金等の情報を計算結果として取得します。また、後述するロケーション構造体アクセス表の”R/W”の箇所を各距離計算コマンドごとに設定する必要があります。

＜ロケーション構造体メンバ表＞

メンバ	データ型	説 明
Node	string	ノードコード (ロケーションの地点を経緯度ではなく、直接ノードコードで指定する場合に使用します。通常は”0”を設定してください。また、”0”を設定し距離計算をおこなった場合、Lon,Lat メンバで指定された地点の最寄ノードのノードコードが設定されます。)
Lon	string	経度 (ロケーションの経度。Node メンバが指定されている場合、本メンバは無視されます。経度は、”135.123456”のように度単位小数点以下6桁の文字列で指定してください。)
Lat	string	緯度 (ロケーションの緯度。Node メンバが指定されている場合、本メンバは無視されます。緯度は、”35.123456”のように度単位小数点以下6桁の文字列で指定してください。)
Time	int	所要時間 (分単位)
DSec	int	所要時間 (0.1 秒単位)
Dist	int	道のり (m 単位)
HighwayTime	int	高速道路区間の所要時間 (分単位)
HighwayDSec	int	高速道路区間の所要時間 (0.1 秒単位)
HighwayDist	int	高速道路区間の道のり (m 単位)
Toll_SS	int	二輪・軽の通行料金 (円単位)
Toll_S	int	普通車の通行料金 (円単位)
Toll_M	int	中型車の通行料金 (円単位)
Toll_L	int	大型車の通行料金 (円単位)
Toll_LL	int	特大車の通行料金 (円単位)
Tag	string	タグ情報 (ユーザが自由に使用できる文字列。メッシュ番号や届先名称などロケーションを識別するための情報を設定します。)



各距離計算コマンドでのロケーション構造体メンバへのアクセスは次のとおりです。

＜ロケーション構造体アクセス表＞

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列挙	計算用道路データ取得	備考
Node	R/W	R/W	R/W	W	R/W		
Lon	R	R	R	R	R/W		
Lat	R	R	R	R	R/W		
Time	W	W	W				
DSec	W	W	W				
Dist	W	W	W				
HighwayTime	W	W					パラメータ構造体の CalcRoute_HighwayOutput が設定されている場合、更新されます
HighwayDSec	W	W					
HighwayDist	W	W					
Toll_SS	W	W					パラメータ構造体の CalcRoute_TollOutput が設定されている場合、更新されます
Toll_S	W	W					
Toll_M	W	W					
Toll_L	W	W					
Toll_LL	W	W					
Tag							

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (6) ポイント構造体配列 Pnts

経緯度が設定されます。

ルート計算／最短ルート計算では、ルート情報の経緯度が設定されます。到達圏／流入圏計算では、到達範囲ポリゴンの各頂点の座標が設定されます。また、後述するポイント構造体アクセス表の”R/W”の箇所を各距離計算コマンドごとに設定する必要があります。

## &lt;ポイント構造体メンバ表&gt;

メンバ	データ型	説 明
Lon	string	経度 (経度は、”135.123456”のように度単位小数点以下 6 桁の文字列で保持します。)
Lat	string	緯度 (緯度は、”35.123456”のように度単位小数点以下 6 桁の文字列で保持します。)

各距離計算コマンドでのポイント構造体メンバへのアクセスは次のとおりです。

## &lt;ポイント構造体アクセス表&gt;

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列举	計算用道路データ取得	備考
Lon	W	W	R/W				
Lat	W	W	R/W				

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (7) 経由道路名構造体配列 Rns

ルート計算／最短ルート計算の実行により得られた走行ルートの道路名／交差点名が設定されます。

## ＜経由道路名構造体メンバ表＞

メンバ	データ型	説 明
LocFlag	int	ロケーションフラグ (本構造体がロケーション構造体で指定された地点であることを示します。)
LinkFlag	int	リンクフラグ (本構造体がリンクであることを示します。)
NLCode	int	ノードリンク番号 (リンクフラグが 0 の場合、ノードコード。リンクフラグが 1 の場合、リンク番号)
Name	string	名称 (ロケーションフラグが 1 の場合、ロケーションの Tag 情報。リンクフラグが 0 の場合、交差点名称。リンクフラグが 1 かつインターチェンジタイプが 0 の場合、道路名称。リンクフラグが 1 かつインターチェンジタイプが 0 でない場合、インターチェンジ名称)
Time	int	所要時間 (分単位)
DSec	int	所要時間 (1/10 秒単位)
Dist	int	道のり (m 単位)
ICType	int	インターチェンジタイプ
Toll_SS	int	二輪・軽の通行料金 (円単位)
Toll_S	int	普通車の通行料金 (円単位)
Toll_M	int	中型車の通行料金 (円単位)
Toll_L	int	大型車の通行料金 (円単位)
Toll_LL	int	特大車の通行料金 (円単位)
Lon	string	経度 (経度は、"135.123456"のように度単位小数点以下 6 桁の文字列で保持します。)
Lat	string	緯度 (緯度は、"35.123456"のように度単位小数点以下 6 桁の文字列で保持します。)

各距離計算コマンドでの経由道路名構造体メンバへのアクセスは次のとおりです。

＜経由道路名構造体アクセス表＞

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列挙	計算用道路データ取得	備考
LocFlag	W	W					
LinkFlag	W	W					
NLCode	W	W					
Name	W	W					
Time	W	W					
DSec	W	W					
Dist	W	W					
ICType	W	W					
Toll_SS	W	W					
Toll_S	W	W					
Toll_M	W	W					
Toll_L	W	W					
Toll_LL	W	W					
Lon	W	W					
Lat	W	W					

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (8) 経由有料道路構造体配列 Trds

ルート計算／最短ルート計算の実行により得られた走行ルートの有料道路の情報が設定されます。

## &lt;経由有料道路構造体メンバ表&gt;

メンバ	データ型	説 明
InRdName	string	入口または課金の有料道路名称 (入口または課金の料金所名称、出口の有料道路名称、出口の料金所名称が空白の場合、ロケーションのタグ情報が格納されます)
InICName	string	入口または課金の料金所名称
OutRdName	string	出口の有料道路名称
OutICName	string	出口の料金所名称
Toll_SS	int	二輪・軽の通行料金 (円単位)
Toll_S	int	普通車の通行料金 (円単位)
Toll_M	int	中型車の通行料金 (円単位)
Toll_L	int	大型車の通行料金 (円単位)
Toll_LL	int	特大車の通行料金 (円単位)
Dist	int	距離 (有料道路のキロポスト 0 または有効な値でない場合があります)

各距離計算コマンドでの経由有料道路構造体メンバへのアクセスは次のとおりです。

＜経由有料道路構造体アクセス表＞

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列举	計算用道路データ取得	備考
InRdName	W	W					
InICName	W	W					
OutRdName	W	W					
OutICName	W	W					
Toll_SS	W	W					本構造体がロケーションの場合、通行料金の合計金額が設定されます
Toll_S	W	W					
Toll_M	W	W					
Toll_L	W	W					
Toll_LL	W	W					
Dist	W	W					

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (9) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	説 明
Folder	string	計算用道路データのフォルダ
Note	string	計算用道路データの説明
IsDetailExist	boolean	詳細ルート対応フラグ
IsTollExist	boolean	通行料金対応フラグ
NWID	guid	計算用道路データ ID
NWID_WithoutSpeed	guid	(道路速度を考慮しない) 計算用道路データ ID

各距離計算コマンドでの計算用道路データ情報構造体メンバへのアクセスは次のとおりです。

<計算用道路データ情報構造体アクセス表>

メンバ	ルート計算	最短ルート計算	到達圏／流入圏計算	最寄ノード取得	近傍ノード列挙	計算用道路データ取得	備考
Folder	W	W	W	W	W	W	
Note	W	W	W	W	W	W	
IsDetailExist	W	W	W	W	W	W	
IsTollExist	W	W	W	W	W	W	
NWID	W	W	W	W	W	W	
NWID_WithoutSpeed	W	W	W	W	W	W	

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## 6. 6. 2 到達圏／流入圏計算サービス

**CalcArea(Environ As Environ, CAParam As CAParam, Pnts As ArrayOfPnt, NWInfo As NWInfo) As int**

※ このサービスは互換性のために残されています。今後新たに到達圏／流入圏計算サービスを利用するプログラムを作成する場合は **CalcArea3** を使用してください。

また単一ポリゴン計算時に到達圏／流入圏内ノード数が 3 万未満の場合、自動でポリゴン補間機能が有効になります。ポリゴン補間機能については **CalcArea2** の説明をご覧ください。

**CAParam** 到達圏／流入圏計算用パラメータ構造体に設定されている内容で到達圏／流入圏計算を行います。到達圏／流入圏計算結果は、**Pnts** ポイント構造体配列に設定されます。**Pnts** ポイント構造体配列には、到達／流入範囲ポリゴンの各頂点の座標が格納されます。

### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
CAParam	到達圏／流入圏計算用パラメータ構造体
Pnts	到達圏／流入圏計算結果を取得するポイント構造体配列
NWInfo	到達圏／流入圏計算時に使用した計算用道路データ構造体

### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から **GetDCWSMessage** で取得できます。



## 備考

本サービスを呼び出す際には、以下構造体にデータを設定してください。  
また、ポリゴン修正機能（Appendix A）も必要に応じてご利用ください。

項番	メンバ	データ型	説 明
(1)	Environ	Environ	環境設定構造体
(2)	CAParam	CAParam	到達圏／流入圏計算用パラメータ構造体

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	必要	距離計算リスナの IP アドレス
Port	int	必要	距離計算リスナのポート番号

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 到達圏／流入圏計算用パラメータ構造体 CAParam

到達圏／流入圏計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 (現状では 0 を指定してください)
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 (現状では 0 を指定してください)
CalcKind	int	R	計算方法 (1:時間最短、2:距離最短)
NotUseHiway	int	R	下記値の論理和 高速道路 (0:使用する、1:使用しない) 北海道～青森間、沖縄～鹿児島間のみフェリーを使用 (2:使用する) フェリー (4:使用する) 鉄道 (8:使用する) 航空 (16:使用する) 内航船 (鉄道使用の場合、特急使用) (64:使用する)
PolygonKind	int	R	ポリゴン種別 (1:単一ポリゴン、2:複数ポリゴン)
StartGeoPnt	GeoPnt	R	到達圏／流入圏計算の中心地点情報
AreaKind	int	R	到達圏／流入圏計算種別 (到達圏／流入圏計算での範囲を時間または距離のどちらで指定するかを指定します。 1:時間圏 2:距離圏)
AreaRange	int	R	到達圏／流入圏範囲 (到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。)
PolygonLevel	int	R	到達圏／流入圏ポリゴンレベル (到達圏／流入圏計算での到達ポリゴンのポリゴンレベルを指定します。ポリゴンレベルが-1 の場合、凸型(簡易)ポリゴンを作成します。ポリゴンレベルが 0～20 の場合、凹凸型(通常)ポリゴンを作成します。)
DonutPolygonLevel	int	R	到達圏／流入圏ドーナツポリゴンレベル (-2:表示しない、-1: 凸型(簡易)ポリゴン、0～20: 凹凸型(通常)ポリゴン)
AreaDirection	int	R	範囲方向指定 (到達圏または流入圏のどちらの計算を行うかを指定します。0:到達圏 4096:流入圏)

到達圏／流入圏計算用パラメータ構造体のメンバの **StartGeoPnt** のメンバとアクセスは次のとおりです。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列 (2011/2/1 現在 未実装)
GeoLevel	int		住所レベル (検索結果の当該住所の住所レベル 1: 都道府県、2:市区町村、8:大字・丁目、16: 街区、32:号)
AddrCode	string		住所コード
Address	string		解析住所文字列 (住所検索において、解析できた住所部分が設定される)
RestAddress	string		非解析住所文字列 (住所検索において、解析出来なかった住所部分が設定される)
ZipCode	string		郵便番号
Node	string	R	ノードコード (地点を経緯度ではなく、直接ノードコードで指定する場合に使用します。通常は”0”を設定してください。)
Lon	string	R	経度 (Node メンバが指定されている場合、本メンバは無視されます。経度は、”135.123456”のように度単位小数点以下6桁の文字列で指定してください。)
Lat	string	R	緯度 (Node メンバが指定されている場合、本メンバは無視されます。緯度は、”35.123456”のように度単位小数点以下6桁の文字列で指定してください。)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

本サービスの呼び出しが成功すると、ポイント構造体配列 **Pnts** の各要素に次のようなデータが設定されます。到達圏／流入圏計算では、到達範囲ポリゴンの各頂点の座標が設定されます。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度 (経度は、"135.123456"のように度単位小数点以下 6 桁の文字列で保持します。)
Lat	string	W	緯度 (緯度は、"35.123456"のように度単位小数点以下 6 桁の文字列で保持します。)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

また、計算用道路データ構造体 **NWInfo** の各要素に次のようなデータが設定されます。

メンバ	データ型	説 明
Folder	string	計算用道路データの格納フォルダ
Note	string	計算用道路データの名称
IsDetailExist	boolean	詳細ルート出力可能フラグ
IsTollExist	boolean	通行料金出力可能フラグ
NWID	guid	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

## 6. 6. 3 到達圏／流入圏計算サービス (2)

**CalcArea2(Environ As Environ, CA2Param As CA2Param, Pnts As ArrayOfPnt, NWInfo As NWInfo) As int**

※ このサービスは互換性のために残されています。今後新たに到達圏／流入圏計算サービスを利用するプログラムを作成する場合は **CalcArea3** を使用してください。

**CA2Param** 到達圏／流入圏計算用パラメータ構造体に設定されているスタート地点からの到達圏／流入圏計算を行います。**Pnts** ポイント構造体配列には、到達／流入範囲ポリゴンの各頂点の座標が格納されます。

**エンドポイント URL**

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
CA2Param	IN / OUT	到達圏／流入圏計算用パラメータ構造体 (ver.2)
Pnts	OUT	ポイント構造体配列
NWInfo	OUT	計算用道路データ情報構造体

**戻り値**

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。

エラー内容はエラー番号から **GetDCWSMessage** で取得できます。

**備考**

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	CA2Param	CA2Param

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 到達圏／流入圏計算用パラメータ構造体 (ver.2) CA2Param

到達圏／流入圏計算を行う際に必要なパラメータを設定します。

計算後に使用されたポリゴン詳細設定が出力されます。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短
PolygonKind	int	R	ポリゴン種別 1 : 単一ポリゴン、2 : 複数ポリゴン
StartGeoPnt	GeoPnt	R	到達圏／流入圏計算の中心地点

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
NotUseHiway	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用する</p> <p>1 : 高速道路を使用しない</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
AreaKind	int	R	<p>到達圏／流入圏計算種別</p> <p>計算範囲の種別を指定します。</p> <p>1 : 時間圏、2 : 距離圏</p>
AreaRange	int	R	<p>到達圏／流入圏範囲</p> <p>到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。</p>
PolygonLevel	int	R	<p>到達圏／流入圏ポリゴンレベル</p> <p>到達ポリゴンのポリゴンレベルを指定します。</p> <p>-1 : 凸型(簡易)ポリゴン</p> <p>0～20 : 凹凸型(通常)ポリゴン (大きいほど細かくなります)</p>
DonutPolygonLevel	int	R	<p>到達圏／流入圏ドーナツポリゴンレベル</p> <p>到達ポリゴン内で到達できない範囲のポリゴンレベルを指定します。</p> <p>-2 : 表示しない</p> <p>-1 : 凸型(簡易)ポリゴン</p> <p>0～20 : 凹凸型(通常)ポリゴン (大きいほど細かくなります)</p>

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
AreaDirection	int	R	範囲方向 到達圏または流入圏を指定します。 0：到達圏、4096：流入圏
PolygonDetail	int	R/W	ポリゴン詳細設定 下記値の論理和 計算時に使用する機能を指定してください。計算後は使用された機能が設定されています。 0：詳細機能を使用しない 1：中間リンク補間 2：末端リンク補間

※ 1 本州—北海道間、本州—沖縄間の道路距離を簡易的に計算するための設定

\* R：リード、W：ライト、R/W：リード&amp;ライトを表しています。

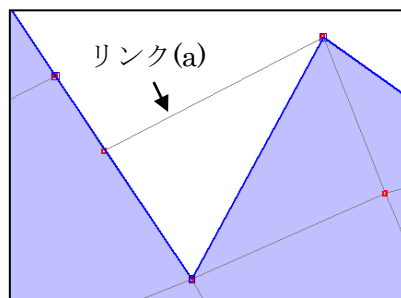
## ※ ポリゴン補間機能について

通常、ポリゴンは到達圏内のノードだけを結んで作成しますが、ポリゴン補間機能を使用すると補助となるポイントを追加してポリゴンを作成することができます。

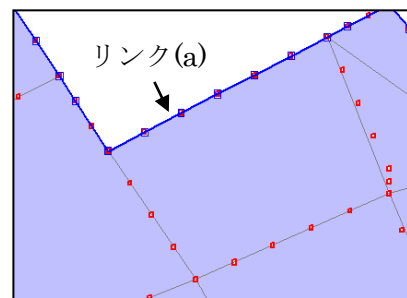
## ・ 中間リンク補間

到達圏内ノード間のリンク上にポイントを追加してポリゴンを作成します。

通常のポリゴン



中間リンク補間使用



上左図は通常のポリゴン作成機能を使用しています。到達圏内ノードのみを考慮するのでリンク(a)を通行することが可能でも到達圏外になっています。

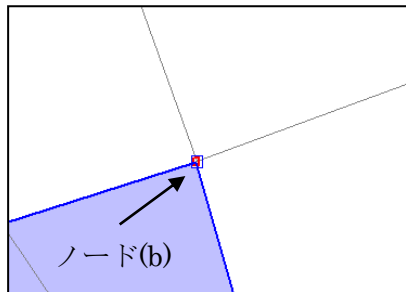
上右図は中間リンク補間機能を使用しています。到達圏内ノード間のリンク上にポイントを追加しているのでリンク(a)に沿ってポリゴンが作成されています。



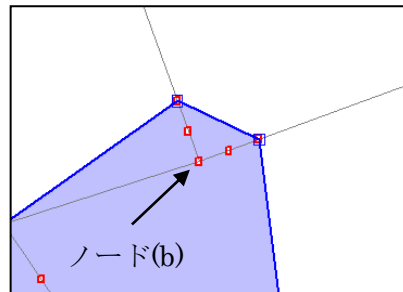
- ・ 末端リンク補間

到達圏の末端リンク上にポイントを追加してポリゴンを作成します。

通常のポリゴン



末端リンク補間使用



例えば 10 分到達圏の計算をしてノード(b)に 9 分で到達できますが、その先のノードには 1 分で到達できないとします。

上左図は通常のポリゴン作成機能を使用しています。到達圏内ノードのみを考慮してポリゴンを作成しているため、端数の 1 分はポリゴンに反映されません。

上右図は末端リンク補間機能を使用しています。ノード(b)の先のリンク上に端数の 1 分で到達可能なポイントを追加してポリゴンを作成しています。

- ・ 制限

中間リンク補間機能と末端リンク補間機能が有効になるのは、単一ポリゴンかつ到達圏内ノード数が 10 万未満の場合です。ポリゴン補間機能が使用されたかどうか確認したい場合は、計算後に CA2Param の PolygonDetail を確認してください。ポリゴン補間機能が使用されなかった場合は設定したフラグが消去されています。

到達圏／流入圏計算用パラメータ構造体のメンバの **StartGeoPnt** のメンバとアクセスは次のとおりです。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列(未使用)
GeoLevel	int		住所レベル(未使用)
AddrCode	string		住所コード(未使用)
Address	string		解析住所文字列(未使用)
RestAddress	string		非解析住所文字列(未使用)
ZipCode	string		郵便番号(未使用)
Node	string	R	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。
Lon	string	R	経度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。
Lat	string	R	緯度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (3) ポイント構造体配列 Pnts

計算後に到達／流入範囲ポリゴンの各頂点の座標が設定されます。

ポイント構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
Lon	string	W	経度（度単位小数点以下 6 桁）
Lat	string	W	緯度（度単位小数点以下 6 桁）

\* R：リード、W：ライト、R/W：リード&ライトを表しています。



ポリゴンデータの格納順は左図のようになっています。

各ポリゴンの区切りとして、Lon と Lat に数値として 0 が格納されているポイント構造体を使用されます。Lon と Lat は数値に変換して 0 かどうか判定してください。

まず到達範囲ポリゴンが格納されています。

次に区切りが一つある場合は、直前の到達範囲ポリゴン内のドーナツポリゴンが格納されています。

区切りが二つ続く場合は、次の到達範囲ポリゴンが格納されています。

区切りが三つ続く場合はポリゴンデータの終了を意味します。

## (4) 計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string	W	未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

#### 6. 6. 4 片道一括計算サービス

**CalcOneWay(Environ As Environ, COWParam As COWParam, GeoLocs As ArrayOfGeoLoc, NWInfo As NWInfo) As int**

※ このサービスは互換性のために残されています。今後新たに片道一括計算サービスを利用するプログラムを作成する場合は **CalcOneWay3** を使用してください。

中心地点から複数の地点までの距離計算、または複数の地点から中心地点までの距離計算を行います。

#### エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	解説
Environ	ユーザ ID およびパスワードが設定された環境設定構造体
COWParam	片道一括計算用パラメータ構造体
GeoLocs	片道一括計算結果を取得するジオロケーション構造体配列
NWInfo	片道一括計算時に使用した計算用道路データ構造体

#### 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から **GetDCWSMessage** で取得できます。

## 備考

本サービスを呼び出す際には、以下構造体にデータを設定してください。

項番	メンバ	データ型	説 明
(1)	Environ	Environ	環境設定構造体
(2)	COWParam	COWParam	片道一括計算用パラメータ構造体
(3)	GeoLocs	ArrayOfGeoLoc	ジオロケーション構造体配列

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	必要	距離計算リスナの IP アドレス
Port	int	必要	距離計算リスナのポート番号

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 片道一括計算用パラメータ構造体 COWParam

片道一括計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子（現状では 0 を指定してください）
CoordinateFormat	int	R	経緯度の座標フォーマット識別子（現状では 0 を指定してください）
CalcKind	int	R	計算方法（1:時間最短、2:距離最短）
NotUseHiway	int	R	下記値の論理和 高速道路（0:使用する、1:使用しない） 北海道～青森間、沖縄～鹿児島間のみ フェリーを使用（2:使用する） フェリー（4:使用する） 鉄道（8:使用する） 航空（16:使用する） 内航船（鉄道使用の場合、特急使用）（64:使用する）
StartGeoPnt	GeoPnt	R	片道一括計算の中心地点情報
AreaKind	int	R	到達圏／流入圏計算種別 （到達圏／流入圏計算での範囲を時間または距離のどちらで指定するかを指定します。 1:時間圏 2:距離圏）
AreaRange	int	R	到達圏／流入圏範囲 （到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。）
AreaDirection	int	R	範囲方向指定 （到達圏または流入圏のどちらの計算を行うかを指定します。0:到達圏 4096:流入圏）

片道一括計算用パラメータ構造体のメンバの **StartGeoPnt** のメンバとアクセスは次のとおりです。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列 (2011/2/1 現在 未実装)
GeoLevel	int		住所レベル (検索結果の当該住所の住所レベル 1: 都道府県、2: 市区町村、8: 大字・丁目、16: 街区、32: 号)
AddrCode	string		住所コード
Address	string		解析住所文字列 (住所検索において、解析できた住所部分が設定される)
RestAddress	string		非解析住所文字列 (住所検索において、解析出来なかった住所部分が設定される)
ZipCode	string		郵便番号
Node	string	R/W	ノードコード (地点を経緯度ではなく、直接ノードコードで指定する場合に使用します。通常は”0”を設定してください。また、”0”を設定し距離計算をおこなった場合、Lon, Lat メンバで指定された地点の最寄ノードのノードコードが設定されます。)
Lon	string	R/W	経度 (Node メンバが指定されている場合、本メンバは無視されます。経度は、”135.123456”のように度単位小数点以下6桁の文字列で指定してください。)
Lat	string	R/W	緯度 (Node メンバが指定されている場合、本メンバは無視されます。緯度は、”35.123456”のように度単位小数点以下6桁の文字列で指定してください。)

\* R : リード、W : ライト、R/W : リード&ライトを表しています。



## (3) ジオロケーション構造体配列 GeoLocs

住所検索処理、住所コード検索処理、郵便番号検索処理を行ない、ノードコード、経緯度を自動設定し、その場所への所要時間、道のりの情報を計算結果として取得します。住所検索処理、住所コード検索処理、郵便番号検索処理では、検索する文字列を設定します。直接ノードコード、経緯度を指定することも可能です。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列 (2011/2/1 現在 未実装)
GeoLevel	int		住所レベル (検索結果の当該住所の住所レベル 1:都道府県、2:市区町村、8:大字・丁目、16:街区、32:号)
AddrCode	string		住所コード
Address	string		解析住所文字列 (住所検索において、解析できた住所)
RestAddress	string		非解析住所文字列 (住所検索において、解析出来なかった住所)
ZipCode	string		郵便番号
Node	string	R/W	ノードコード (ジオロケーションの地点を経緯度ではなく、直接ノードコードで指定する場合に使用します。通常は"0"を設定してください。)
Lon	string	R/W	経度 (ジオロケーションの経度。Node メンバが指定されている場合、本メンバは無視されます。)
Lat	string	R/W	緯度 (ジオロケーションの緯度。Node メンバが指定されている場合、本メンバは無視されます。)
Time	int	W	所要時間 (分単位)
DSec	int	W	所要時間 (0.1 秒単位)
Dist	int	W	道のり (m 単位)
HighwayTime	int		高速道路区間の所要時間 (分単位)
HighwayDSec	int		高速道路区間の所要時間 (0.1 秒単位)
HighwayDist	int		高速道路区間の道のり (m 単位)
Toll_SS	int		二輪・軽の通行料金 (円単位)
Toll_S	int		普通車の通行料金 (円単位)
Toll_M	int		中型車の通行料金 (円単位)
Toll_L	int		大型車の通行料金 (円単位)
Toll_LL	int		特大車の通行料金 (円単位)
Tag	string		タグ情報

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

また、計算用道路データ構造体 NWInfo の各要素に次のようなデータが設定されます。

メンバ	データ型	説 明
Folder	string	計算用道路データの格納フォルダ
Note	string	計算用道路データの名称
IsDetailExist	boolean	詳細ルート出力可能フラグ
IsTollExist	boolean	通行料金出力可能フラグ
NWID	guid	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

## 6. 6. 5 片道一括計算サービス (2)

**CalcOneWay2(Environ As Environ, COW2Param As COW2Param, GeoLocs As ArrayOfGeoLoc, NWInfo As NWInfo) As int**

※ このサービスは互換性のために残されています。今後新たに片道一括計算サービスを利用するプログラムを作成する場合は CalcOneWay3 を使用してください。

中心地点から複数の地点までの距離計算、または複数の地点から中心地点までの距離計算を行います。

## エンドポイント URL

距離計算サーバ取得サービスで取得した距離計算サーバの URL

パラメータ	入出力	解説
Environ	IN	環境設定構造体
COW2Param	IN	片道一括計算用パラメータ構造体 (ver.2)
GeoLocs	OUT	ジオロケーション構造体配列
NWInfo	OUT	計算用道路データ情報構造体

## 戻り値

正常に計算できた場合 0、計算出来なかった場合 0 以外（エラー番号）。  
エラー内容はエラー番号から GetDCWSMessage で取得できます。

## 備考

本サービスを呼び出す際には、以下の引数にデータを設定してください。

項番	引数名	データ型
(1)	Environ	Environ
(2)	COW2Param	COW2Param
(3)	GeoLocs	ArrayOfGeoLoc

## (1) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや距離計算リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。距離計算リスナの IP アドレス、ポート番号は、距離計算サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	アクセス	説 明
UserID	string	R	ユーザ ID
UUID	string	R	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	R	距離計算リスナの IP アドレス
Port	int	R	距離計算リスナのポート番号

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

(注) 距離計算サーバ取得サービスで、距離計算サーバのエンドポイント URL が得られます。距離計算サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (2) 片道一括計算用パラメータ構造体 (ver.2) COW2Param

片道一括計算を行う際に必要なパラメータを設定します。

メンバ	データ型	アクセス	説 明
Datum	int	R	経緯度の座標系識別子 結果経緯度の測地系を指定します。 0 : 日本測地系、1 : 世界測地系
CoordinateFormat	int	R	経緯度の座標フォーマット識別子 現状では 0 を指定してください
CalcKind	int	R	計算方法 1 : 時間最短、2 : 距離最短
StartGeoPnt	GeoPnt	R	片道一括計算の中心地点情報
AreaKind	int	R	到達圏／流入圏計算種別 計算範囲の種別を指定します。 1 : 時間圏、2 : 距離圏

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
NotUseHiway	int	R	<p>使用交通機関フラグ</p> <p>計算用道路データによって使用できる交通機関は異なります。</p> <p>下記値の論理和です。</p> <p>0 : 高速道路を使用する</p> <p>1 : 高速道路を使用しない</p> <p>2 : 北海道～青森間、沖縄～鹿児島間のみフェリーを使用する※ 1</p> <p>4 : フェリーを使用する</p> <p>8 : 鉄道を使用する</p> <p>16 : 航空を使用する</p> <p>64 : 内航船を使用する、または旅客鉄道を使用する場合特急を使用する</p>
AreaRange	int	R	<p>到達圏／流入圏範囲</p> <p>到達圏／流入圏計算での範囲を指定します。時間圏の場合、0.1 秒単位、距離圏の場合、m 単位で指定します。</p>
AreaDirection	int	R	<p>範囲方向</p> <p>到達圏または流入圏を指定します。</p> <p>0 : 到達圏、4096 : 流入圏</p>
HighwayOutput	int	R	<p>高速道路区間出力フラグ</p> <p>ジオロケーション構造体の高速道路区間の出力を行うか指定します。</p> <p>0 : 出力しない、1 : 出力する</p>
TollOutput	int	R	<p>通行料金出力フラグ</p> <p>ジオロケーション構造体の通行料金の出力を行うか指定します。</p> <p>0 : 出力しない、1 : 出力する</p>

※ 1 本州―北海道間、本州―沖縄間の道路距離を簡易的に計算するための設定

\* R : リード、W : ライト、R/W : リード&amp;ライトを表しています。

片道一括計算用パラメータ構造体のメンバの **StartGeoPnt** のメンバとアクセスは次のとおりです。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列(未使用)
GeoLevel	int		住所レベル(未使用)
AddrCode	string		住所コード(未使用)
Address	string		解析住所文字列(未使用)
RestAddress	string		非解析住所文字列(未使用)
ZipCode	string		郵便番号(未使用)
Node	string	R	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。
Lon	string	R	経度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。
Lat	string	R	緯度（度単位小数点以下 6 桁） Node を指定した場合、使用されません。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

## (3) ジオロケーション構造体配列 GeoLocs

片道一括計算を行うロケーションを設定します。

計算終了後結果が格納されます。

ジオロケーション構造体の内容は以下の通りです。

メンバ	データ型	アクセス	説 明
SearchString	string		検索文字列 (未使用)
GeoLevel	int		住所レベル (未使用)
AddrCode	string		住所コード (未使用)
Address	string		解析住所文字列 (未使用)
RestAddress	string		非解析住所文字列 (未使用)
ZipCode	string		郵便番号 (未使用)
Node	string	R/W	ノードコード 地点をノードコードで指定する場合に使用します。通常は”0”を設定してください。 計算後は最寄ノードコードが出力されます。
Lon	string	R/W	経度 (度単位小数点以下 6 桁) Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの経度が出力されます。
Lat	string	R/W	緯度 (度単位小数点以下 6 桁) Node を指定した場合、使用されません。 Node を指定して計算した場合、最寄ノードの緯度が出力されます。
Time	int	W	所要時間 (分単位)
DSec	int	W	所要時間 (0.1 秒単位)
Dist	int	W	道のり (m 単位)
HighwayTime	int	W	高速道路区間の所要時間 (分単位)
HighwayDSec	int	W	高速道路区間の所要時間 (0.1 秒単位)
HighwayDist	int	W	高速道路区間の道のり (m 単位)

次ページに続く

前ページからの続き

メンバ	データ型	アクセス	説 明
Toll_SS	int	W	二輪・軽の通行料金（円単位）
Toll_S	int	W	普通車の通行料金（円単位）
Toll_M	int	W	中型車の通行料金（円単位）
Toll_L	int	W	大型車の通行料金（円単位）
Toll_LL	int	W	特大車の通行料金（円単位）
Tag	string		タグ情報 ユーザが自由に使用できる文字列。地点名称などの識別情報を設定できます。

\* R：リード、W：ライト、R/W：リード&ライトを表しています。

#### （４）計算用道路データ情報構造体 NWinfo

計算に使用した計算用道路データ情報が設定されます。

メンバ	データ型	アクセス	説 明
Folder	string	W	未使用
Note	string	W	計算用道路データの説明
IsDetailExist	boolean	W	詳細ルート出力可能フラグ
IsTollExist	boolean	W	通行料金出力可能フラグ
NWID	guid	W	計算用道路データを識別するためのネットワーク ID
NWID_WithoutSpeed	guid	W	速度情報を無視した計算用道路データを識別するためのネットワーク ID 番号

\* R：リード、W：ライト、R/W：リード&ライトを表しています。



## 6. 6. 6 住所検索サービス

**InvokeService(GeoCommand As GeoCommand) As boolean**

※ このサービスは互換性のために残されています。今後新たにプログラムを作成する場合は使用しないでください。

住所検索サービスでは次の 5 処理を行うことができます。

各処理に応じたコマンド文字列を **GeoCommand** 住所検索コマンド構造体に設定し、住所検索を行います。検索結果は **GeoCommand** 住所検索コマンド構造体の各メンバ構造体に設定されます。

処 理	コマンド文字列	解 説
住所検索機能	SEARCHADDRESS	ジオアイテム構造体配列に設定されている漢字住所文字列を検索し、経緯度を算出します。
住所コード検索機能	SEARCHADDRESS CODE	ジオアイテム構造体配列に設定されている住所コードを検索し、経緯度を算出します。
下位住所列举機能	ENUMLOWERADD RESSBYCODE	パラメータ構造体に設定されている住所コードに該当する住所の下位住所を検索し、ジオアイテム構造体配列に列举します。
郵便番号検索機能	SEARCHZIP	ジオアイテム構造体配列に設定されている郵便番号文字列を検索し、経緯度を算出します。
郵便番号対応住所列举機能	ENUMADDRESSBY ZIP	パラメータ構造体に設定されている郵便番号文字列を検索し、該当する住所を、ジオアイテム構造体配列に列举します。

**エンドポイント URL**

住所検索サーバ取得サービスで取得した住所検索サーバの URL

**パラメータ                      解説**

GeoCommand                      住所検索コマンド構造体

**戻り値**

正常に計算できた場合 **True**、計算出来なかった場合 **False**。

## 備考

本サービスを呼び出す際には、GeoCommand 住所検索コマンド構造体の各メンバにデータを設定してください。

## ○ 住所検索コマンド構造体 GeoCommand

住所検索に必要な全ての情報を設定、または取得します。

項番	メンバ	データ型	説 明
(1)	Command	string	住所検索コマンド文字列
(2)	ResultMessage	string	住所検索結果文字列
(3)	Environ	Environ	環境設定構造体
(4)	Param	Param	パラメータ設定構造体
(5)	GeoItems	ArrayOfGeoItem	ジオアイテム構造体配列
(6)	GeoInfo	GeoInfo	住所データ情報構造体

## (1) 住所検索コマンド文字列 Command

住所検索サーバへどのような処理要求を行うかを設定します。各処理に応じたコマンド文字列を設定してください。

処 理	コマンド文字列
住所検索処理	SEARCHADDRESS
住所コード検索処理	SEARCHADDRESSCODE
下位住所列举処理	ENUMLOWERADDRESSBYCODE
郵便番号検索処理	SEARCHZIP
郵便番号対応住所列举処理	ENUMADDRESSBYZIP

## (2) 住所検索結果文字列 ResultMessage

住所検索を行い、検索結果に何らかの問題が発生した場合に、その状況が文字列として設定されます。

## (3) 環境設定構造体 Environ

ユーザ認証に必要なユーザ ID、パスワードや住所検索リスナの IP アドレス、ポート番号を設定します。ユーザ ID およびパスワードは、弊社からの「サービス開始通知書」に記載されている文字列を設定してください。住所検索リスナの IP アドレス、ポート番号は、住所検索サーバ取得サービスで得られた情報を設定してください。

メンバ	データ型	設定	説 明
UserID	string	必要	ユーザ ID
UUID	string	必要	パスワード
HWUID	string		(空文字列)
WSURL	string		(空文字列) (注)
Host	string	必要	住所検索リスナの IP アドレス
Port	int	必要	住所検索リスナのポート番号

(注) 住所検索サーバ取得サービスで、住所検索サーバのエンドポイント URL が得られます。住所検索サーバへアクセスする際に、エンドポイント URL を使用していますので、環境設定構造体の WSURL メンバにエンドポイント URL を設定する必要はありません。

## (4) パラメータ構造体 Param

住所検索を行う際に必要なパラメータを設定します。また、後述するパラメータ構造体アクセス表の”R/W”の箇所を各距離計算コマンドごとに設定する必要があります。

メンバ	データ型	説 明
Datum	int	経緯度の座標系識別子(現状では 0 を指定してください)
CoordinateFormat	int	経緯度の座標フォーマット識別子(現状では 0 を指定してください)
AddressLevel	short	住所レベル (住所検索処理で、検索したい住所レベルを指定します。1:都道府県、2:市区町村、8:大字・丁目、16:街区、32:号)
SearchEnumString	String	列挙検索文字列 (下位住所列挙処理で、検索する住所コードを指定します。NULL 文字を設定すると都道府県を列挙します。2 桁都道府県コードを設定すると市区町村を列挙します。5 桁市区町村コードを指定すると大字・丁目を列挙します。11 桁大字・丁目コードを指定すると街区を列挙します。16 桁街区コードを指定すると号を列挙します。 郵便番号対応住所列挙処理で、検索する郵便番号を指定します。)
SearchFlag	String	検索フラグ (郵便番号検索処理での検索フラグを指定します。1:先頭データ。郵便番号に該当する住所データの先頭データを検索する。8:中心位置。郵便番号に該当する住所の平均経緯度位置を検索する。16:中心住所。郵便番号に該当する住所の平均経緯度が一番近い住所データを検索する。)

各住所検索コマンドでのパラメータ構造体メンバへのアクセスは次のとおりです。

＜パラメータ構造体アクセス表＞

メンバ	住所 検索 処理	住所 コード 検索 処理	下位住所 列挙 処理	郵便 番号 検索 処理	郵便 番号 対応住所 列挙 処理	備考
Datum						(未使用)
CoordinateFormat						(未使用)
AddressLevel	R					
SearchEnumString			R		R	
SearchFlag				R		

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (5) ジオアイテム構造体配列 GeoItems

住所検索処理・住所コード検索処理・郵便番号検索処理では、検索する文字列を設定します。また、全ての処理において検索した計算結果を取得します。後述するジオアイテム構造体アクセス表の”R/W”の箇所を各距離計算コマンドごとに設定する必要があります。

＜ジオアイテム構造体メンバ表＞

メンバ	データ型	説 明
SearchString	string	検索文字列
ResultLevel	int	住所レベル (検索結果の当該住所の住所レベル 1:都道府県、2:市区町村、8:大字・丁目、16:街区、32:号)
ResultCode	string	住所コード
ResultAddress	string	解析住所文字列 (住所検索において、解析できた住所部分が設定される)
ResultRestAddress	string	非解析住所文字列 (住所検索において、解析出来なかった住所部分が設定される)
ResultZip	string	郵便番号
Lon	string	経度
Lat	string	緯度
Tag	string	タグ情報 (ユーザが自由に使用できる文字列。メッシュ番号や届先名称などロケーションを識別するための情報を設定します。)

各住所検索コマンドでのジオアイテム構造体メンバへのアクセスは次のとおりです。

＜ジオアイテム構造体アクセス表＞

メンバ	住所 検索 処理	住所 コード 検索 処理	下位住所 列挙 処理	郵便 番号 検索 処理	郵便 番号 対応住所 列挙 処理	備考
SearchString	R	R	W	R	W	
ResultLevel	W	W	W	W	W	郵便番号検索では、検索に成功した場合 1、失敗した場合 0 が設定されます。
ResultCode	W	W	W	W	W	検索した住所レベル、検索フラグによっては出力されない場合があります。
ResultAddress	W	W	W	W	W	検索した住所レベル、検索フラグによっては出力されない場合があります。
ResultRestAddress	W					
ResultZip		W	W	W	W	検索した住所レベル、検索フラグによっては出力されない場合があります。
Lon	W	W	W	W	W	
Lat	W	W	W	W	W	
Tag						

\* R : リード、W : ライト、R/W : リード&ライトを表しています。

## (6) 住所データ情報構造体 GeoInfo

検索に使用した住所データ情報が設定されます。

メンバ	データ型	説 明
Folder	string	住所データのフォルダ
Note	string	住所データの説明
GeoID	guid	住所データ ID

各距離計算コマンドでの住所データ情報構造体メンバへのアクセスは次のとおりです。

## ＜住所データ情報構造体アクセス表＞

メンバ	住所 検索 処理	住所 コード 検索 処理	下位住所 列挙 処理	郵便 番号 検索 処理	郵便 番号 対応住所 列挙 処理	備考
Folder	W	W	W	W	W	
Note	W	W	W	W	W	
GeoID	W	W	W	W	W	

\* R：リード、W：ライト、R/W：リード&ライトを表しています。



## 6. 7 結果コード一覧

## (1) 距離計算サービス

結果コード	説明
10001	不明なエラーが発生しました。
10002	データベースのオープンに失敗しました。
10003	本サービスの実行権限がありません。
10004	ユーザ認証が許可されませんでした。
10005	計算制限パラメータの取得に失敗しました。
10006	サービスコードが存在しません。
10007	計算回数が最大計算回数に達しています。試用会員などの場合、計算回数が制限されている場合があります。
10008	住所検索回数が最大住所検索回数に達しています。試用会員などの場合、住所検索回数が制限されている場合があります。
10009	前回の処理終了から計算間隔時間が経過していません。
10010	現在処理中です。処理中に次の処理は出来ません。
10011	計算開始時刻、計算終了時刻の取得に失敗しました。
10012	本サーバの使用は許可されていません。
10013	計算回数の更新に失敗しました。
10014	住所検索回数の更新に失敗しました。
10015	ユーザ認証中に例外が発生しました。
10016	コマンドを距離計算リスナ用に変換中に例外が発生しました。
10017	リスナへの接続に失敗しました。
10018	リスナへのコマンド送信中に例外が発生しました。
10019	リスナから返信受信中にタイムアウトしました。
10020	リスナから返信受信中に例外が発生しました。
10021	リスナからの受信データをデシリアライズ中に例外が発生しました。
10022	リスナからの受信データをサービス用に変換中に例外が発生しました。
10023	地点数がルート計算の最大地点数を超過しています。
10024	地点数が最短ルート計算の最大地点数を超過しています。
10025	範囲時間が最大到達／流入時間を超過しています。
10026	範囲距離が最大到達／流入距離を超過しています。
10027	地点数が片道一括計算の最大地点数を超過しています。
10028	ポリゴンレベルに不正な値が設定されています。
10029	ドーナツポリゴンレベルに不正な値が設定されています。

次ページに続く

前ページからの続き

結果コード	説明
10030	道路速度変更が許可されていません。
10031	道路速度変更可否チェック中に例外が発生しました。
10032	計算終了処理中に例外が発生しました。
10033	直線距離計算中に例外が発生しました。
10034	直線距離計算に失敗しました。
10035	ロケーション数が2未満です。
11000	距離計算リスナで、不明なエラーが発生しました。
11001	距離計算リスナで、デシリアライズ中に例外が発生しました。
11002	距離計算リスナで、データ受信中にタイムアウトしました。
11003	距離計算リスナで、データ受信中に例外が発生しました。
11004	距離計算リスナで、距離計算子オブジェクトの生成に失敗しました。
11005	距離計算リスナで、距離計算親オブジェクトへのアタッチに失敗しました。
11006	距離計算リスナで、地点が設定されていません。
11007	距離計算リスナで、設定された地点の数が多すぎます。
11008	距離計算リスナで、中心地点のノードが見つかりませんでした。
11009	距離計算リスナで、中心地点のノードが計算用道路データに存在しません。
11010	距離計算リスナで、到達圏／流入圏計算が失敗しました。（DLL レベル）
11011	距離計算リスナで、ポリゴン作成が失敗しました。（DLL レベル）
11012	距離計算リスナで、地点を2点以上設定する必要があります。
11013	距離計算リスナで、地点を3点以上設定する必要があります。
11014	距離計算リスナで、地点のノードが見つかりませんでした。
11015	距離計算リスナで、地点のノードが計算用道路データに存在しません。
11016	距離計算リスナで、最短ルート計算が失敗しました。到達できない地点が存在します。（DLL レベル）
11017	距離計算リスナで、最短ルート計算後のルート計算に失敗しました。
11018	距離計算リスナで、ポリゴン分割に失敗しました。（DLL レベル）
11019	距離計算リスナで、近傍リンク取得に失敗しました。（DLL レベル）
11020	距離計算リスナで、道路拡張情報取得に失敗しました。（DLL レベル）
11021	距離計算リスナで、リンク形状取得に失敗しました。（DLL レベル）
11022	距離計算リスナで、分割ポリゴン作成用バッファの確保に失敗しました。しばらく待ってから再実行してください。
11023	距離計算リスナで、リンクが設定されていませんでした。

次ページに続く

前ページからの続き

結果コード	説明
11024	距離計算リスナで、リンクに無効な文字列が設定されています。
11025	距離計算リスナで、リンクが見つかりませんでした。
11026	距離計算リスナで、リンクの速度設定に失敗しました。(DLL レベル)
11027	距離計算リスナで、高速道路区間の取得に失敗しました。(DLL レベル)
11028	距離計算リスナで、通行料金の取得に失敗しました。(DLL レベル)
11029	距離計算リスナで、計算用道路データに通行料金データが見つかりませんでした。
11030	距離計算リスナで、指定範囲リンク取得に失敗しました。(指定範囲オーバー)
11031	距離計算リスナで、近傍ノード取得に失敗したロケーションが存在します。
11032	距離計算リスナで、計算用道路データの取得に失敗しました。
11033	距離計算リスナで、地点を 1 点以上設定する必要があります。
11034	距離計算リスナで、経緯度の変換に失敗しました。
11035	距離計算リスナで、経緯度の復元に失敗しました。
11036	距離計算リスナで、計算処理中に例外が発生しました。
12000	距離計算リスナで、ルート計算に失敗しました。
12001～ 12998	距離計算リスナで、ルート計算の第 $x$ 区間の距離計算に失敗しました。
12999	距離計算リスナで、ルート計算の第 999 区間以降の距離計算に失敗しました。

## (2) 住所検索サービス

結果コード	説明
30001	不明なエラーが発生しました。
30002	データベースのオープンに失敗しました。
30003	本サービスの実行権限がありません。
30004	ユーザ認証が許可されませんでした。
30005	計算制限パラメータの取得に失敗しました。
30006	サービスコードが存在しません。
30007	計算回数が最大計算回数に達しています。試用会員などの場合、計算回数が制限されている場合があります。
30008	住所検索回数が最大住所検索回数に達しています。試用会員などの場合、住所検索回数が制限されている場合があります。
30009	前回の処理終了から計算間隔時間が経過していません。
30010	現在処理中です。処理中に次の処理は出来ません。
30011	計算開始時刻、計算終了時刻の取得に失敗しました。
30012	本サーバの使用は許可されていません。
30013	計算回数の更新に失敗しました。
30014	住所検索回数の更新に失敗しました。
30015	ユーザ認証中に例外が発生しました。
30016	コマンドを距離計算リスナ用に変換中に例外が発生しました。
30017	リスナへの接続に失敗しました。
30018	リスナへのコマンド送信中に例外が発生しました。
30019	リスナから返信受信中にタイムアウトしました。
30020	リスナから返信受信中に例外が発生しました。
30021	リスナからの受信データをデシリアライズ中に例外が発生しました。
30022	リスナからの受信データをサービス用に変換中に例外が発生しました。
30023	指定ユーザ ID で住所検索は利用できません。
30024	アイテム数が最大アイテム数を超過しています。
30025	処理するアイテムが存在しません。

次ページに続く

前ページからの続き

結果コード	説明
31000	住所検索リスナで、不明なエラーが発生しました。
31001	住所検索リスナで、デシリアライズ中に例外が発生しました。
31002	住所検索リスナで、データ受信中にタイムアウトしました。
31003	住所検索リスナで、データ受信中に例外が発生しました。
31004	住所検索リスナで、住所検索オブジェクトの生成に失敗しました。
31005	住所検索リスナで、測地系変換機能が有効になりませんでした。
31006	住所検索リスナで、検索住所コードの長さが正しくありません。
31007	住所検索リスナで、計算処理中に例外が発生しました。
31008	住所検索リスナで、未定義の命令を受信しました。
31009	住所検索リスナで、経緯度が正しい形式で入力されていません。

## 6. 8 Visual Studio.NET での定数定義 DLL の利用

開発環境に Visual Studio.NET を用いる場合、**ACT** 距離計算サービスの定数を定義したアセンブリを利用することができます。アセンブリファイルは、DCWSCnst.dll という名前です。各定数は、名前空間 Act.Dcws 内の列挙型およびクラスとして定義されています。

列挙型・クラス	定数	対応パラメータ
Command	CalcRoute CalcOptRoute CalcArea GetNearestNode EnumNearNode GetNetworkInfo	距離計算コマンド文字列
Datum	Tokyo (0:日本測地系) Jgd2000 (1:世界測地系)	座標系識別子
CalcKind	Time (1:時間最短) Dist (2:距離最短)	計算種別
UseHighway	Use (0:高速道路を使用する) NotUse (1:高速道路を使用しない) PayRoadFerry (2:北海道~青森、鹿児島~沖縄間でフェリーを使用する) Ferry (4:フェリーを使用する) Railroad (8:鉄道を使用する) Airway (16:航空機を使用する) Ship (64:内航船を使用する)	高速道路、輸送手段 (旧サービス用)
Transport	None (0:高速道路を使用しない) Highway (1:高速道路を使用する) PayRoadFerry (2:北海道~青森、鹿児島~沖縄間でフェリーを使用する) Ferry (4:フェリーを使用する) Railroad (8:鉄道を使用する) Airway (16:航空機を使用する) Ship (64:内航船を使用する) LimitedExpressTrain (64:旅客鉄道を使用する場合、特急を使用する)	使用交通機関
AreaKind	Time (1:時間圏計算) Dist (2:距離圏計算)	到達圏／流入圏計算種別
RouteKind	Simple (0:簡易ルート) Detail (1:詳細ルート)	ルート種別指定
AreaDirection	OutFlow (0:到達圏計算) InFlow (4096:流入圏計算)	範囲方向指定
HighwayOutput	NotOutput (0:出力しない) Output (1:出力する)	高速道路区間出力フラグ
TollOutput	NotOutput (0:出力しない) Output (1:出力する)	通行料金出力フラグ

(次ページへ続く)

列挙型・クラス	定数	対応パラメータ
LocsOutput	NotOutput (0:出力しない) Output (1:出力する)	ロケーション構造体配列出力フラグ
PntsOutput	NotOutput (0:出力しない) Output (1:出力する)	ポイント構造体配列出力フラグ
RnsOutput	NotOutput (0:出力しない) Output (1:出力する)	経由道路名構造体配列出力フラグ
TrdsOutput	NotOutput (0:出力しない) Output (1:出力する)	経由有料道路構造体配列出力フラグ
PolygonKind	SinglePolygon (1:単一) MultiPolygon (2:複数)	描画ポリゴン種別
PolygonDetail	None (0:なし) ExpandMiddle (1:中間リンク補間) ExpandTerminal (2:末端リンク補間)	ポリゴン詳細フラグ
SpeedFlag	None (0:なし) TemporarySpeed (16:メモリ上速度) PermanentSpeed (32:ファイル上速度) ZeroReserve (64:速度 0 の方向は変更しない)	速度フラグ
NwidClassA	Car (1:自動車) Airway (2:航空機) Ferry (4:フェリー) Railroad (8:旅客鉄道) RailroadFreight (16:貨物鉄道) Ship (64:内航船) Walk (256:歩行者)	計算用道路データネットワーク ID 対応輸送手段
GeoCommandStr	SearchAddress SearchAddressCode EnumLowerAddressByCode SearchZip EnumAddressByZip GetGeoInfo	住所検索コマンド文字列
AddressLevel	None (0:なし) Pref (1:都道府県) City (2:市区町村) Street (8:大字・丁目) Block (16:街区) House (32:号)	住所レベル
AddressCodeLength	Pref (2:都道府県) City (5:市区町村) Street (11:大字・丁目) Block (16:街区) House (32:号)	住所コード長
SearchAddressFlag	None (0:なし) GetFullAddressss (1:全住所取得)	住所検索用検索フラグ
SearchZipFlag	None (0:なし) SelectTop (1:先頭データ) SelectCenter (8:中心位置) SelectCenterAddress (16:中心住所)	郵便番号検索用検索フラグ

## 6. 9 サンプルコード

Visual Basic .NET でのサンプルコードを示します。

6. 8 で解説した DCWSCnst.dll をプロジェクトで参照してください。

### (1) ルート計算

```
' 東京都庁から大阪府庁までのルート計算を行うサンプル
Dim adminService As New DCAdmin.ACTDCADMIN ' 管理サービス
Dim adminEnviron As New DCAdmin.Environ ' 管理サービス用環境設定
Dim nwInfos() As DCAdmin.NWInfo ' 計算用道路データ情報配列
Dim dcService As New DCService.ACTDCWS ' 距離計算サービス
Dim dcEnviron As New DCService.Environ ' 距離計算サービス用環境設定
Dim crParam As New DCService.CRParam ' ルート計算用パラメータ構造体
Dim locs() As DCService.Loc ' ロケーション構造体配列
Dim pnts() As DCService.Pnt ' 計算結果を受け取るポイント配列
Dim rns() As DCService.Rn ' 経由道路名構造体配列
Dim trds() As DCService.Trd ' 経由有料道路構造体配列
Dim nwInfo As DCService.NWInfo ' 計算結果を受け取る計算用道路データ情報

' ユーザID、パスワード設定
AdminEnviron.UserID = "UserID"
AdminEnviron.UUID = "Password"

' 計算用道路データ情報取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableNWInfos(AdminEnviron, NWInfos)
MsgBox("計算用道路データ数:" & NWInfos.Length.ToString)

' 距離計算サーバ取得サービス呼び出し
adminService.GetUsableDCServer(AdminEnviron, NWInfos(0)) ' 最初の計算用道路データを指定
MsgBox("距離計算サーバ エンドポイントURL:" & vbLf & AdminEnviron.WSURL & vbLf & _
      "IP:" & AdminEnviron.Host & vbLf & "Port:" & AdminEnviron.Port.ToString)

' 環境設定をコピー
dcEnviron.UserID = adminEnviron.UserID
dcEnviron.UUID = adminEnviron.UUID
dcEnviron.Host = adminEnviron.Host
dcEnviron.Port = adminEnviron.Port

' パラメータ（時間最短、高速使用する、簡易ルート出力）を設定
crParam.CalcKind = Act.Dcws.CalcKind.Time
crParam.Transport = Act.Dcws.UseHighway.Use
crParam.PntsOutput = Act.Dcws.PntsOutput.Output
crParam.RouteKind = Act.Dcws.RouteKind.Simple

' ロケーションを設定
locs = New DCService.Loc(1) {}
locs(0) = New DCService.Loc
locs(0).Lon = "139.695000"
locs(0).Lat = "35.686389"
locs(0).Tag = "東京都庁"
locs(1) = New DCService.Loc
locs(1).Lon = "135.522500"
locs(1).Lat = "34.683056"
locs(1).Tag = "大阪府庁"

' 距離計算サービス呼び出し
DCService.Url = AdminEnviron.WSURL
dcService.CalcRoute(dcEnviron, crParam, locs, pnts, rns, trds, nwInfo)
MsgBox("所要時間(分):" & locs(1).Time & _
      "道のり(m):" & locs(1).Dist & _
      "走行ルートのポイント数:" & Pnts.Length.ToString)
```



## (2) 到達圏／流入圏計算

```

' 東京都庁までの流入圏計算とポリゴン取得を行うサンプル
Dim adminService As New DCAdmin.ACTDCADMIN ' 管理サービス
Dim adminEnviron As New DCAdmin.Environ ' 管理サービス用環境設定
Dim nwInfos() As DCAdmin.NWInfo ' 計算用道路データ情報配列
Dim dcService As New DCService.ACTDCWS ' 距離計算サービス
Dim dcEnviron As New DCService.Environ ' 距離計算サービス用環境設定
Dim ca3Param As New DCService.CA3Param ' 到達圏／流入圏計算用パラメータ
Dim pnts() As DCService.Pnt ' 計算結果を受け取るポイント配列
Dim nwInfo As DCService.NWInfo ' 計算結果を受け取る計算用道路データ情報

' ユーザID、パスワード設定
AdminEnviron.UserID = "UserID"
AdminEnviron.UUID = "Password"

' 計算用道路データ情報取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableNWInfos(adminEnviron, nwInfos)
MsgBox("計算用道路データ数:" & nwInfos.Length.ToString)

' 距離計算サーバ取得サービス呼び出し
adminService.GetUsableDCServer(adminEnviron, nwInfos(0)) ' 最初の計算用道路データを指定
MsgBox("距離計算サーバエンドポイントURL:" & vbCrLf & adminEnviron.WSURL & vbCrLf & _
      "IP:" & adminEnviron.Host & vbCrLf & "Port:" & adminEnviron.Port.ToString)

' 環境設定をコピー
dcEnviron.UserID = adminEnviron.UserID
dcEnviron.UUID = adminEnviron.UUID
dcEnviron.Host = adminEnviron.Host
dcEnviron.Port = adminEnviron.Port

' パラメータ（時間最短、高速使用する、流入圏計算、90分圏、時間圏、
' 単一ポリゴン、ポリゴンレベル3、計算の中心は東京都庁）を設定
ca3Param.CalcKind = Act.Dcws.CalcKind.Time
ca3Param.Transport = Act.Dcws.Transport.Highway
ca3Param.AreaDirection = Act.Dcws.AreaDirection.InFlow
ca3Param.AreaKind = Act.Dcws.AreaKind.Time
ca3Param.AreaRange = 90 * 60 * 10
ca3Param.PolygonKind = Act.Dcws.PolygonKind.SinglePolygon
ca3Param.PolygonLevel = 3
ca3Param.DonutPolygonLevel = -2
ca3Param.PolygonDetail = Act.Dcws.PolygonDetail.None
ca3Param.StartPnt = New DCService.NodePnt
ca3Param.StartPnt.Node = "0"
ca3Param.StartPnt.Lon = "139.695000"
ca3Param.StartPnt.Lat = "35.686389"

' 距離計算サービス呼び出し
dcService.Url = adminEnviron.WSURL
dcService.CalcArea3(dcEnviron, ca3Param, pnts, nwInfo)
MsgBox("流入圏ポリゴンのポイント数:" & pnts.Length.ToString)

```

## (3) 片道一括計算

```

' 神奈川、埼玉、千葉県庁から東京都庁までの片道一括計算を行うサンプル
Dim adminService As New DCAdmin.ACTDCADMIN ' 管理サービス
Dim adminEnviron As New DCAdmin.Environ ' 管理サービス用環境設定
Dim nwInfos() As DCAdmin.NWInfo ' 計算用道路データ情報配列
Dim dcService As New DCService.ACTDCWS ' 距離計算サービス
Dim dcEnviron As New DCService.Environ ' 距離計算サービス用環境設定
Dim cow3Param As New DCService.COW3Param ' 片道一括計算用パラメータ
Dim loc As DCService.Loc ' ロケーション
Dim locs() As DCService.Loc ' 計算結果を受け取るロケーション配列
Dim nwInfo As DCService.NWInfo ' 計算結果を受け取る計算用道路データ情報
Dim message As New StringBuilder() ' 結果メッセージ

' ユーザID、パスワード設定
AdminEnviron.UserID = "UserID"
AdminEnviron.UUID = "Password"

' 計算用道路データ情報取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableNWInfos(adminEnviron, nwInfos)
MsgBox("計算用道路データ数:" & nwInfos.Length.ToString)

' 距離計算サーバ取得サービス呼び出し
adminService.GetUsableDCServer(adminEnviron, nwInfos(0)) ' 最初の計算用道路データを指定
MsgBox("距離計算サーバエンドポイントURL:" & vbCrLf & adminEnviron.WSURL & vbCrLf & _
      "IP:" & adminEnviron.Host & vbCrLf & "Port:" & adminEnviron.Port.ToString)

' 環境設定をコピー
dcEnviron.UserID = adminEnviron.UserID
dcEnviron.UUID = adminEnviron.UUID
dcEnviron.Host = adminEnviron.Host
dcEnviron.Port = adminEnviron.Port

' パラメータ（時間最短、高速使用する、流入圏計算、
' 時間圏、90分圏、計算の中心は東京都庁）を設定
cow3Param.CalcKind = Act.Dcws.CalcKind.Time
cow3Param.Transport = Act.Dcws.Transport.Highway
cow3Param.AreaDirection = Act.Dcws.AreaDirection.InFlow
cow3Param.AreaKind = Act.Dcws.AreaKind.Time
cow3Param.AreaRange = 90 * 60 * 10
cow3Param.HighwayOutput = Act.Dcws.HighwayOutput.NotOutput
cow3Param.TollOutput = Act.Dcws.TollOutput.NotOutput
cow3Param.StartPnt = New DCService.NodePnt
cow3Param.StartPnt.Node = "0"
cow3Param.StartPnt.Lon = "139.695000"
cow3Param.StartPnt.Lat = "35.686389"

' ロケーションを設定
ReDim locs(2)
locs(0) = New DCService.Loc
locs(0).Lon = "139.645833"
locs(0).Lat = "35.444722"
locs(0).Tag = "神奈川県庁"
locs(1) = New DCService.Loc
locs(1).Lon = "139.652222"
locs(1).Lat = "35.854167"
locs(1).Tag = "埼玉県庁"
locs(2) = New DCService.Loc
locs(2).Lon = "140.126667"
locs(2).Lat = "35.601389"
locs(2).Tag = "千葉県庁"

' 距離計算サービス呼び出し
dcService.Url = adminEnviron.WSURL
dcService.CalcOneWay3(dcEnviron, cow3Param, locs, nwInfo)
For Each loc In locs
    message.AppendLine(loc.Tag & "から都庁まで 所要時間(分):" & loc.Time & _
      " 道のり(m):" & loc.Dist)
Next
MsgBox(message.ToString())

```

## (4) 住所検索

```

' ACT住所"東京都港区新橋3-7-4"を住所検索し経緯度を表示するサンプル
Dim adminService As New DCAdmin.ACTDCADMIN      ' 管理サービス
Dim adminEnviron As New DCAdmin.Environ          ' 管理サービス用環境設定
Dim gcService As New GCService.ACTGCWS          ' 住所検索サービス
Dim gcEnviron As New GCService.Environ          ' 住所検索サービス用環境設定
Dim param As New GCService.CommonParam          ' 共通パラメータ
Dim items(0) As GCService.SearchAddressItem     ' 住所検索アイテム配列
Dim geoInfo As GCService.GeoInfo                ' 住所データ情報

' ユーザID、パスワード設定
adminEnviron.UserID = "UserID"
adminEnviron.UUID = "Password"

' 住所検索サーバ取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableGeoServer(adminEnviron)
MsgBox("距離計算サーバエンドポイントURL:" & vbCrLf & adminEnviron.WSURL & vbCrLf & _
    "IP:" & adminEnviron.Host & vbCrLf & "Port:" & adminEnviron.Port.ToString)

' 環境設定をコピー
gcEnviron.UserID = adminEnviron.UserID
gcEnviron.UUID = adminEnviron.UUID
gcEnviron.Host = adminEnviron.Host
gcEnviron.Port = adminEnviron.Port

' 共通パラメータを設定 (日本測地系)
param.Datum = Act.Dcws.Datum.Tokyo
param.CoordinateFormat = 0
param.SearchFlag = 0

' 住所検索アイテムを設定
items(0) = New GCService.SearchAddressItem
items(0).SearchAddress = "東京都港区新橋3-7-4"

' 住所検索サービス呼び出し (街区レベルで検索)
gcService.Url = adminEnviron.WSURL
gcService.SearchAddress(gcEnviron, param, Act.Dcws.AddressLevel.Block, items, geoInfo)
MsgBox("検索住所:" & items(0).SearchAddress & vbCrLf & _
    "経度:" & items(0).Lon & vbCrLf & _
    "緯度:" & items(0).Lat)

```

## (5) 下位住所列举

```

' 東京都の市区町村を列举するサンプル
Dim adminService As New DCAdmin.ACTDCADMIN ' 管理サービス
Dim adminEnviron As New DCAdmin.Environ ' 管理サービス用環境設定
Dim gcService As New GCService.ACTGCWS ' 住所検索サービス
Dim gcEnviron As New GCService.Environ ' 住所検索サービス用環境設定
Dim param As New GCService.CommonParam ' 共通パラメータ
Dim items() As GCService.AddressItem ' 住所アイテム配列
Dim geoInfo As GCService.GeoInfo ' 住所データ情報
Dim message As New StringBuilder() ' 結果メッセージ

' ユーザID、パスワード設定
adminEnviron.UserID = "UserID"
adminEnviron.UUID = "Password"

' 住所検索サーバ取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableGeoServer(adminEnviron)
MsgBox("距離計算サーバエンドポイントURL:" & vbCrLf & adminEnviron.WSURL & vbCrLf & _
    "IP:" & adminEnviron.Host & vbCrLf & "Port:" & adminEnviron.Port.ToString)

' 環境設定をコピー
gcEnviron.UserID = adminEnviron.UserID
gcEnviron.UUID = adminEnviron.UUID
gcEnviron.Host = adminEnviron.Host
gcEnviron.Port = adminEnviron.Port

' 共通パラメータを設定 (日本測地系)
param.Datum = Act.Dcws.Datum.Tokyo
param.CoordinateFormat = 0
param.SearchFlag = 0

' 住所検索サービス呼び出し (東京都の住所コード=13)
gcService.Url = adminEnviron.WSURL
gcService.EnumLowerAddressByCode(gcEnviron, param, "13", items, geoInfo)
For Each item As GCService.AddressItem In items
    message.Append(item.Address & " 住所コード:" & item.AddressCode & " ")
Next
MsgBox(message.ToString())

```

## (6) 最寄住所取得

```

' 東京都庁の最寄住所を取得するサンプル
Dim adminService As New DCAdmin.ACTDCADMIN
Dim adminEnviron As New DCAdmin.Environ
Dim gcService As New GCService.ACTGCWS
Dim gcEnviron As New GCService.Environ
Dim param As New GCService.CommonParam
Dim items(0) As GCService.GetNearestAddressItem
Dim geoInfo As GCService.GeoInfo

' 管理サービス
' 管理サービス用環境設定
' 住所検索サービス
' 住所検索サービス用環境設定
' 共通パラメータ
' 最寄住所取得アイテム配列
' 住所データ情報

' ユーザID、パスワード設定
adminEnviron.UserID = "UserID"
adminEnviron.UUID = "Password"

' 住所検索サーバ取得サービス呼び出し
adminService.Url = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
adminService.GetUsableGeoServer(adminEnviron)
MsgBox("距離計算サーバエンドポイントURL:" & vbCrLf & adminEnviron.WSURL & vbCrLf & _
    "IP:" & adminEnviron.Host & vbCrLf & "Port:" & adminEnviron.Port.ToString)

' 環境設定をコピー
gcEnviron.UserID = adminEnviron.UserID
gcEnviron.UUID = adminEnviron.UUID
gcEnviron.Host = adminEnviron.Host
gcEnviron.Port = adminEnviron.Port

' 共通パラメータを設定 (日本測地系)
param.Datum = Act.Dcws.Datum.Tokyo
param.CoordinateFormat = 0
param.SearchFlag = 0

' 最寄住所取得アイテムを設定
items(0) = New GCService.GetNearestAddressItem
items(0).SearchLon = "139.695000"
items(0).SearchLat = "35.686389"

' 最寄住所取得サービス呼び出し
gcService.Url = adminEnviron.WSURL
gcService.GetNearestAddress(gcEnviron, param, items, geoInfo)
MsgBox(String.Format("検索経度:" & items(0).SearchLon & vbCrLf & _
    "検索緯度:" & items(0).SearchLat & vbCrLf & _
    "最寄住所:" & items(0).FullAddress))

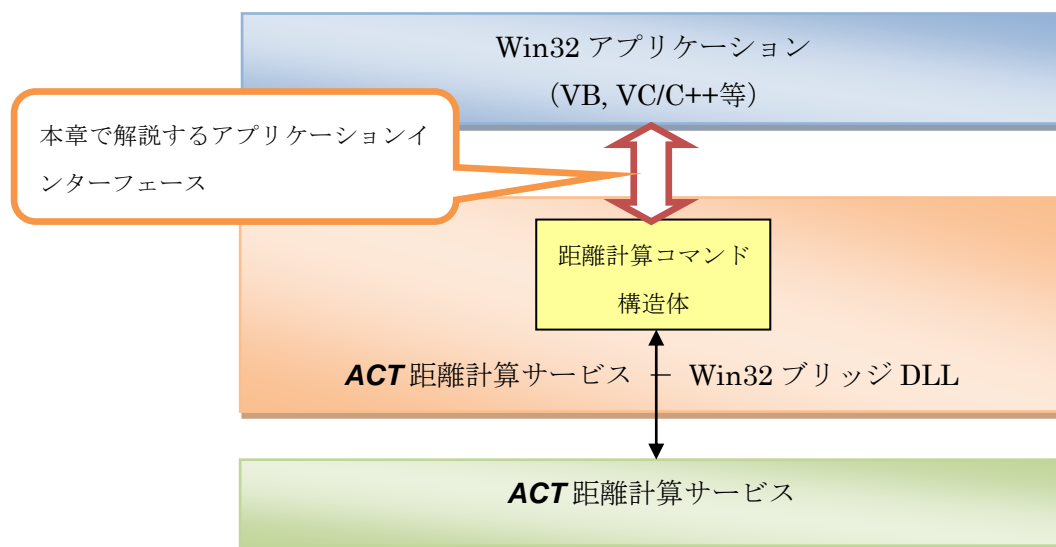
```

## 7. ブリッジ DLL を利用したプログラミング

「**ACT** 距離計算サービス – Win32 ブリッジ DLL」は、XML/Web サービスに対応していない開発環境で、**ACT** 距離計算サービスを呼び出すための Win32DLL です。本章では、「**ACT** 距離計算サービス – Win32 ブリッジ DLL」の仕様について解説します。サービスの呼び出し手順や各種構造体の仕様については「6. WSDL ファイルを利用したプログラミング」を参照してください。また、DLL やヘッダファイルを開発環境へ取り込む方法については、各開発環境のマニュアルを参照してください。

### 7. 1 概要

「**ACT** 距離計算サービス – Win32 ブリッジ DLL」（ファイル名”DWSBrdge.dll”）は、**ACT** 距離計算サービスと Win32 アプリケーションとの橋渡しを行います。また、「**ACT** 距離計算サービス – Win32 ブリッジ DLL」の内部に、距離計算コマンド構造体を保持しており、データの橋渡しは、距離計算コマンド構造体内の各構造体毎に行います。



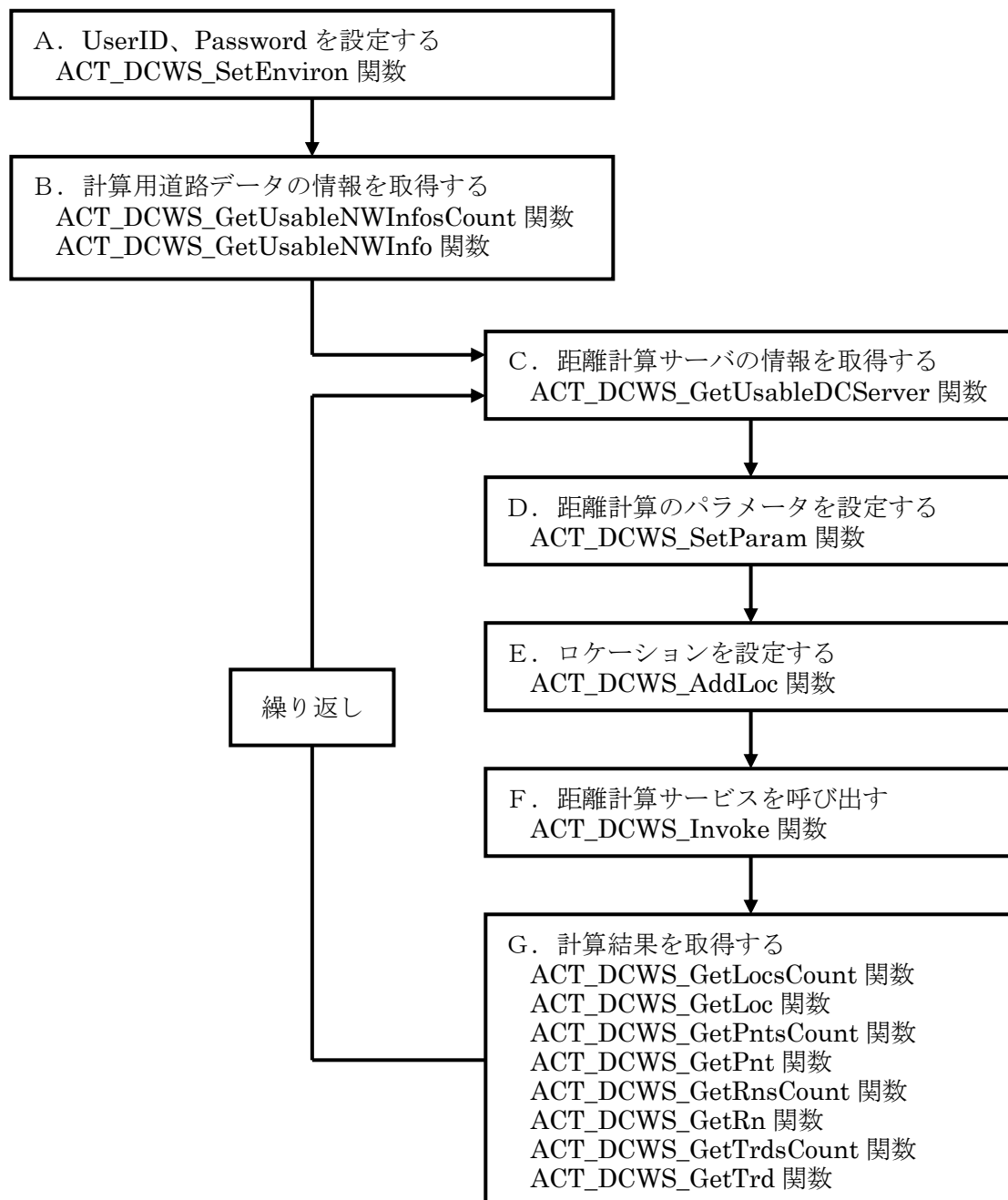
## 7. 2 呼び出しシーケンス

**ACT** 距離計算サービスを呼び出すシーケンスです。

全てのシーケンスで、予め ACT\_DCWS\_SetEnviron 関数を使用し、ユーザ I D とパスワードを設定しておく必要があります。

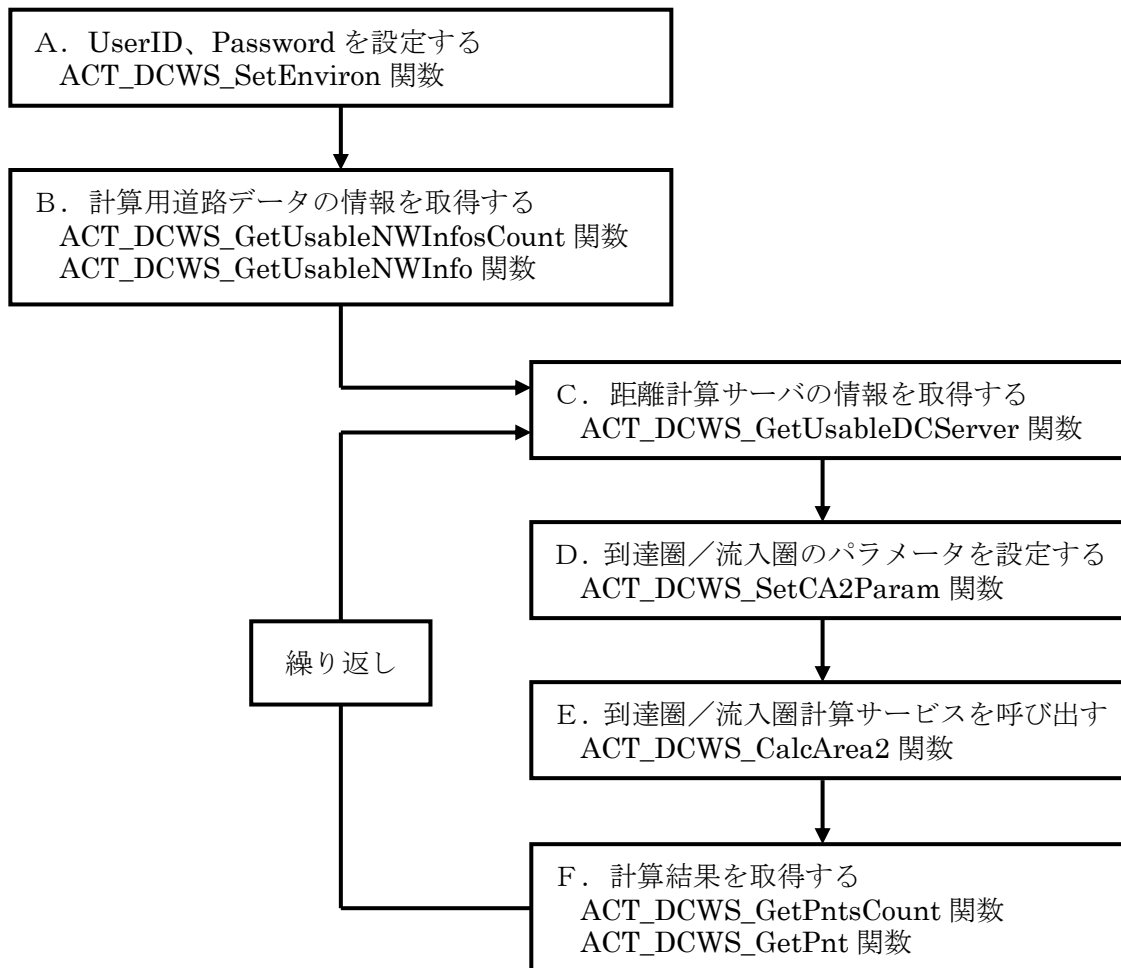
### (1) 距離計算を行う場合

距離計算を実行する基本的なシーケンスは次のとおりです。



## (2) 到達圏／流入圏計算を行う場合

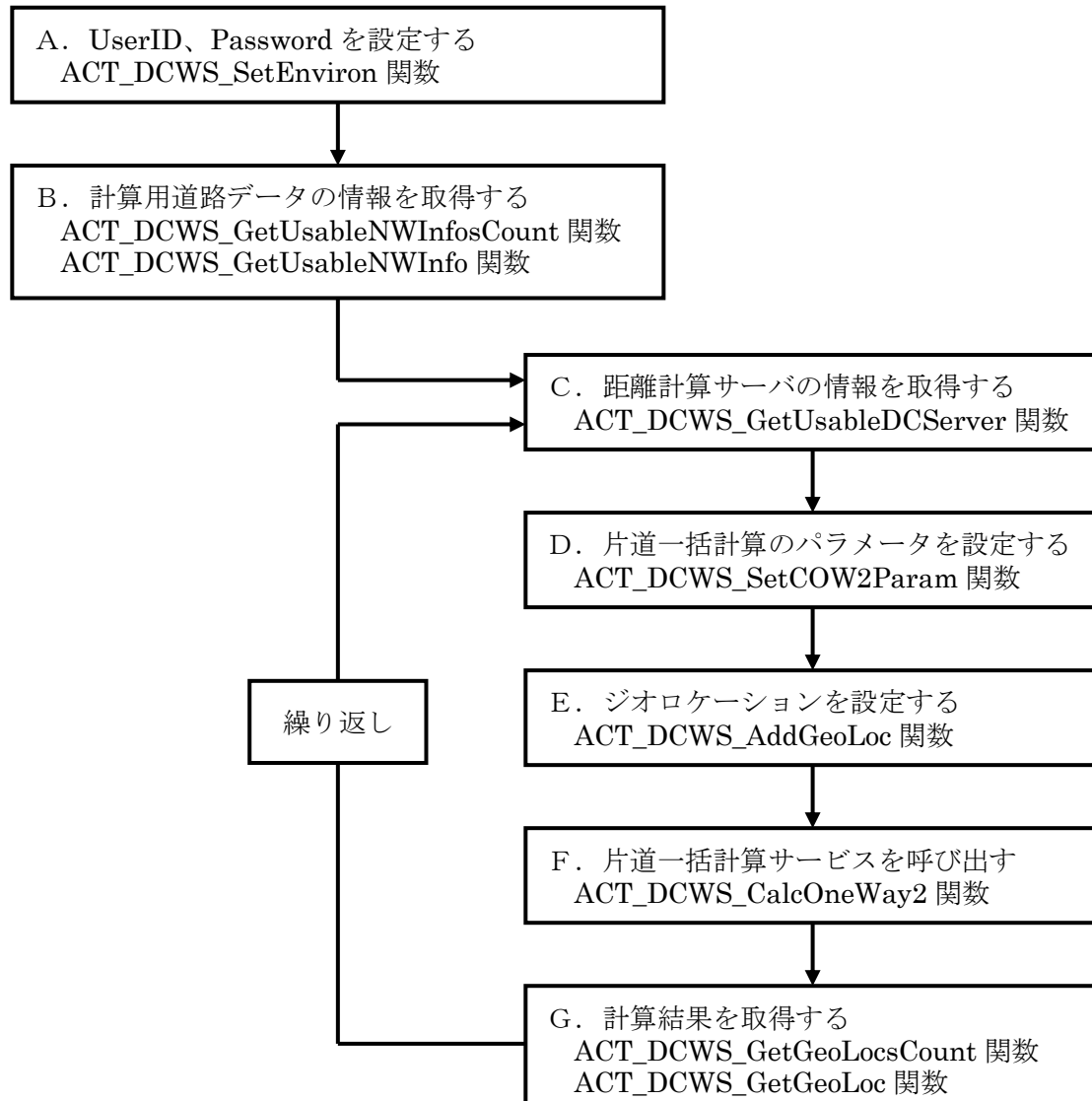
到達圏／流入圏計算を実行する基本的なシーケンスは次のとおりです。





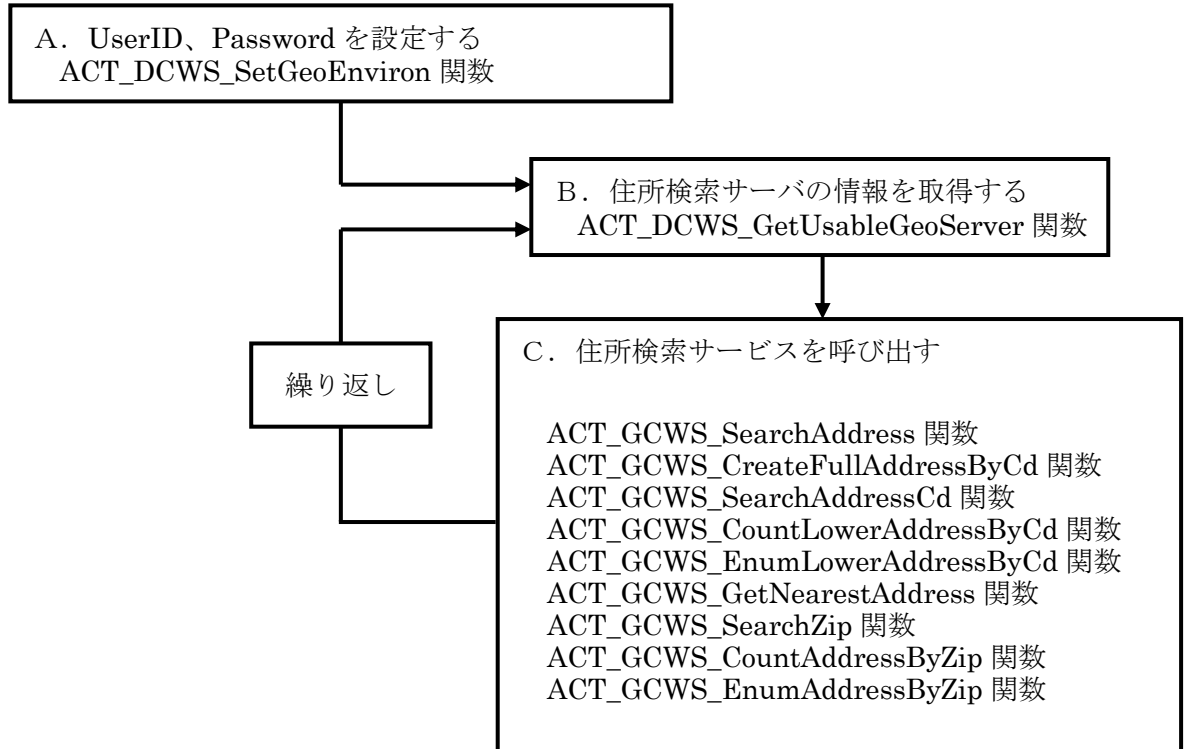
(3) 片道一括計算を行う場合

片道一括計算を実行する基本的なシーケンスは次のとおりです。



## (4) 住所検索を行う場合

住所検索を実行する基本的なシーケンスは次のとおりです。



## 7. 3 関数一覧

「**ACT** 距離計算サービス – Win32 ブリッジ DLL」には、3 種類の関数群が存在します。

項番	種類	解説
(1)	サービス呼び出し関数	<b>ACT</b> 距離計算サービスの各サーバのサービス呼び出しを行う関数
(2)	データ取得／設定関数	距離計算コマンド構造体等のデータの取得／設定を行う関数
(3)	その他の関数	バージョン情報やエラー情報を取得する関数

## (1) サービス呼び出し関数

サーバ	関数名	解 説
管理サーバ	<a href="#">ACT_DCWS_GetUsableNWInfosCount</a>	計算用道路データ情報取得サービス呼び出し、計算用道路データ情報数を取得します
	<a href="#">ACT_DCWS_GetUsableNWInfo</a>	計算用道路データ情報を取得します
	<a href="#">ACT_DCWS_GetUsableDCServer</a>	距離計算サーバ取得サービス呼び出し、距離計算サーバの情報を取得します
	<a href="#">ACT_DCWS_GetUserInfo</a>	ユーザ情報を取得します
距離計算サーバ	<a href="#">ACT_DCWS_Invoke</a>	距離計算サービスを呼び出し、距離計算を実行します
	<a href="#">ACT_DCWS_CalcArea2</a>	到達圏/流入圏計算を実行します
	<a href="#">ACT_DCWS_CalcOneWay2</a>	片道一括計算を実行します
住所検索サーバ	<a href="#">ACT_DCWS_GetUsableGeoServer</a>	住所検索サーバ取得サービスを呼び出し、住所検索サーバの情報を取得します
	<a href="#">ACT_DCWS_GeoInvoke</a>	住所検索サービスを呼び出し、住所検索を実行します (このサービスは互換性のため残されています)
	<a href="#">ACT_GCWS_SearchAddress</a>	住所検索サービスを呼び出し、住所検索を実行します
	<a href="#">ACT_GCWS_CreateFullAddressByCd</a>	住所検索サービスを呼び出し、全住所取得を実行します
	<a href="#">ACT_GCWS_SearchAddressCd</a>	住所検索サービスを呼び出し、住所コード検索を実行します
	<a href="#">ACT_GCWS_CountLowerAddressesByCd</a>	住所検索サービスを呼び出し、下位住所数取得を実行します

	<a href="#"><u>ACT_GCWS_EnumLowerAddressesByCd</u></a>	住所検索サービスを呼び出し、下位住所 列挙を実行します
	<a href="#"><u>ACT_GCWS_GetNearestAddress</u></a>	住所検索サービスを呼び出し、最寄住所 取得を実行します
	<a href="#"><u>ACT_GCWS_SearchZip</u></a>	住所検索サービスを呼び出し、郵便番号 検索を実行します
	<a href="#"><u>ACT_GCWS_CountAddressByZip</u></a>	住所検索サービスを呼び出し、郵便番号 対応住所数取得を実行します
	<a href="#"><u>ACT_GCWS_EnumAddressByZip</u></a>	住所検索サービスを呼び出し、郵便番号 対応住所列挙を実行します

## (2) データ取得／設定関数

関数名	解 説
<a href="#"><u>ACT_DCWS_GetEnviron</u></a>	環境設定構造体を取得します
<a href="#"><u>ACT_DCWS_SetEnviron</u></a>	環境設定構造体を設定します
<a href="#"><u>ACT_DCWS_GetParam</u></a>	パラメータ構造体を取得します
<a href="#"><u>ACT_DCWS_SetParam</u></a>	パラメータ構造体を設定します
<a href="#"><u>ACT_DCWS_GetLocsCount</u></a>	ロケーション構造体配列の要素数を取得します
<a href="#"><u>ACT_DCWS_ClearLocs</u></a>	ロケーション構造体配列をクリアし、要素数を 0 にします
<a href="#"><u>ACT_DCWS_GetLoc</u></a>	ロケーション構造体を取得します
<a href="#"><u>ACT_DCWS_AddLoc</u></a>	ロケーション構造体を追加します
<a href="#"><u>ACT_DCWS_GetPntsCount</u></a>	ポイント構造体配列の要素数を取得します
<a href="#"><u>ACT_DCWS_GetPnt</u></a>	ポイント構造体を取得します
<a href="#"><u>ACT_DCWS_GetRnsCount</u></a>	経由道路名構造体配列の要素数を取得します
<a href="#"><u>ACT_DCWS_GetRn</u></a>	経由道路名構造体を取得します
<a href="#"><u>ACT_DCWS_GetTrdsCount</u></a>	経由有料道路構造体配列の要素数を取得します
<a href="#"><u>ACT_DCWS_GetTrd</u></a>	経由有料道路構造体を取得します
<a href="#"><u>ACT_DCWS_GetNWInfo</u></a>	計算用道路データ情報構造体を取得します
<a href="#"><u>ACT_DCWS_GetGeoEnviron</u></a>	住所検索用環境設定構造体を取得します
<a href="#"><u>ACT_DCWS_SetGeoEnviron</u></a>	住所検索用環境設定構造体を設定します
<a href="#"><u>ACT_DCWS_GetGeoParam</u></a>	住所検索用パラメータ構造体を取得します
<a href="#"><u>ACT_DCWS_SetGeoParam</u></a>	住所検索用パラメータ構造体を設定します
<a href="#"><u>ACT_DCWS_GetGeoItemsCount</u></a>	ジオアイテム構造体配列の要素数を取得します
<a href="#"><u>ACT_DCWS_ClearGeoItems</u></a>	ジオアイテム構造体配列をクリアし、要素数を 0 にします
<a href="#"><u>ACT_DCWS_GetGeoItem</u></a>	ジオアイテム構造体を取得します
<a href="#"><u>ACT_DCWS_AddGeoItem</u></a>	ジオアイテム構造体を追加します
<a href="#"><u>ACT_DCWS_GetGeoInfo</u></a>	住所データ情報構造体を取得します

(次ページへ続く)

(前ページから継続)

関数名	解 説
<a href="#">ACT_DCWS_GetCA2Param</a>	到達圏/流入圏計算用パラメータ構造体を取得します
<a href="#">ACT_DCWS_SetCA2Param</a>	到達圏/流入圏計算用パラメータ構造体を設定します
<a href="#">ACT_DCWS_GetCOW2Param</a>	片道一括計算用パラメータ構造体を取得します
<a href="#">ACT_DCWS_SetCOW2Param</a>	片道一括計算用パラメータ構造体を設定します
<a href="#">ACT_DCWS_GetGeoLocsCount</a>	ジオロケーション構造体配列の要素数を取得します
<a href="#">ACT_DCWS_ClearGeoLocs</a>	ジオロケーション構造体配列をクリアし、要素数を 0 にします
<a href="#">ACT_DCWS_GetGeoLoc</a>	ジオロケーション構造体を取得します
<a href="#">ACT_DCWS_AddGeoLoc</a>	ジオロケーション構造体を追加します

(3) その他の関数

関数名	解 説
<a href="#">ACT_DCWS_GetLibVersion</a>	バージョン番号取得します
<a href="#">ACT_DCWS_LastErrorCode</a>	エラーコードを取得します
<a href="#">ACT_DCWS_LastErrorMessage</a>	エラーメッセージを取得します
<a href="#">ACT_DCWS_SetProxy</a>	プロキシ情報を設定します
<a href="#">ACT_DCWS_SetProxyInfoA</a>	プロキシ情報を設定します(マルチバイト文字)
<a href="#">ACT_DCWS_SetProxyInfoW</a>	プロキシ情報を設定します(ワイド文字)
<a href="#">ACT_DCWS_GetDCWSMessage</a>	距離計算サーバ側で発生するエラーメッセージを取得します
<a href="#">ACT_DCWS_GetGCWSMessage</a>	住所検索サーバ側で発生するエラーメッセージを取得します

## 7. 4 関数仕様

### (1) サービス呼び出し関数

#### **int WINAPI ACT\_DCWS\_GetUsableNWInfosCount(void)**

管理サーバに対し計算用道路データ情報取得サービスを読み出し、計算用道路データ情報数を取  
得します。

パラメータ	解説
-------	----

---

なし

#### 戻り値

計算用道路データ情報数。異常終了時は-1。

#### 備考

計算用道路データ情報取得サービスを読み出し、計算用道路データ情報数を取得します。

本関数に限らず、サービス呼び出し関数を読み出す前には予め ACT\_DCWS\_SetEnviron 関  
数を使用し、ユーザ I D、パスワード、距離計算サービス各サーバの URL を設定しておく  
必要があります。

本関数に限らず、異常終了時は ACT\_DCWS\_LastErrorCode 関数を使用して、エラー  
コードを取得します。エラーコードが 10,000 未満の場合は 7. 8 エラーコード一覧の  
エラーです。ACT\_DCWS\_LastErrorMessage 関数を使用して、エラー文字列を取得します。  
エラーコードが 10,000 以上の場合は距離計算サーバのエラーです。

ACT\_DCWS\_GetDCWSMessage、ACT\_GCWS\_GetGCWSMessage 関数を使用して、  
距離計算サーバからエラー文字列を取得します。

**int WINAPI ACT\_DCWS\_GetUsableNWInfo(NWInfo \* pNWInfo, int nIndex)**

管理サーバに対し計算用道路データ情報を取得します。

パラメータ	解説
NWInfo * pNWInfo	計算用道路データ構造体バッファへのポインタ
int nIndex	取得したい計算用道路データ構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定した計算用道路データ情報を取得します。



## **int WINAPI ACT\_DCWS\_GetUsableDCServer(NWInfo \* pNWInfo)**

管理サーバに対し距離計算サーバ取得サービスを呼び出し、距離計算サーバの情報を取得します。取得した距離計算サーバの情報は、DLL 内部の環境設定構造体に格納します。

パラメータ	解説
NWInfo * pNWInfo	計算したい計算用道路データの情報を設定した計算用道路データ構造体へのポインタ

### **戻り値**

正常終了時は 1、異常終了時は 0。

### **備考**

距離計算サーバ取得サービスを呼び出し、距離計算サーバの情報を取得します。取得した距離計算サーバの情報は、DLL 内部の環境設定構造体に格納します。

**int WINAPI ACT\_DCWS\_GetUserInfo (UserInfo \* pUserInfo)**

管理サーバに対しユーザ情報取得サービスを呼び出し、ユーザ情報を取得します。

---

**パラメータ**

**解説**

UserInfo \* pUserInfo

ユーザ情報構造体を返す領域

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

ユーザ ID、パスワードを指定し、ユーザ情報を取得します。

**int WINAPI ACT\_DCWS\_Invoke(LPSTR lpszCommand, LPSTR lpszResultMessage)**

距離計算サーバに対し距離計算サービス呼び出し、距離計算を実行します。

パラメータ	解説
LPSTR lpszCommand	距離計算コマンド文字列へのポインタ
LPSTR lpszResultMessage	距離計算サービスの結果メッセージを格納するバッファへのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**結果メッセージ例**

- ・連続して距離計算を実行した時  
前回の処理が終了してから 10 秒経過していません。
- ・ルート計算時に到達できない地点を選択した時  
ノード 0000000000000000 - 9999999999999999 の距離計算に失敗しました。

**備考**

距離計算サービス呼び出し、距離計算を実行します。

距離計算コマンド文字列には、次の文字列を設定してください。

"CALCROUTE"	ルート計算
"CALCOPTROUTE"	最短ルート計算
"CALCAREA"	到達圏／流入圏計算
"GETNEARESTNODE"	最寄ノード取得
"ENUMNEARNODE"	近傍ノード列挙
"GETNETWORKINFO"	計算用道路データ情報取得

距離計算サービスの結果メッセージを格納するバッファは、256 バイト以上のエリアを確保してください。

到達圏／流入圏計算を行う場合、ポリゴン修正機能（Appendix A）、ポリゴン始点包含機能（Appendix B）も必要に応じてご利用ください。

**int WINAPI ACT\_DCWS\_CalcArea2 (void)**

距離計算サーバに対し到達圏/流入圏計算サービスを呼び出し、到達圏/流入圏計算を実行します。

**パラメータ****解説**

---

なし

**戻り値**

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

**備考**

到達圏/流入圏計算サービスを呼び出し、到達圏/流入圏計算を実行します。

予め **ACT\_DCWS\_SetCA2Param** 関数を使用し、到達圏／流入圏計算用パラメータを設定する必要があります。

ポリゴン修正機能（Appendix A）も必要に応じてご利用ください。

## **int WINAPI ACT\_DCWS\_CalcArea (void)**

距離計算サーバに対し到達圏/流入圏計算サービスを呼び出し、到達圏/流入圏計算を実行します。

※ このサービスは互換性のために残されています。今後新たに到達圏／流入圏計算サービスを利用するプログラムを作成する場合は **ACT\_DCWS\_CalcArea2** を使用してください。

また単一ポリゴン計算時に到達圏／流入圏内ノード数が 3 万未満の場合、自動でポリゴン補完機能が有効になります。

---

### **パラメータ**

### **解説**

なし

### **戻り値**

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

### **備考**

到達圏/流入圏計算サービスを呼び出し、到達圏/流入圏計算を実行します。

予め **ACT\_DCWS\_SetCAParam** 関数を使用し、到達圏／流入圏計算用パラメータを設定する必要があります。

ポリゴン修正機能（Appendix A）も必要に応じてご利用ください。

**int WINAPI ACT\_DCWS\_CalcOneWay2 (void)**

距離計算サーバに対し片道一括計算サービスを呼び出し、片道一括計算を実行します。

---

パラメータ	解説
-------	----

---

なし	
----	--

**戻り値**

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

**備考**

片道一括計算サービスを呼び出し、片道一括計算を実行します。

予め **ACT\_DCWS\_SetCOW2Param** 関数、**ACT\_DCWS\_AddGeoLoc** 関数を使用し、片道一括計算用パラメータとジオロケーションを設定する必要があります。

## **int WINAPI ACT\_DCWS\_CalcOneWay (void)**

距離計算サーバに対し片道一括計算サービスを呼び出し、片道一括計算を実行します。

※ このサービスは互換性のために残されています。今後新たに到達圏／流入圏計算サービスを利用するプログラムを作成する場合は ACT\_DCWS\_CalcOneway2 を使用してください。

---

### **パラメータ**

### **解説**

---

なし

### **戻り値**

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

### **備考**

片道一括計算サービスを呼び出し、片道一括計算を実行します。

予め **ACT\_DCWS\_SetCOWParam** 関数、**ACT\_DCWS\_AddGeoLoc** 関数を使用し、片道一括計算用パラメータとジオロケーションを設定する必要があります。

**int WINAPI ACT\_DCWS\_GetUsableGeoServer(void)**

住所検索サーバに対し住所検索サーバ取得サービスを呼び出し、住所検索サーバの情報を取得します。取得した住所検索サーバの情報は、DLL 内部の住所検索用環境設定構造体に格納します。

---

パラメータ	解説
-------	----

---

なし	
----	--

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

住所検索サーバ取得サービスを呼び出し、住所検索サーバの情報を取得します。取得した住所検索サーバの情報は、DLL 内部の住所検索用環境設定構造体に格納します。



**int WINAPI ACT\_DCWS\_GeoInvoke(LPSTR lpszCommand, LPSTR lpszResultMessage)**

住所検索サーバに対し住所検索サービスを呼び出し、住所検索を実行します。

パラメータ	解説
LPSTR lpszCommand	住所検索コマンド文字列へのポインタ
LPSTR lpszResultMessage	住所検索サービスの結果メッセージを格納するバッファへのポインタ

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

住所検索サービスを呼び出し、住所検索を実行します。

住所検索コマンド文字列には、次の文字列を設定してください。

"SEARCHADDRESS"	住所検索
"SEARCHADDRESSCODE"	住所コード検索
"ENUMLOWERADDRESSBYCODE"	下位住所列挙
"SEARCHZIP"	郵便番号検索
"ENUMADDRESSBYZIP"	郵便番号対応住所列挙
"GETGEOINFO"	住所データ情報取得

住所検索サービスの結果メッセージを格納するバッファは、256 バイト以上のエリアを確保してください。

```
int WINAPI ACT_GCWS_SearchAddress(
    Environ* pEnviron,
    GeoCommonParam* pGeoCommonParam,
    int searchAddressLevel,
    int countSearchAddressItem,
    GeoSearchAddressItem* pGeoSearchAddressItem,
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、住所検索を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
int searchAddressLevel	住所検索レベル (0～32)
int countSearchAddressItem	住所検索アイテム数
GeoSearchAddressItem* pGeoSearchAddressItem	住所検索アイテム
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

GeoSearchAddressItem の szSearchString の住所を検索します。

```
int WINAPI ACT_GCWS_CreateFullAddressByCd(  
    Environ* pEnviron,  
    GeoCommonParam* pGeoCommonParam,  
    int countCreateFullAddressByCodeItem,  
    GeoCreateFullAddressByCodeItem* pGeoCreateFullAddressByCodeItem,  
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、全住所取得を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
int countCreateFullAddressByCodeItem	全住所アイテム数
GeoCreateFullAddressByCodeItem* pGeoCreateFullAddressByCodeItem	全住所アイテム
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

GeoCreateFullAddressByCodeItem の szSearchCodeString の住所コードを検索します。

```
int WINAPI ACT_GCWS_SearchAddressCd(
    Environ* pEnviron,
    GeoCommonParam* pGeoCommonParam,
    int countSearchAddressCodeItem,
    GeoSearchAddressCodeItem* pGeoSearchAddressCodeItem,
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、住所コード検索を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
int countSearchAddressCodeItem	住所コード検索アイテム数
GeoSearchAddressCodeItem* pGeoSearchAddressCodeItem	住所コード検索アイテム
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

GeoSearchAddressCodeItem の szSearchCodeString の住所コードを検索します。

```
int WINAPI ACT_GCWS_CountLowerAddressByCd(  
    Environ* pEnviron,  
    GeoCommonParam* pGeoCommonParam,  
    char* pSearchAddressCode,  
    int* pCountLowerAddressByCode,  
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、下位住所数取得を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
char* pSearchAddressCode	検索住所コード
int* pCountLowerAddressByCode	下位住所数
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

pSearchAddressCode の住所コードを検索し、下位住所数を pCountLowerAddressByCode に返却します。

```

int WINAPI ACT_GCWS_EnumLowerAddressByCd(
    Environ* pEnviron,
    GeoCommonParam* pGeoCommonParam,
    char* pSearchAddressCode,
    int countGeoAddressItem,
    GeoAddressItem *pGeoAddressItem,
    int* pCountLowerAddressByCode,
    GeoInfo* pGeoInfo)

```

住所検索サーバに対し住所検索サービスを呼び出し、下位住所列挙を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
char* pSearchAddressCode	検索住所コード
int countGeoAddressItem	下位住所アイテム数
GeoAddressItem *pGeoAddressItem	下位住所アイテム
int* pCountLowerAddressByCode	書き込み下位住所数
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

pSearchAddressCode の住所コードを検索し、GeoAddressItem に下位住所を列挙します。  
countGeoAddressItem が書き込み下位住所数よりも少ない場合、エラーになります。  
エラーの場合も pCountLowerAddressByCode には書き込み下位住所数が返却されます。

```
int WINAPI ACT_GCWS_GetNearestAddress(  
    Environ* pEnviron,  
    GeoCommonParam* pGeoCommonParam,  
    int countNearestAddressItem,  
    GeoNearestAddressItem* pGeoNearestAddressItem,  
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、最寄住所取得を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
int countNearestAddressItem	最寄住所アイテム数
GeoNearestAddressItem* pGeoNearestAddressItem	最寄住所アイテム
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

GeoNearestAddressItem の szLon, szLat（緯度、経度）から住所を検索します。

```
int WINAPI ACT_GCWS_SearchZip(  
    Environ* pEnviron,  
    GeoCommonParam* pGeoCommonParam,  
    int countSearchZipItem,  
    GeoSearchZipItem* pGeoSearchZipItem,  
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、郵便番号検索を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
int countSearchZipItem	郵便番号検索アイテム数
GeoSearchZipItem* pGeoSearchZipItem	郵便番号検索アイテム
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

GeoSearchZipItem の szZip から住所を検索します。



```
int WINAPI ACT_GCWS_CountAddressByZip(  
    Environ* pEnviron,  
    GeoCommonParam* pGeoCommonParam,  
    char* pSearchZipCode,  
    int* pCountAddressByZip,  
    GeoInfo* pGeoInfo)
```

住所検索サーバに対し住所検索サービスを呼び出し、郵便番号対応住所数取得を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
char* pSearchZipCode	検索郵便番号
int* pCountAddressByZip	検索郵便番号対応住所数
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

pSearchZipCode の郵便番号を検索し、検索郵便番号対応住所数を pCountAddressByZip に返却します。

```

int WINAPI ACT_GCWS_EnumAddressByZip(
    Environ* pEnviron,
    GeoCommonParam* pGeoCommonParam,
    char* pSearchZipCode,
    int countGeoZipItem,
    GeoZipItem* pGeoZipItem,
    int* pCountAddressByZip,
    GeoInfo* pGeoInfo)

```

住所検索サーバに対し住所検索サービスを呼び出し、郵便番号対応住所列挙を実行します。

パラメータ	解説
Environ * pEnv	環境設定
GeoCommonParam* pGeoCommonParam	住所検索共通パラメータ
char* pSearchZipCode	検索郵便番号
int countGeoZipItem	郵便番号アイテム数
GeoZipItem* pGeoZipItem	郵便番号アイテム
int* pCountAddressByZip	書き込み郵便番号アイテム数
GeoInfo* pGeoInfo	住所データ情報

#### 戻り値

正常終了時は 0、異常終了時は 0 以外（エラーコード）。

#### 備考

pSearchZipCode の郵便番号を検索し、GeoZipItem に検索郵便対応住所を列挙します。  
countGeoZipItem が書き込み郵便番号アイテム数よりも少ない場合、エラーになります。  
エラーの場合も pCountAddressByZip には書き込み郵便番号アイテム数が返却されます。

(2) データ取得／設定関数

**int WINAPI ACT\_DCWS\_GetEnviron(Environ \* pEnv)**

環境設定構造体を取得します。

パラメータ	解説
Environ * pEnv	環境設定構造体を格納するバッファへのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持している環境設定構造体の情報を取得します。

```
int WINAPI ACT_DCWS_SetEnviron(Environ * pEnv)
```

環境設定構造体を設定します。

パラメータ	解説
-------	----

---

Environ * pEnv	環境設定構造体へのポインタ
----------------	---------------

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

環境設定構造体の情報を DLL 内部に設定します。

**int WINAPI ACT\_DCWS\_GetParam(Param \* pPar)**

パラメータ構造体を取得します。

パラメータ	解説
-------	----

Param * pPar	パラメータ構造体を格納するバッファへのポインタ
--------------	-------------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持しているパラメータ構造体の情報を取得します。

**int WINAPI ACT\_DCWS\_SetParam(Param \* pPar)**

パラメータ構造体を設定します。

パラメータ	解説
Param * pPar	パラメータ構造体へのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

パラメータ構造体の情報を DLL 内部に設定します。

## **int WINAPI ACT\_DCWS\_GetLocsCount(void)**

ロケーション構造体配列の要素数を取得します。

<b>パラメータ</b>	<b>解説</b>
--------------	-----------

---

なし

## **戻り値**

ロケーション構造体配列の要素数。異常終了時は-1。

## **備考**

DLL 内部に保持しているロケーション構造体配列の要素数を取得します。

**int WINAPI ACT\_DCWS\_ClearLocs(void)**

ロケーション構造体配列をクリアし、要素数を 0 にします。

**パラメータ****解説**

---

なし

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持しているロケーション構造体配列をクリアし、要素数を 0 にします。



**int WINAPI ACT\_DCWS\_GetLoc(Loc \* pLoc, int nIndex)**

ロケーション構造体を取得します。

パラメータ	解説
Loc * pLoc	取得したロケーション構造体を格納するバッファへのポインタ
int nIndex	取得したいロケーション構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定したロケーション構造体を取得します。

**int WINAPI ACT\_DCWS\_AddLoc(Loc \* pLoc)**

ロケーション構造体を追加します。

パラメータ	解説
-------	----

---

Loc * pLoc	追加したいロケーション構造体へのポインタ
------------	----------------------

**戻り値**

追加後のロケーション総数。 異常終了時は-1

**備考**

ロケーション構造体の情報を DLL 内部のロケーション構造体配列に追加します。

## **int WINAPI ACT\_DCWS\_GetPntsCount(void)**

ポイント構造体配列の要素数を取得します。

パラメータ	解説
-------	----

---

なし	
----	--

### **戻り値**

ポイント構造体配列の要素数。異常終了時は-1。

### **備考**

DLL 内部に保持しているポイント構造体配列の要素数を取得します。

**int WINAPI ACT\_DCWS\_GetPnt(Pnt \* pPnt, int nIndex)**

ポイント構造体を取得します。

パラメータ	解説
Pnt * pPnt	取得したポイント構造体を格納するバッファへのポインタ
int nIndex	取得したいポイント構造体のインデックス（0～）

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

インデックスで指定したポイント構造体を取得します。

## **int WINAPI ACT\_DCWS\_GetRnsCount(void)**

経由道路名構造体配列の要素数を取得します。

パラメータ	解説
-------	----

---

なし

### **戻り値**

経由道路名構造体配列の要素数。異常終了時は-1。

### **備考**

DLL 内部に保持している経由道路名構造体配列の要素数を取得します。

**int WINAPI ACT\_DCWS\_GetRn(Rn \* pRn, int nIndex)**

経由道路名構造体を取得します。

パラメータ	解説
Rn * pRn	取得した経由道路名構造体を格納するバッファへのポインタ
int nIndex	取得したい経由道路名構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定した経由道路名構造体を取得します。

## **int WINAPI ACT\_DCWS\_GetTrdsCount(void)**

経由有料道路構造体配列の要素数を取得します。

パラメータ	解説
-------	----

---

なし	
----	--

### **戻り値**

経由有料道路構造体配列の要素数。異常終了時は-1。

### **備考**

DLL 内部に保持している経由有料道路構造体配列の要素数を取得します。

**int WINAPI ACT\_DCWS\_GetTrd(Trd \* pTrd, int nIndex)**

経由有料道路構造体を取得します。

パラメータ	解説
Trd * pTrd	取得した経由有料道路構造体を格納するバッファへのポインタ
int nIndex	取得したい経由有料道路構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定した経由有料道路構造体を取得します。



**int WINAPI ACT\_DCWS\_GetNWInfo(NWInfo \* pNWInfo)**

計算用道路データ情報構造体を取得します。

パラメータ	解説
NWInfo * pNWInfo	取得した計算用道路データ情報構造体を格納するバッファへのポインタ

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

DLL 内部の計算用道路データ情報構造体を取得します。

**int WINAPI ACT\_DCWS\_GetGeoEnviron(Environ \* pEnv)**

住所検索用環境設定構造体を取得します。

---

**パラメータ**

**解説**

Environ \* pEnv

住所検索用環境設定構造体を格納するバッファへのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持している住所検索用環境設定構造体の情報を取得します。

**int WINAPI ACT\_DCWS\_SetGeoEnviron(Environ \* pEnv)**

住所検索用環境設定構造体を設定します。

パラメータ	解説
-------	----

---

Environ * pEnv	住所検索用環境設定構造体へのポインタ
----------------	--------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

住所検索用環境設定構造体の情報を DLL 内部に設定します。

```
int WINAPI ACT_DCWS_GetGeoParam(GeoParam * pGeoPar)
```

住所検索用パラメータ構造体を取得します。

パラメータ	解説
-------	----

GeoParam * pGeoPar	住所検索用パラメータ構造体を格納するバッファへのポインタ
--------------------	------------------------------

戻り値

正常終了時は 1、異常終了時は 0。

備考

DLL 内部に保持している住所検索用パラメータ構造体の情報を取得します。

**int WINAPI ACT\_DCWS\_SetGeoParam(GeoParam \* pGeoPar)**

住所検索用パラメータ構造体を設定します。

パラメータ	解説
-------	----

---

GeoParam * pGeoPar	住所検索用パラメータ構造体へのポインタ
--------------------	---------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

住所検索用パラメータ構造体の情報を DLL 内部に設定します。

**int WINAPI ACT\_DCWS\_GetGeoItemsCount(void)**

ジオアイテム構造体配列の要素数を取得します。

---

パラメータ	解説
-------	----

---

なし	
----	--

**戻り値**

ジオアイテム構造体配列の要素数。異常終了時は-1。

**備考**

DLL 内部に保持しているジオアイテム構造体配列の要素数を取得します。

## **int WINAPI ACT\_DCWS\_ClearGeoItems(void)**

ジオアイテム構造体配列をクリアし、要素数を 0 にします。

パラメータ	解説
-------	----

---

なし	
----	--

### **戻り値**

正常終了時は 1、異常終了時は 0。

### **備考**

DLL 内部に保持しているジオアイテム構造体配列をクリアし、要素数を 0 にします。

**int WINAPI ACT\_DCWS\_GetGeoItem(GeoItem \* pGeoItem, int nIndex)**

ジオアイテム構造体を取得します。

パラメータ	解説
GeoItem * pGeoItem	取得したジオアイテム構造体を格納するバッファへのポインタ
int nIndex	取得したいジオアイテム構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定したジオアイテム構造体を取得します。



**int WINAPI ACT\_DCWS\_AddGeoItem(GeoItem \* pGeoItem)**

ジオアイテム構造体を追加します。

**パラメータ**

**解説**

---

GeoItem * pGeoItem	追加したいジオアイテム構造体へのポインタ
--------------------	----------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

ジオアイテム構造体の情報を DLL 内部のジオアイテム構造体配列に追加します。

```
int WINAPI ACT_DCWS_GetGeoInfo(GeoInfo * pGeoInfo)
```

住所データ情報構造体を取得します。

---

**パラメータ****解説**

GeoInfo * pGeoInfo	取得した住所データ情報構造体を格納するバッファへのポインタ
--------------------	-------------------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部の住所データ情報構造体を取得します。

**int WINAPI ACT\_DCWS\_GetCA2Param(CA2Param \* pCA2Par)**

到達圏／流入圏計算用パラメータ構造体を取得します。(ACT\_DCWS\_CalcArea2 用)

---

**パラメータ**

**解説**

---

CA2Param \* pCA2Par    到達圏／流入圏計算用パラメータ構造体を格納するバッファへのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持している到達圏／流入圏計算用パラメータ構造体の情報を取得します。

```
int WINAPI ACT_DCWS_GetCAParam(CAParam * pCAPar)
```

到達圏／流入圏計算用パラメータ構造体を取得します。

パラメータ	解説
CAParam * pCAPar	到達圏／流入圏計算用パラメータ構造体を格納するバッファへのポインタ

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

DLL 内部に保持している到達圏／流入圏計算用パラメータ構造体の情報を取得します。

**int WINAPI ACT\_DCWS\_SetCA2Param(CA2Param \* pCA2Par)**

到達圏／流入圏計算用パラメータ構造体を設定します。(ACT\_DCWS\_CalcArea2 用)

パラメータ	解説
-------	----

---

CA2Param * pCA2Par	到達圏／流入圏計算用パラメータ構造体へのポインタ
--------------------	--------------------------

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

到達圏／流入圏計算用パラメータ構造体の情報を DLL 内部に設定します。

**int WINAPI ACT\_DCWS\_SetCAParam(CAParam \* pCAPar)**

到達圏／流入圏計算用パラメータ構造体を設定します。

---

**パラメータ**

**解説**

CAParam * pCAPar	到達圏／流入圏計算用パラメータ構造体へのポインタ
------------------	--------------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

到達圏／流入圏計算用パラメータ構造体の情報を DLL 内部に設定します。

**int WINAPI ACT\_DCWS\_GetCOW2Param(COW2Param \* pCOW2Par)**

片道一括計算用パラメータ構造体を取得します。(ACT\_DCWS\_CalcOneway2 用)

パラメータ	解説
-------	----

---

COW2Param * pCOW2Par	片道一括計算用パラメータ構造体を格納するバッファへのポインタ
----------------------	--------------------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持している片道一括計算用パラメータ構造体の情報を取得します。

**int WINAPI ACT\_DCWS\_GetCOWParam(COWParam \* pCOWPar)**

片道一括計算用パラメータ構造体を取得します。

パラメータ	解説
-------	----

---

COWParam * pCOWPar	片道一括計算用パラメータ構造体を格納するバッファへのポインタ
--------------------	--------------------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持している片道一括計算用パラメータ構造体の情報を取得します。



**int WINAPI ACT\_DCWS\_SetCOW2Param(COW2Param \* pCOW2Par)**

片道一括計算用パラメータ構造体を設定します。(ACT\_DCWS\_CalcOneway2 用)

パラメータ	解説
-------	----

---

COW2Param * pCOW2Par	片道一括計算用パラメータ構造体へのポインタ
----------------------	-----------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

片道一括計算用パラメータ構造体の情報を DLL 内部に設定します。

**int WINAPI ACT\_DCWS\_SetCOWParam(COWParam \* pCOWPar)**

片道一括計算用パラメータ構造体を設定します。

パラメータ	解説
-------	----

---

COWParam * pCOWPar	片道一括計算用パラメータ構造体へのポインタ
--------------------	-----------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

片道一括計算用パラメータ構造体の情報を DLL 内部に設定します。

## **int WINAPI ACT\_DCWS\_GetGeoLocsCount(void)**

ジオロケーション構造体配列の要素数を取得します。

パラメータ	解説
-------	----

---

なし	
----	--

## **戻り値**

ジオロケーション構造体配列の要素数。異常終了時は-1。

## **備考**

DLL 内部に保持しているジオロケーション構造体配列の要素数を取得します。

**int WINAPI ACT\_DCWS\_ClearGeoLocs(void)**

ジオロケーション構造体配列をクリアし、要素数を 0 にします。

パラメータ	解説
-------	----

---

なし

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

DLL 内部に保持しているジオロケーション構造体配列をクリアし、要素数を 0 にします。

**int WINAPI ACT\_DCWS\_GetGeoLoc(GeoLoc \* pGeoLoc, int nIndex)**

ジオロケーション構造体を取得します。

パラメータ	解説
GeoLoc * pGeoLoc	取得したジオロケーション構造体を格納するバッファへのポインタ
int nIndex	取得したいジオロケーション構造体のインデックス（0～）

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

インデックスで指定したジオロケーション構造体を取得します。

**int WINAPI ACT\_DCWS\_AddGeoLoc(GeoLoc \* pGeoLoc)**

ジオロケーション構造体を追加します。

---

**パラメータ**

**解説**

GeoLoc \* pGeoLoc

追加したいジオロケーション構造体へのポインタ

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

ジオロケーション構造体の情報を DLL 内部のジオロケーション構造体配列に追加します。

(3) その他の関数

**int WINAPI ACT\_DCWS\_GetLibVersion(void)**

DLL のバージョン番号を取得します。

パラメータ	解説
-------	----

---

なし

**戻り値**

DLL のバージョン番号

**備考**

DLL のバージョン番号を取得します。

```
int WINAPI ACT_DCWS_LastErrorCode(void);
```

DLL 内部で最後に発生したエラーのエラーコードを取得します。

パラメータ	解説
-------	----

なし	
----	--

#### 戻り値

エラーコード

#### 備考

最後に発生したエラーのエラーコードを取得します。

エラーコードにはブリッジ DLL 内部で発生するものと、距離計算サーバ内部で発生するものとの2種類あります。ここでのエラーコードはブリッジ DLL 内部で発生するものです。エラー文字列の取得は ACT\_DCWS\_LastErrorMessage で行います。

距離計算サーバでのエラーは ACT\_DCWS\_CalcArea、ACT\_DCWS\_CalcOneWay の戻り値で返却されます。エラー文字列の取得は ACT\_DCWS\_GetDCWSMessage で行います。



**int WINAPI ACT\_DCWS\_LastErrorMessage(LPSTR lpszMessage)**

DLL 内部で直近に発生したエラーのエラーメッセージを取得します。

パラメータ	解説
-------	----

LPSTR lpszMessage	エラーメッセージを格納するバッファへのポインタ
-------------------	-------------------------

**戻り値**

正常終了時は最後に発生したエラーのエラーコード、異常終了時は -1。

**備考**

直近に発生したエラーのエラーメッセージを取得します。エラーメッセージを格納するバッファは、512 バイト以上のエリアを確保してください。

**int WINAPI ACT\_DCWS\_SetProxy (LPSTR lpszServer, int nPort, LPSTR lpszUserName, LPSTR lpszPassword)**

プロキシ情報を設定します。

※ この関数は互換性のために残されています。今後新たにプロキシ情報をセットする場合は ACT\_DCWS\_SetProxyInfoA, または ACT\_DCWS\_SetProxyInfoW を使用してください。

パラメータ	解説
LPSTR lpszServer	サーバー名 (最大 2,047 バイト)
int nPort	ポート No
LPSTR lpszUserName	ユーザー名 (最大 2,047 バイト)
LPSTR lpszPassword	パスワード (最大 2,047 バイト)

#### 戻り値

正常終了時は 0、異常終了時は -1。

#### 備考

プロキシ情報を DLL 内部に設定します。

```
int WINAPI ACT_DCWS_SetProxyInfoA( int proxySetting, char* lpszAutoScriptUrl, char*
lpszServer, char* lpszPort, char* lpszUserName, char* lpszPassword)
```

プロキシ情報を設定します。(マルチバイト文字)

パラメータ	解説
int proxySetting	プロキシ設定(DCWS_PROXY_?)
char* lpszAutoScriptUrl	自動スクリプト URL (DCWS_PROXY_USER_AUTOSCRIPT 指定時参照)
char* lpszServer	サーバー名 (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)
char* lpszPort	ポート No (DCWS_PROXY_USER_USER 指定時参照)
char* lpszUserName	ユーザー名 (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)
char* lpszPassword	パスワード (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)

## 戻り値

0:正常終了 -1:異常終了

## 備考

プロキシ情報を DLL 内部に設定します。

プロキシ設定の値は、DCWS\_PROXY\_??の論理和を入力します。

ユーザ指定 (DCWS\_PROXY\_USER\_USER) を指定した場合、

自動設定 (DCWS\_PROXY\_USER\_AUTOCONFIG) 、

自動スクリプト (DCWS\_PROXY\_USER\_AUTOSCRIPT ) 、

プロキシ指定 (DCWS\_PROXY\_USER\_USER) のいずれかを指定してください。

ユーザ指定の場合、設定取得に失敗した場合は、エラーが返却されます。

例 1) IE のプロキシを使用

```
proxySetting = DCWS_PROXY_USE | DCWS_PROXY_TYPE_NAMED_PROXY |
DCWS_PROXY_IE_USE
```

例 2) プロキシを指定

```
proxySetting = DCWS_PROXY_USE | DCWS_PROXY_TYPE_NAMED_PROXY |
DCWS_PROXY_IE_NOUSE | DCWS_PROXY_USER_USER
```

```
int WINAPI ACT_DCWS_SetProxyInfoW( int proxySetting, LPWSTR lpszAutoScriptUrl,
LPWSTR lpszServer, LPWSTR lpszPort, LPWSTR lpszUserName, LPWSTR lpszPassword )
```

プロキシ情報を設定します。(ワイド文字)

パラメータ	解説
int proxySetting	プロキシ設定(DCWS_PROXY_?)
LPWSTR lpszAutoScriptUrl	自動スクリプト URL (DCWS_PROXY_USER_AUTOSCRIPT 指定時参照)
LPWSTR lpszServer	サーバー名 (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)
LPWSTR lpszPort	ポート No (DCWS_PROXY_USER_USER 指定時参照)
LPWSTR lpszUserName	ユーザー名 (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)
LPWSTR lpszPassword	パスワード (最大 2,047 バイト) (DCWS_PROXY_USER_USER 指定時参照)

#### 戻り値

0:正常終了 -1:異常終了

#### 備考

プロキシ情報を DLL 内部に設定します。

プロキシ設定の値は、DCWS\_PROXY\_??の論理和を入力します。

プロキシ設定の設定例は ACT\_DCWS\_SetProxyInfoA の備考を参照ください。

**int WINAPI ACT\_DCWS\_GetDCWSMessage (int nMessageNo, LPSTR lpszMessage)**

サーバ側で発生したエラーメッセージ番号からメッセージを取得します。

パラメータ	解説
int nMessageNo	取得したいメッセージの番号
LPSTR lpszMessage	メッセージを格納するバッファへのポインタ

#### 戻り値

正常終了時は 1、異常終了時は 0。

#### 備考

メッセージを格納するバッファは、512 バイト以上のエリアを確保してください。

**int WINAPI ACT\_GCWS\_GetGCWSMessage (int nMessageNo, LPSTR lpszMessage)**

サーバ側で発生したエラーメッセージ番号からメッセージを取得します。（住所検索用）

パラメータ	解説
-------	----

---

int nMessageNo	取得したいメッセージの番号
----------------	---------------

LPSTR lpszMessage	メッセージを格納するバッファへのポインタ
-------------------	----------------------

**戻り値**

正常終了時は 1、異常終了時は 0。

**備考**

メッセージを格納するバッファは、512 バイト以上のエリアを確保してください。

## 7. 5 定数定義

```

// Invoke コマンド
#define COMMAND_CALCROUTE "CALCROUTE" // ルート計算
#define COMMAND_CALCOPTROUTE "CALCOPTROUTE" // 最短ルート計算
#define COMMAND_CALCAREA "CALCAREA" // 到達圏計算
#define COMMAND_GETNETWORKINFO "GETNETWORKINFO" // ネットワーク情報取得
#define COMMAND_GETNEARESTNODE "GETNEARESTNODE" // 近傍ノード取得
#define COMMAND_ENUMNEARNODE "ENUMNEARNODE" // 近傍ノード候補取得

// 計算方法(時間最短:1 距離最短:2)
#define CALCKIND_TIME 1 // 時間最短
#define CALCKIND_DIST 2 // 距離最短

// 高速使用(使用する:0 使用しない:1)
#define USEHIGHWAY_USE 0 // 使用する
#define USEHIGHWAY_NOTUSE 1 // 使用しない

// 交通機関フラグ
#define USEHIGHWAY_PAYROADFR 2 // 有料道路/フェリー道路
#define USEHIGHWAY_FERRY 4 // フェリー
#define USEHIGHWAY_RAILROAD 8 // 鉄道
#define USEHIGHWAY_AIRWAY 16 // 航空
#define USEHIGHWAY_SHIP 64 // 内航船
#define USEHIGHWAY_WALK 256 // 歩行者

// Locs(ロケーション情報)出力フラグ(出力:1 しない:0)
#define LOCSOUTPUT_OUTPUT 1 // 出力
#define LOCSOUTPUT_NOTOUTPUT 0 // しない
// Pnts(ルート形状、ポリゴン形状)出力フラグ(出力:1 しない:0)
#define PNTSOUTPUT_OUTPUT 1 // 出力
#define PNTSOUTPUT_NOTOUTPUT 0 // しない
// Rns(経由道路名)出力フラグ(出力:1 しない:0)
#define RNSOUTPUT_OUTPUT 1 // 出力
#define RNSOUTPUT_NOTOUTPUT 0 // しない
// Trds(経由有料道路)出力フラグ(出力:1 しない:0)
#define TRDSOUTPUT_OUTPUT 1 // 出力
#define TRDSOUTPUT_NOTOUTPUT 0 // しない
// 高速区間計算フラグ(計算する:1 しない:0)
#define HIGHWAYOUTPUT_OUTPUT 1 // 計算する
#define HIGHWAYOUTPUT_NOTOUTPUT 0 // しない
// 通行料金計算フラグ(計算する:1 しない:0)
#define TOLLOUTPUT_OUTPUT 1 // 計算する
#define TOLLOUTPUT_NOTOUTPUT 0 // しない
// 詳細ルートフラグ(詳細ルート:1 簡易ルート:0)
#define ROUTEKIND_DETAIL 1 // 詳細ルート
#define ROUTEKIND_SIMPLE 0 // 簡易ルート
// 時間圏/距離圏指定(時間圏:1 距離圏:2)
#define AREAKIND_TIME 1 // 時間圏
#define AREAKIND_DIST 2 // 距離圏
// 流入圏計算フラグ(到達圏:0x0000 流入圏:0x1000)
#define AREADIRECTION_OUTFLOW 0x0000 // 到達圏
#define AREADIRECTION_INFLOW 0x1000 // 流入圏

```

```

// 文字列領域の最大サイズ (文字列終了を示すヌルを含む)
#define MAX_USERID_LEN 64 // ユーザ ID
#define MAX_UUID_LEN 64 // ユーザユニーク ID
#define MAX_HWUID_LEN 64 // ハードウェアユニーク ID
#define MAX_WSURL_LEN 256 // 距離計算サービス URL
#define MAX_HOST_LEN 64 // 距離計算サーバ ホスト名 or IP アドレス("xxx.xxx.xxx.xxx"形式)
#define MAX_NODECODE_LEN 32 // ノードコード
#define MAX_LINKCODE_LEN 32 // リンクコード
#define MAX_TAG_LEN 128 // タグ
#define MAX_ROADNAME_LEN 64 // 道路名・交差点名
#define MAX_ICNAME_LEN 64 // IC 名
#define MAX_FOLDER_LEN 260 // 計算用道路データ格納フォルダ
#define MAX_NOTE_LEN 256 // 備考
#define MAX_NETWORKID_LEN 64 // ネットワーク ID
#define MAX_LONLAT_LEN 32 // 経度/緯度
#define MAX_ERRMSG_LEN 512 // エラーメッセージ
#define MAX_RESULTMSG_LEN 256 // 結果メッセージ
#define MAX_STR64_LEN 64 // 汎用文字列(64 バイト)
#define MAX_ZIPCODE_LEN 16 // 郵便番号
#define MAX_ADDRESSCODE_LEN 24 // 住所コード
#define MAX_ADDRESSNAME_LEN 40 // 住所名称
#define MAX_KANANAME_LEN 64 // 住所よみ
#define MAX_ADDRESS_LEN 256 // 住所全体
// ポリゴン種別
#define POLYGONKIND_SINGLEPOLYGON 1 // シングルポリゴン
#define POLYGONKIND_MULTIPOLYGON 2 // 複数ポリゴン

// 住所検索 Invoke コマンド
#define COMMAND_SEARCHADDRESS "SEARCHADDRESS"
#define COMMAND_SEARCHADDRESSCODE "SEARCHADDRESSCODE"
#define COMMAND_ENUMLOWERADDRESSBYCODE "ENUMLOWERADDRESSBYCODE"
#define COMMAND_SEARCHZIP "SEARCHZIP"
#define COMMAND_ENUMADDRESSBYZIP "ENUMADDRESSBYZIP"
#define COMMAND_GETGEOINFO "GETGEOINFO"
// 住所レベル
#define ADDRLEVEL_NONE 0; // なし
#define ADDRLEVEL_PREF 1; // 都道府県
#define ADDRLEVEL_CITY 2; // 市区町村
#define ADDRLEVEL_STREET 8; // 大字・丁目
#define ADDRLEVEL_BLOCK 16; // 街区
#define ADDRLEVEL_HOUSE 32; // 号
// 郵便番号検索フラグ
#define ZIP_SELTOP 0x01; // 候補が複数の時、先頭を採用する
#define ZIP_SELECTR 0x08; // 候補が複数の時、「住所の中心」を採用する
#define ZIP_SELECTRADR 0x10; // 候補が複数の時、「中心に近い住所」を採用する

// プロキシフラグ
#define DCWS_PROXY_NOUSE 0 // プロキシ使用しない
#define DCWS_PROXY_USE 0x1 // プロキシ使用する
#define DCWS_PROXY_TYPE_DEFAULT_PROXY 0 // デフォルトプロキシを使用する
#define DCWS_PROXY_TYPE_NAMED_PROXY 0x2 // ユーザ指定プロキシを使用する
#define DCWS_PROXY_IE_NOUSE 0 // IE の設定を使用しない
#define DCWS_PROXY_IE_USE 0x4 // IE の設定を使用する
#define DCWS_PROXY_USER_AUTOCONFIG 0x8 // 自動設定を取得する
#define DCWS_PROXY_USER_AUTOSCRIPT 0x10 // 自動スクリプトを取得する
#define DCWS_PROXY_USER_USER 0x20 // プロキシ設定を指定
// プロキシ文字列
#define DCWS_PROXY_MAX_STRING_LENGTH 2048

```



## 7. 6 構造体定義

## (1) 環境設定構造体 (528 バイト)

```
typedef struct tagEnviron {
    char szUserID[64];           // ユーザ ID
    char szUUID[64];            // ユーザユニーク ID
    char szHWUID[64];           // ハードウェアユニーク ID
    char szWSUrl[256];          // 距離計算サービス URL
    char szHost[64];            // 距離計算リスナ IP アドレス("xxx.xxx.xxx.xxx"形式)
    int nPort;                  // 距離計算リスナ ポート番号
    char szDummy[12];           // ダミー
} Environ;
```

## (2) パラメータ構造体 (176 バイト)

```
typedef struct tagParam {
    int nDatum;                 // 座標系
    int nCoordinateFormat;      // 座標フォーマット
    int nCalcKind;              // 計算方法(時間最短:1 距離最短:2)
    int nNotUseHiway;           // 対応輸送手段 [注 (2) - 1]
    int nLocsOutput;            // Locs(ロケーション情報)出力フラグ
    int nPntsOutput;            // Pnts(ルート形状、ポリゴン形状)出力フラグ
    int nRnsOutput;             // Rns(経由道路名)出力フラグ
    int nTrdsOutput;            // Trds(経由有料道路)出力フラグ
    int nCalcRoute_HighwayOutput; // 高速区間計算フラグ(計算する:1 しない:0)
    int nCalcRoute_TollOutput;   // 通行料金計算フラグ(計算する:1 しない:0)
    int nCalcRoute_RouteKind;    // 詳細ルートフラグ(詳細ルート:1 簡易ルート:0)
    char szDummy[4];            // ダミー
    char szCalcArea_StartNode[32]; // 到達圏計算の中心ノード
    char szCalcArea_StartLon[32]; // 到達圏計算の中心位置の経度(単位:度)
    char szCalcArea_StartLat[32]; // 到達圏計算の中心位置の緯度(単位:度)
    int nCalcArea_AreaKind;      // 時間圏/距離圏指定(時間圏:1 距離圏:2)
    int nCalcArea_Range;         // 範囲指定 (単位:0.1 秒 or メートル) (-1 は FULL 計算)
    int nCalcArea_PolygonLevel;  // ポリゴンレベル (-1:凸型 0-20:凹凸型)
    int nCalcArea_AreaDirection; // 流入圏計算フラグ
    int nCalcArea_PolygonDetail; // ポリゴン詳細
    int nCalcOneway_HighwayOutput; // 高速区間計算フラグ(計算する:1 しない:0)
    int nCalcOneway_TollOutput;   // 通行料金計算フラグ(計算する:1 しない:0)
    char szDummy2[4];           // ダミー
} Param;
```

[注 (2) - 1]

対応輸送手段は以下論理和

高速道路 (0:使用する、1:使用しない)

北海道～青森間、沖縄～鹿児島間のみフェリーを使用 (2:使用する)

フェリー (4:使用する)

鉄道 (8:使用する)

航空 (16:使用する)

内航船 (鉄道使用の場合、特急使用) (64:使用する)

## (3) ロケーション構造体 (272 バイト)

```

typedef struct tagLoc {
    char szNode[32];           // ノードコード
    char szLon[32];           // 経度(単位:度)
    char szLat[32];           // 緯度(単位:度)
    int nTime;                 // 所要時間(分)
    int nDSec;                 // 所要時間(0.1 秒)
    int nDist;                 // 道のり(m)
    int nHighwayTime;         // 高速所要時間(分)
    int nHighwayDSec;         // 高速所要時間(0.1 秒)
    int nHighwayDist;         // 高速道のり(m)
    int nToll_SS;             // 通行料金(二輪・軽)
    int nToll_S;              // 通行料金(普通)
    int nToll_M;              // 通行料金(中型)
    int nToll_L;              // 通行料金(大型)
    int nToll_LL;             // 通行料金(特大)
    char szDummy[4];          // ダミー
    char szTag[128];          // タグ
} Loc;

```

## (4) ポイント構造体 (64 バイト)

```

typedef struct tagPnt {
    char szLon[32];           // 経度(単位:度)
    char szLat[32];           // 緯度(単位:度)
} Pnt;

```

## (5) 経由道路名構造体 (224 バイト)

```

typedef struct tagRn {
    int nLocFlag;              // ロケーションフラグ
    int nLinkFlag;            // リンクフラグ
    char szDummy1[8];          // ダミー
    char szNLCode[32];         // ノードリンクコード
    char szName[64];           // 道路名・交差点名
    int nTime;                 // 所要時間(分)
    int nDSec;                 // 所要時間(0.1 秒)
    int nDist;                 // 道のり(m)
    int nICType;              // IC 種別
    int nToll_SS;             // 通行料金(二輪・軽)
    int nToll_S;              // 通行料金(普通)
    int nToll_M;              // 通行料金(中型)
    int nToll_L;              // 通行料金(大型)
    int nToll_LL;             // 通行料金(特大)
    char szDummy2[12];         // ダミー
    char szLon[32];           // 経度(単位:度)
    char szLat[32];           // 緯度(単位:度)
} Rn;

```

## (6) 経由有料道路構造体 (288 バイト)

```
typedef struct tagTrd {  
    char szInRdName[64];           // 入口道路名  
    char szInICName[64];           // 入口 IC  
    char szOutRdName[64];          // 出口道路名  
    char szOutICName[64];          // 出口 IC  
    int  nToll_SS;                  // 通行料金(二輪・軽)  
    int  nToll_S;                   // 通行料金(普通)  
    int  nToll_M;                   // 通行料金(中型)  
    int  nToll_L;                   // 通行料金(大型)  
    int  nToll_LL;                  // 通行料金(特大)  
    int  nDist;                     // キロポスト(km)  
    char szDummy[8];               // ダミー  
} Trd;
```

## (7) 計算用道路データ情報構造体 (672 バイト)

```
typedef struct tagNWInfo {  
    char szFolder[260];             // 計算用道路データ格納フォルダ  
    char szDummy1[12];             // ダミー  
    char szNote[256];              // 備考  
    int  nIsDetailExist;           // 詳細データ存在する:1   しない:0  
    int  nIsTollExist;             // 有料データ存在する:1   しない:0  
    char szDummy2[8];              // ダミー  
    char szNWID[64];               // ネットワーク ID  
    char szNWID_WithoutSpeed[64];  // ネットワーク ID(道路速度を考慮しない場合)  
} NWInfo;
```

## (8) 住所検索パラメータ構造体 (272 バイト)

```
typedef struct tagGeoParam {  
    int  nDatum;                   // 座標系  
    int  nCoordinateFormat;        // 座標フォーマット  
    int  nAddressLevel;            // 住所レベル  
    int  nSearchFlag;              // 検索フラグ  
    char szSearchEnumString[256];  // 列挙検索文字列  
} GeoParam;
```

## ( 9 ) ジオアイテム構造体 ( 928 バイト )

```
typedef struct tagGeoItem {  
    char szSearchString[256];           // 検索文字列  
    short nResultLevel;                 // 検索住所レベル  
    short nUpperLevel;                  // 上の住所レベル  
    short nLowerLevel;                  // 下の住所レベル  
    short nDummy1;                       // ダミー  
    char szResultCode[24];               // 解析住所コード  
    char szResultAddress[40];            // 解析住所名称  
    char szDummy3[8];                   // ダミー  
    char szResultRestAddress[256];        // 解析不可部分住所  
    char szResultKana[64];               // 住所よみ  
    char szResultKataKana[64];           // 住所よみ  
    char szResultZip[16];                // 郵便番号  
    char szLon[32];                      // 経度(単位:度)  
    char szLat[32];                      // 緯度(単位:度)  
    char szTag[128];                     // タグ  
} GeoItem;
```

## ( 1 0 ) 住所データ情報構造体 ( 592 バイト )

```
typedef struct tagGeoInfo {  
    char szFolder[260];                  // 計算用道路データ格納フォルダ  
    char szDummy1[12];                   // ダミー  
    char szNote[256];                     // 備考  
    char szGeoID[64];                     // ネットワーク ID  
} GeoInfo;
```

## ( 1 1 ) ユーザ情報構造体 (464 バイト)

```

typedef struct tagUserInformation {
    char szMemberKind[MAX_STR64_LEN]; // 会員種別
    char szExpiryDate[MAX_STR64_LEN]; // 会員有効期限
    int nCalcInterval; // 計算間隔(秒)
    int nCalcRouteMaxLocs; // ルート計算時最大地点数
    int nCalcOptRouteMaxLocs; // 最短ルート計算時最大値点数
    int nCalcAreaMaxTime; // 到達圏最大範囲(分)
    int nCalcAreaMaxDist; // 到達圏最大範囲(km)
    int nCalcAreaMaxLocs; // 到達圏最大地点数
    int nGeoMaxAddressLevel; // 最大住所検索レベル
    int nGeoInterval; // 住所検索間隔
    int nGeoEnumInterval; // 住所列举間隔
    int nGeoMaxLocs; // 住所検索最大地点数
    int nCalcCount; // 距離計算回数
    int nMaxCalcCount; // 最大距離計算回数
    int nGeoCount; // 住所検索回数
    int nMaxGeoCount; // 最大住所検索回数
    char szDummy[8]; // ダミー
    int nExpand_I32_1Integer; // 拡張用数値 1
    int nExpand_I32_2Integer; // 拡張用数値 2
    int nExpand_I32_3Integer; // 拡張用数値 3
    int nExpand_I32_4Integer; // 拡張用数値 4
    char szExpand_Str_1[MAX_STR64_LEN]; // 拡張用文字列 1
    char szExpand_Str_2[MAX_STR64_LEN]; // 拡張用文字列 2
    char szExpand_Str_3[MAX_STR64_LEN]; // 拡張用文字列 3
    char szExpand_Str_4[MAX_STR64_LEN]; // 拡張用文字列 4
} UserInformation;

```

## ( 1 2 ) ジオポイント構造体 (704 バイト) 【V2.0 新規追加】

```

typedef struct tagGeoPnt {
    char szSearchString[MAX_ADDRESS_LEN]; // 検索文字列
    int nGeoLevel; // 検索住所レベル
    char szDummy[12]; // ダミー
    char szAddrCode[MAX_ADDRESSCODE_LEN]; // 住所コード
    char szAddress[MAX_ADDRESSNAME_LEN]; // 住所名称
    char szRestAddress[MAX_ADDRESS_LEN]; // 解析不可部分住所
    char szZipCode[MAX_ZIPCODE_LEN]; // 郵便番号
    char szLon[MAX_LONLAT_LEN]; // 経度(単位:度)
    char szLat[MAX_LONLAT_LEN]; // 緯度(単位:度)
    char szNode[MAX_NODECODE_LEN]; // ノードコード
} GeoPnt;

```

## (13) 到達圏／流入圏計算用パラメータ構造体 (768 バイト) 【V2.0 新規追加】

```
typedef struct tagCAParam {
    int    nDatum;                // 座標系
    int    nCoordinateFormat;    // 座標フォーマット
    int    nCalcKind;            // 計算方法(時間最短:1 距離最短:2)
    int    nNotUseHiway;        // 対応輸送手段 [注 (2) - 1]
    int    nPolygonKind;        // ポリゴン種別(列挙型 POLYGONKIND)
    char    szDummy1[12];        // ダミー
    GeoPnt StartGeoPnt;          // GeoPnt 構造体
    int    nAreaKind;            // 時間圏/距離圏指定(時間圏:1 距離圏:2)
    int    nAreaRange;          // 範囲指定 (単位: 0.1 秒 or メートル) (マイナス値は FULL 計算)
    int    nPolygonLevel;       // ポリゴンレベル (-1:凸型 0-20:凹凸型)
    int    nDonutPolygonLevel;  // ドーナツポリゴンレベル (-2:なし -1:凸型 0-20:凹凸型)
    int    nAreaDirection;      // 流入圏計算フラグ
    char    szDummy2[12];        // ダミー
} CAParam;
```

## (14) 片道一括計算用パラメータ構造体 (736 バイト) 【V2.0 新規追加】

```
typedef struct tagCOWParam {
    int    nDatum;                // 座標系
    int    nCoordinateFormat;    // 座標フォーマット
    int    nCalcKind;            // 計算方法(時間最短:1 距離最短:2)
    int    nNotUseHiway;        // 対応輸送手段 [注 (2) - 1]
    GeoPnt StartGeoPnt;          // GeoPnt 構造体
    int    nAreaKind;            // 時間圏/距離圏指定(時間圏:1 距離圏:2)
    int    nAreaRange;          // 範囲指定 (単位: 0.1 秒 or メートル) (マイナス値は FULL 計算)
    int    nAreaDirection;      // 流入圏計算フラグ
    char    szDummy[4];          // ダミー
} COWParam;
```

## (15) 到達圏／流入圏計算用パラメータ構造体 2 (768 バイト) 【V2.1 新規追加】

```
typedef struct tagCA2Param {
    int    nDatum;                // 座標系
    int    nCoordinateFormat;    // 座標フォーマット
    int    nCalcKind;            // 計算方法(時間最短:1 距離最短:2)
    int    nNotUseHiway;        // 対応輸送手段 [注(2) - 1]
    int    nPolygonKind;        // ポリゴン種別(列挙型 POLYGONKIND)
    char szDummy1[12];          // ダミー
    GeoPnt StartGeoPnt;         // GeoPnt 構造体
    int    nAreaKind;            // 時間圏/距離圏指定(時間圏:1 距離圏:2)
    int    nAreaRange;          // 範囲指定 (単位: 0.1 秒 or メートル) (マイナス値は FULL 計算)
    int    nPolygonLevel;       // ポリゴンレベル (-1:凸型 0-20:凹凸型)
    int    nDonutPolygonLevel;  // ドーナツポリゴンレベル (-2:なし -1:凸型 0-20:凹凸型)
    int    nAreaDirection;      // 流入圏計算フラグ
    int    nPolygonDetail;      // ポリゴン詳細フラグ
    char szDummy2[8];          // ダミー
} CA2Param;
```

## (16) 片道一括計算用パラメータ構造体 2 (752 バイト) 【V2.1 新規追加】

```
typedef struct tagCOW2Param {
    int    nDatum;                // 座標系
    int    nCoordinateFormat;    // 座標フォーマット
    int    nCalcKind;            // 計算方法(時間最短:1 距離最短:2)
    int    nNotUseHiway;        // 対応輸送手段 [注(2) - 1]
    GeoPnt StartGeoPnt;         // GeoPnt 構造体
    int    nAreaKind;            // 時間圏/距離圏指定(時間圏:1 距離圏:2)
    int    nAreaRange;          // 範囲指定 (単位: 0.1 秒 or メートル) (マイナス値は FULL 計算)
    int    nAreaDirection;      // 流入圏計算フラグ
    int    nHighwayOutput;      // 高速区間計算フラグ
    int    nTollOutput;         // 通行料金計算フラグ
    char szDummy[12];          // ダミー
} COW2Param;
```

## (17) ジオロケーション構造体 (880 バイト) 【V2.0 新規追加】

```

typedef struct tagGeoLoc {
    char szSearchString[MAX_ADDRESS_LEN];           // 検索文字列
    int nGeoLevel;                                   // 検索住所レベル
    char szDummy1[12];                               // ダミー
    char szAddrCode[MAX_ADDRESSCODE_LEN];           // 住所コード
    char szAddress[MAX_ADDRESSNAME_LEN];            // 住所名称
    char szRestAddress[MAX_ADDRESS_LEN];            // 解析不可部分住所
    char szZipCode[MAX_ZIPCODE_LEN];               // 郵便番号
    char szNode[MAX_NODECODE_LEN];                 // ノード
    char szLon[MAX_LONLAT_LEN];                    // 経度(単位:度)
    char szLat[MAX_LONLAT_LEN];                    // 緯度(単位:度)
    int nTime;                                       // 所要時間(分)
    int nDSec;                                       // 所要時間(0.1 秒)
    int nDist;                                       // 道のり(m)
    int nHighwayTime;                               // 高速所要時間(分)
    int nHighwayDSec;                               // 高速所要時間(0.1 秒)
    int nHighwayDist;                               // 高速道のり(m)
    int nToll_SS;                                    // 通行料金(二輪・軽)
    int nToll_S;                                     // 通行料金(普通)
    int nToll_M;                                     // 通行料金(中型)
    int nToll_L;                                     // 通行料金(大型)
    int nToll_LL;                                    // 通行料金(特大)
    char szDummy2[4];                               // ダミー
    char szTag[MAX_TAG_LEN];                        // タグ
} GeoLoc;

```

## (18) 住所検索共通パラメータ構造体 (16 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoCommonParam {
    int nDatum;                                     // Total:16bytes
    int nCoordinateFormat;                         // 4:測地系
    int nSearchFlag;                               // 4:座標フォーマット
    char szDummy[4];                               // 4:サーチフラグ(主に郵便番号検索で使用)
    // 4:ダミー
} GeoCommonParam;

```

## (19) 住所検索アイテム構造体 (1248 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoSearchAddressItem {
    char szSearchString[MAX_ADDRESS_LEN];           // Total: 1248bytes
    int nAddressLevel;                             // 256:検索文字列
    char szDummy[4];                               // 4:住所レベル
    char szAddressCode[MAX_ADDRESSCODE_LEN];       // 4:ダミー
    char szAnalyzedAddress[MAX_ADDRESS_LEN];       // 24:住所コード
    char szRestAddress[MAX_ADDRESS_LEN];           // 256:解析住所名称
    char szLon[MAX_LONLAT_LEN];                    // 256:解析不可部分住所
    char szLat[MAX_LONLAT_LEN];                    // 32:経度(単位:度)
    char szFullAddress[MAX_ADDRESS_LEN];           // 32:緯度(単位:度)
    char szTag[MAX_TAG_LEN];                       // 256:全住所
    // 128:タグ
} GeoSearchAddressItem;

```



## (20) 住所アイテム構造体 (504 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoAddressItem {
    int nResultLevel; // Total: 504bytes // 4:検索住所レベル
    int nUpperLevel; // 4:上の住所レベル
    int nLowerLevel; // 4:下の住所レベル
    int szDummy; // 4:ダミー
    char szAddressCode[MAX_ADDRESSCODE_LEN]; // 24:住所コード
    char szAddress[MAX_ADDRESS_LEN]; // 256:住所
    char szResultKana[MAX_KANANAME_LEN]; // 64:ひらがな住所
    char szResultKataKana[MAX_KANANAME_LEN]; // 64:カタカナ住所
    char szResultZip[MAX_ZIPCODE_LEN]; // 16:郵便番号
    char szLon[MAX_LONLAT_LEN]; // 32:経度(単位:度)
    char szLat[MAX_LONLAT_LEN]; // 32:緯度(単位:度)
} GeoAddressItem;

```

## (21) 住所コード検索アイテム構造体 (664 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoSearchAddressCodeItem {
    char szSearchCodeString[MAX_ADDRESSCODE_LEN]; // Total: 664bytes // 24:検索住所コード
    char szDummy[8]; // 8:ダミー
    GeoAddressItem geoAddressItem; // 504:住所アイテム
    char szTag[MAX_TAG_LEN]; // 128:タグ
} GeoSearchAddressCodeItem;

```

## (22) 全住所取得アイテム構造体 (1984 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoCreateFullAddressByCodeItem {
    char szSearchCodeString[MAX_ADDRESSCODE_LEN]; // Total: 1984bytes // 24:検索住所コード
    int nAddressLevel; // 4:住所レベル
    char szDummy[4]; // 4:ダミー
    char szAddressCode[MAX_ADDRESSCODE_LEN]; // 24:住所コード
    char szDummy2[8]; // 8:ダミー
    char szResultFullAddress[MAX_ADDRESS_LEN]; // 256:全住所
    char szResultPref[MAX_ADDRESS_LEN]; // 256:都道府県
    char szResultCity[MAX_ADDRESS_LEN]; // 256:市区町村
    char szResultStreet[MAX_ADDRESS_LEN]; // 256:大字丁目
    char szResultBlock[MAX_ADDRESS_LEN]; // 256:番地
    char szResultHouse[MAX_ADDRESS_LEN]; // 256:号
    char szResultOther[MAX_ADDRESS_LEN]; // 256:その他
    char szTag[MAX_TAG_LEN]; // 128:タグ
} GeoCreateFullAddressByCodeItem;

```

## (23) 最寄住所取得アイテム構造体 (968 バイト) 【V2.4 新規追加】

```

typedef struct tagGeoNearestAddressItem {
    char szLon[MAX_LONLAT_LEN]; // Total: 968bytes // 32:経度(単位:度)
    char szLat[MAX_LONLAT_LEN]; // 32:緯度(単位:度)
    char szFullAddress[MAX_ADDRESS_LEN]; // 256:全住所
    GeoAddressItem geoAddressItem; // 504:住所アイテム
    double nDistance; // 8:距離
    char szDummy[8]; // 8:ダミー
    char szTag[MAX_TAG_LEN]; // 128:タグ
} GeoNearestAddressItem;

```

## ( 2 4 ) 郵便番号検索アイテム構造体 (496 バイト) 【V2.4 新規追加】

```
typedef struct tagGeoSearchZipItem {                // Total: 496bytes
    char szZip[MAX_ZIPCODE_LEN];                    // 16:郵便番号
    char szAddressCode[MAX_ADDRESSCODE_LEN];        // 24:住所コード
    char szDummy[8];                                // 8:ダミー
    char szFullAddress[MAX_ADDRESS_LEN];             // 256:全住所名称
    char szLon[MAX_LONLAT_LEN];                     // 32:経度(単位:度)
    char szLat[MAX_LONLAT_LEN];                     // 32:緯度(単位:度)
    char szTag[MAX_TAG_LEN];                         // 128:タグ
} GeoSearchZipItem;
```

## ( 2 5 ) 郵便番号アイテム構造体 (368 バイト) 【V2.4 新規追加】

```
typedef struct tagGeoZipItem {                      // Total: 368bytes
    int nAddressLevel;                              // 4:住所レベル
    char szDummy[4];                                // 4:ダミー
    char szAddressCode[MAX_ADDRESSCODE_LEN];        // 24:住所コード
    char szFullAddress[MAX_ADDRESS_LEN];            // 256:全住所名称
    char szZip[MAX_ZIPCODE_LEN];                    // 16:郵便番号
    char szLon[MAX_LONLAT_LEN];                     // 32:経度(単位:度)
    char szLat[MAX_LONLAT_LEN];                     // 32:緯度(単位:度)
} GeoZipItem;
```

## 7. 7 サンプルコード

Visual Basic Ver.6 でのサンプルコードを示します。

## (1) ルート計算

```
' 東京都庁から大阪府庁までのルート計算を行うサンプル
Dim Environ As DCWSInc.Environ      ' 環境設定構造体
Dim Param As DCWSInc.Param          ' パラメータ構造体
Dim Loc As DCWSInc.Loc              ' ロケーション構造体
Dim NWInfo As DCWSInc.NWInfo        ' 計算用道路データ情報構造体
Dim ResultMsg As String              ' 距離計算結果文字列
Dim nRet As Integer                 ' ブリッジ関数の戻り値

' ユーザ ID、パスワード設定
Environ.szUserID = " UserID "
Environ.szUUID = " Password "
Environ.szHWUID = ""
Environ.szHost = ""
Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
nRet = ACT_DCWS_SetEnviron(Environ)

' 計算用道路データ情報取得サービス呼び出し
MsgBox ("計算用道路データ数：" & ACT_DCWS_GetUsableNWInfosCount())

' 距離計算サーバ取得サービス呼び出し
nRet = ACT_DCWS_GetUsableNWInfo(NWInfo, 0) ' 最初の計算用道路データを指定
nRet = ACT_DCWS_GetUsableDCServer(NWInfo)
nRet = ACT_DCWS_GetEnviron(Environ)
MsgBox ("距離計算サーバ エンドポイント URL:" & Environ.szWSUrl)
MsgBox ("距離計算サーバ IP:" & Environ.szHost)
MsgBox ("距離計算サーバ Port:" & Environ.nPort)

' パラメータ（時間最短、高速使用する、簡易ルート出力）を設定
Param.nCalcKind = CALCKIND_TIME
Param.nNotUseHiway = USEHIGHWAY_USE
Param.nPntsOutput = PNTSOUTPUT_OUTPUT
Param.nCalcRoute_RouteKind = ROUTEKIND_SIMPLE
nRet = ACT_DCWS_SetParam(Param)

' ロケーションを設定
nRet = ACT_DCWS_ClearLocs()
Loc.szLon = "139.695000"
Loc.szLat = "35.686389"
Loc.szTag = "東京都庁"
nRet = ACT_DCWS_AddLoc(Loc)
Loc.szLon = "135.522500"
Loc.szLat = "34.683056"
Loc.szTag = "大阪府庁"
nRet = ACT_DCWS_AddLoc(Loc)

' 距離計算サービス呼び出し
ResultMsg = Space$(MAX_RESULTMSG_LEN)
nRet = ACT_DCWS_Invoke(COMMAND_CALCROUTE, ResultMsg)
nRet = ACT_DCWS_GetLoc(Loc, 1)
MsgBox ("所要時間(分):" & Loc.nTime & _
        " 道のり(m):" & Loc.nDist & _
        " 走行ルートのポイント数:" & ACT_DCWS_GetPntsCount())
```

## (2) 流入圏計算

```

' 東京都庁までの流入圏計算とポリゴン取得を行うサンプル
Dim Environ As DCWSInc.Environ      ' 環境設定構造体
Dim CA2Param As DCWSInc.CA2Param    ' 到達圏／流入圏パラメータ構造体
Dim NWInfo As DCWSInc.NWInfo        ' 計算用道路データ情報構造体
Dim ResultMsg As String              ' 距離計算結果文字列
Dim nRet As Integer                  ' ブリッジ関数の戻り値

' ユーザ ID、パスワード設定
Environ.szUserID = " UserID "
Environ.szUUID = " Password "
Environ.szHWUID = ""
Environ.szHost = ""
Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asm"
nRet = ACT_DCWS_SetEnviron(Environ)

' 計算用道路データ情報取得サービス呼び出し
MsgBox("計算用道路データ数 : " & ACT_DCWS_GetUsableNWInfosCount())

' 距離計算サーバ取得サービス呼び出し
nRet = ACT_DCWS_GetUsableNWInfo(NWInfo, 0) ' 最初の計算用道路データを指定
nRet = ACT_DCWS_GetUsableDCServer(NWInfo)
nRet = ACT_DCWS_GetEnviron(Environ)
MsgBox ("距離計算サーバ エンドポイント URL:" & Environ.szWSUrl)
MsgBox ("距離計算サーバ IP:" & Environ.szHost)
MsgBox ("距離計算サーバ Port:" & Environ.nPort)

' パラメータ（時間最短、高速使用する、流入圏計算、90 分圏、時間圏、
' 単一ポリゴン、ポリゴンレベル 3、）を設定
CA2Param.nCalcKind = CALCKIND_TIME
CA2Param.nNotUseHiway = USEHIGHWAY_USE
CA2Param.nAreaDirection = AREADIRECTION_INFLOW
CA2Param.nAreaRange = CLng (90) * 60 * 10
CA2Param.nAreaKind = AREAKIND_TIME
CA2Param.nPolygonKind = POLYGONKIND_SINGLE
CA2Param.nPolygonLevel = 3
CA2Param.nDonutPolygonLevel = -2
CA2Param.nPolygonDetail = 0
CA2Param.StartGeoPnt.szLon = "139.695000"
CA2Param.StartGeoPnt.szLat = "35.686389"
nRet = ACT_DCWS_SetCA2Param(CA2Param)

' 距離計算サービス呼び出し
ResultMsg = Space(MAX_RESULTMSG_LEN)
nRet = ACT_DCWS_CalcArea20
MsgBox("流入圏ポリゴンのポイント数:" & ACT_DCWS_GetPntsCount())

```

## (3) 片道一括計算

```

' 神奈川、埼玉、千葉県庁から東京都庁までの片道一括計算を行うサンプル
Dim Environ As DCWSInc.Environ          ' 環境設定構造体
Dim COW2Param As DCWSInc.COW2Param      ' 片道一括計算パラメータ構造体
Dim GeoLoc As DCWSInc.GeoLoc            ' ジオロケーション構造体
Dim NWInfo As DCWSInc.NWInfo            ' 計算用道路データ情報構造体
Dim ResultMsg As String                  ' 距離計算結果文字列
Dim nRet As Integer                      ' ブリッジ関数の戻り値

' ユーザ ID、パスワード設定
Environ.szUserID = " UserID "
Environ.szUUID = " Password "
Environ.szHWUID = ""
Environ.szHost = ""
Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asm"
nRet = ACT_DCWS_SetEnviron(Environ)

' 計算用道路データ情報取得サービス呼び出し
MsgBox("計算用道路データ数: " & ACT_DCWS_GetUsableNWInfosCount())

' 距離計算サーバ取得サービス呼び出し
nRet = ACT_DCWS_GetUsableNWInfo(NWInfo, 0) ' 最初の計算用道路データを指定
nRet = ACT_DCWS_GetUsableDCServer(NWInfo)
nRet = ACT_DCWS_GetEnviron(Environ)
MsgBox("距離計算サーバ エンドポイント URL:" & Environ.szWSUrl)
MsgBox("距離計算サーバ IP:" & Environ.szHost)
MsgBox("距離計算サーバ Port:" & Environ.nPort)

' パラメータ（時間最短、高速使用する、時間圏、
'           90 分圏、流入圏計算）を設定
COW2Param.nCalcKind = CALCKIND_TIME
COW2Param.nNotUseHiway = USEHIGHWAY_USE
COW2Param.nAreaDirection = AREADIRECTION_INFLOW
COW2Param.nAreaKind = AREAKIND_TIME
COW2Param.nAreaRange = CLng(90) * 60 * 10
COW2Param.StartGeoPnt.szLon = "139.695000"
COW2Param.StartGeoPnt.szLat = "35.686389"
COW2Param.nHighwayOutput = 0
COW2Param.nTollOutput = 0
nRet = ACT_DCWS_SetCOW2Param(COW2Param)

' ロケーションを設定
nRet = ACT_DCWS_ClearLocs()
GeoLoc.szLon = "139.645833"
GeoLoc.szLat = "35.444722"
GeoLoc.szTag = "神奈川県庁"
nRet = ACT_DCWS_AddGeoLoc(GeoLoc)
GeoLoc.szLon = "139.652222"
GeoLoc.szLat = "35.854167"
GeoLoc.szTag = "埼玉県庁"
nRet = ACT_DCWS_AddGeoLoc(GeoLoc)
GeoLoc.szLon = "140.126667"
GeoLoc.szLat = "35.601389"
GeoLoc.szTag = "千葉県庁"
nRet = ACT_DCWS_AddGeoLoc(GeoLoc)

' 距離計算サービス呼び出し
ResultMsg = Space(MAX_RESULTMSG_LEN)
nRet = ACT_DCWS_CalcOneWay2()
nRet = ACT_DCWS_GetGeoLoc(GeoLoc, 0)
MsgBox (Left(GeoLoc.szTag, InStr(GeoLoc.szTag, vbNullChar) - 1)
        & "から都庁まで 所要時間(分):" & GeoLoc.nTime & " 道のり(m):" & GeoLoc.nDist)
nRet = ACT_DCWS_GetGeoLoc(GeoLoc, 1)
MsgBox (Left(GeoLoc.szTag, InStr(GeoLoc.szTag, vbNullChar) - 1)
        & "から都庁まで 所要時間(分):" & GeoLoc.nTime & " 道のり(m):" & GeoLoc.nDist)
nRet = ACT_DCWS_GetGeoLoc(GeoLoc, 2)
MsgBox (Left(GeoLoc.szTag, InStr(GeoLoc.szTag, vbNullChar) - 1)
        & "から都庁まで 所要時間(分):" & GeoLoc.nTime & " 道のり(m):" & GeoLoc.nDist)

```

## (4) 住所検索

' ACT 住所 "東京都港区新橋 3-7-4" を住所検索し経緯度を表示するサンプル

Dim Environ As Environ ' 環境設定構造体

Dim nRet As Integer ' ブリッジ関数の戻り値

Dim geoCommonParam As GeoCommonParam

Dim geoSearchAddressItem(1) As GeoSearchAddressItem

Dim geoInfo As geoInfo

' ユーザ ID、パスワード設定

Environ.szUserID = "UserID"

Environ.szUUID = "Password"

Environ.szHWUID = ""

Environ.szHost = ""

Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"

nRet = ACT\_DCWS\_SetGeoEnviron(Environ)

' 位置検索サーバ取得サービス呼び出し

nRet = ACT\_DCWS\_GetUsableGeoServer

' 位置検索サーバ情報取得

nRet = ACT\_DCWS\_GetGeoEnviron(Environ)

' パラメータを設定

geoCommonParam.nDatum = 0

geoCommonParam.nCoordinateFormat = 0

' 住所アイテムを設定

geoSearchAddressItem(0).szSearchString = "東京都港区新橋 3-7-4"

' 住所検索サービス呼び出し

```
nRet = ACT_GCWS_SearchAddress(Environ,
                                geoCommonParam,
                                ADDRLEVEL_HOUSE,
                                1,
                                geoSearchAddressItem(0),
                                geoInfo)
```

' 検索結果表示

MsgBox ("検索住所：" & geoSearchAddressItem(0).szAnalyzedAddress)

MsgBox ("経度：" & geoSearchAddressItem(0).szLon)

MsgBox ("緯度：" & geoSearchAddressItem(0).szLat)

MsgBox ("住所コード：" & geoSearchAddressItem(0).szAddressCode)

## (5) 下位住所列挙

```

' 東京都の市区町村を列挙するサンプル
Dim Environ As Environ ' 環境設定構造体
Dim nRet As Integer ' ブリッジ関数の戻り値
Dim geoCommonParam As GeoCommonParam
Dim geoInfo As geoInfo
Dim szSearchCodeString As String
Dim countLowerAddressByCode As Integer
Dim geoAddressItem() As GeoAddressItem

' ユーザ ID、パスワード設定
Environ.szUserID = "UserID"
Environ.szUUID = "Password"
Environ.szHWUID = ""
Environ.szHost = ""
Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
nRet = ACT_DCWS_SetGeoEnviron(Environ)

' 位置検索サーバ取得サービス呼び出し
nRet = ACT_DCWS_GetUsableGeoServer

' 位置検索サーバ情報取得
nRet = ACT_DCWS_GetGeoEnviron(Environ)

' パラメータを設定
geoCommonParam.nDatum = 0
geoCommonParam.nCoordinateFormat = 0

' 住所コードを設定
szSearchCodeString = "13"

' 下位住所数取得
nRet = ACT_GCWS_CountLowerAddressByCode(Environ,
                                         geoCommonParam,
                                         szSearchCodeString,
                                         countLowerAddressByCode,
                                         geoInfo)

' 必要ワーク確保
ReDim geoAddressItem(countLowerAddressByCode)

' 下位住所数取得
nRet = ACT_GCWS_EnumLowerAddressByCode(Environ,
                                         geoCommonParam,
                                         szSearchCodeString,
                                         countLowerAddressByCode,
                                         geoAddressItem, geoInfo)

' 検索結果表示
MsgBox(" 下位住所数 : " & countLowerAddressByCode)
MsgBox(" 検索住所 : " & geoAddressItem(0).szAddress)

```

## (6) 最寄住所取得

```

' 東京都庁(139.695000, 35.686389)を検索し最寄住所と距離を表示するサンプル
Dim Environ As Environ ' 環境設定構造体
Dim nRet As Integer ' ブリッジ関数の戻り値
Dim geoCommonParam As GeoCommonParam
Dim geoInfo As geoInfo
Dim geoNearestAddressItem As GeoNearestAddressItem
Dim countNearestAddressItem As Integer

' ユーザ ID、パスワード設定
Environ.szUserID = "UserID"
Environ.szUUID = "Password"
Environ.szHWUID = ""
Environ.szHost = ""
Environ.szWSUrl = "http://distcalc.act-inc.co.jp/dcadmin/admin.asmx"
nRet = ACT_DCWS_SetGeoEnviron(Environ)

' 位置検索サーバ取得サービス呼び出し
nRet = ACT_DCWS_GetUsableGeoServer

' 位置検索サーバ情報取得
nRet = ACT_DCWS_GetGeoEnviron(Environ)

' パラメータを設定
geoCommonParam.nDatum = 0
geoCommonParam.nCoordinateFormat = 0

' 住所アイテムを設定
countNearestAddressItem = 1
geoNearestAddressItem.szLon = "139.695000"
geoNearestAddressItem.szLat = "35.686389"

' 最寄住所取得サービス呼び出し
nRet = ACT_GCWS_GetNearestAddress(Environ,
                                   geoCommonParam,
                                   countNearestAddressItem,
                                   GeoNearestAddressItem,
                                   geoInfo)

' 検索結果表示
MsgBox("検索住所 : " & geoNearestAddressItem.szFullAddress)
MsgBox("検索経度 : " & geoNearestAddressItem.geoAddressItem.szLon)
MsgBox("検索緯度 : " & geoNearestAddressItem.geoAddressItem.szLat)
MsgBox("距離 : " & geoNearestAddressItem.nDistance)

```



## 7. 8 エラーコード一覧

No	カテゴリ	エラーコード	エラーメッセージ
1	管理サーバ	1051	GetUsableNWInfosCount 実行中に例外が発生しました。
2		1061	GetUsableNWInfo 実行中に例外[%s]が発生しました
3		1071	GetUsableDCServer 実行中に例外[%s]が発生しました
4	距離計算 サーバ	1101	Invoke コマンドが設定されていません
5		1111	Invoke に失敗しました
6		1112	Invoke 実行中に例外[%s]が発生しました
7		1201	Environment 取得に失敗しました
8		1202	Environment 設定に失敗しました
9		1311	Parameter 取得に失敗しました
10		1312	Parameter 設定に失敗しました
11		1401	Locs 取得に失敗しました
12		1402	Locs 設定に失敗しました
13		1403	Locs クリアに失敗しました
14		1411	Loc 取得に失敗しました
15		1412	Loc がありません
16		1413	Index に対応する Loc がありません
17		1421	Loc 追加に失敗しました
18		1501	Pnts 取得に失敗しました
19		1511	Pnt 取得に失敗しました
20		1512	Pnt がありません
21		1513	Index に対応する Pnt がありません
22		1601	Rns 取得に失敗しました
23		1611	Rn 取得に失敗しました
24		1612	Rn がありません
25		1613	Index に対応する Rn がありません
26		1701	Trds 取得に失敗しました
27		1711	Trd 取得に失敗しました
28		1712	Trd がありません
29		1713	Index に対応する Trd がありません
30		1801	NWInfo 取得に失敗しました
31		1811	NWInfo 取得に失敗しました
32		1901	Locs 保存がキャンセルされました
33		1902	Locs 保存ファイルオープンに失敗しました
34		1903	Locs 保存ファイル書き込みに失敗しました
35		1911	Loc 保存がキャンセルされました

36		1912	Loc 保存ファイルオープンに失敗しました
37		1913	Loc 保存ファイル書き込みに失敗しました
38		1921	Loc 読込ファイルが存在しません
39		1922	Loc 読込ファイルオープンに失敗しました
40		1923	Loc 読込ファイルからの読込みに失敗しました
41		1924	Loc 読込ファイルの%d 行目のカラム数が足りません
42	住所検索 サーバ	2001	GetUsableGeoServer 実行中に例外[%s]が発生しました
43		2101	Invoke コマンドが設定されていません
44		2111	Invoke に失敗しました
45		2112	GeoInvoke 実行中に例外[%s]が発生しました
46		2201	Environment (住所検索用) 取得に失敗しました
47		2202	Environment (住所検索用) 取得に失敗しました
48		2301	Parameter (住所検索用) 取得に失敗しました
49		2302	Parameter (住所検索用) 設定に失敗しました
50		2401	GeoItems 取得に失敗しました
51		2402	GeoItems 設定に失敗しました
52		2403	GeoItems クリアに失敗しました
53		2411	GeoItem 取得に失敗しました
54		2412	GeoItem がありません
55		2413	Index に対応する GeoItem がありません
56		2421	GeoItem 追加に失敗しました
57		2500	SearchAddress に失敗しました
58		2501	SearchAddressCode に失敗しました
59		2502	CreateFullAddressByCode に失敗しました
60		2503	CountLowerAddressByCode に失敗しました
61		2504	EnumLowerAddressByCode に失敗しました
62		2505	EnumLowerAddressByCode で引数 GeoAddressItem 数が足りません
63		2506	GetNearestAddress に失敗しました
64		2507	SearchZip に失敗しました
65		2508	CountAddressByZip に失敗しました
66		2509	EnumAddressByZip に失敗しました
67		2510	EnumAddressByZip で引数 GeoAddressItem 数が足りません
68		2901	GeoInfo 取得に失敗しました
69		2911	GeoInfo がありません
70	管理サーバ Ver2	3001	User Infomation 取得に失敗しました

## ブリッジ DLL を利用したプログラミング

71	距離計算	4001	CalcArea に失敗しました
72	サーバ	4101	CalcOneWay に失敗しました
73	Ver2	4102	CalcOneWay GeoLoc がありません
74		4011	CalcArea2 に失敗しました
75		4111	CalcOneWay2 に失敗しました
76		4112	CalcOneWay2 GeoLoc がありません
77		4301	CalcArea Parameter 取得に失敗しました
78		4302	CalcArea Parameter 設定に失敗しました
79		4311	CalcOneWay Parameter 取得に失敗しました
80		4312	CalcOneWay Parameter 設定に失敗しました
81		4321	CalcArea2 Parameter 取得に失敗しました
82		4322	CalcArea2 Parameter 設定に失敗しました
83		4331	CalcOneWay2 Parameter 取得に失敗しました
84		4332	CalcOneWay2 Parameter 設定に失敗しました
85		4401	GeoLocs 取得に失敗しまし
86		4402	GeoLocs 設定に失敗しました
87		4403	GeoLocs クリアに失敗しました
88		4411	GeoLoc 取得に失敗しました
89		4412	GeoLoc がありません
90		4413	Index に対応する GeoLoc がありません
91		4421	GeoLoc 追加に失敗しました
92		4501	DCWSMessage 取得に失敗しました
93		4502	GCWSMessage 取得に失敗しました
94	共通	5000	接続時エラー：URL が不正です
95	http 通信	5001	接続時エラー：セッションが不正です
96		5002	接続時エラー：接続が不正です
97		5003	接続時エラー：要求が不正です
98		5004	接続時エラー：コマンドが不正です
99		5006	接続時エラー：レスポンスが不正です
100		5007	接続時エラー：ヘッダーが不正です
101		5008	接続時エラー：結果読み込み時にエラーが発生しました
102		5010	サーバー接続に失敗しました。エラーコード：%d
103		5011	サーバー接続に失敗しました。プロキシ設定：Internet Explorer の設定。 エラーコード：%d
104		5012	サーバー接続に失敗しました。プロキシ設定：Internet Explorer の設定： 設定を自動的に検出する。エラーコード：%d
105		5013	サーバー接続に失敗しました。プロキシ設定：Internet Explorer の設定： 自動構成スクリプトを使用する。エラーコード：%d

106		5014	サーバー接続に失敗しました。プロキシ設定 : Internet Explorer の設定 : プロキシサーバーの設定。エラーコード : %d
107		5015	サーバー接続に失敗しました。プロキシ設定 : Internet Explorer の設定 : 設定を自動的に検出する。自動構成スクリプトを使用する。エラーコード : %d
108		5016	サーバー接続に失敗しました。プロキシ設定 : Internet Explorer の設定 : 設定を自動的に検出する。プロキシサーバーの設定。エラーコード : %d
109		5017	サーバー接続に失敗しました。プロキシ設定 : Internet Explorer の設定 : 自動構成スクリプトを使用する。プロキシサーバーの設定。エラーコード : %d
110		5018	サーバー接続に失敗しました。プロキシ設定 : Internet Explorer の設定 : 設定を自動的に検出する。自動構成スクリプトを使用する。プロキシサーバーの設定。エラーコード : %d
111		5019	サーバー接続に失敗しました。プロキシ設定 : 設定を自動的に検出 : エラーコード : %d
112		5020	サーバー接続に失敗しました。プロキシ設定 : 自動構成スクリプト : エラーコード : %d
113		5021	サーバー接続に失敗しました。プロキシ設定 : 指定されたプロキシサーバー : エラーコード : %d
114		5022	サーバー接続に失敗しました。プロキシ設定 : デフォルトプロキシ : エラーコード : %d
115		5100	プロキシ設定 : デフォルトプロキシの取得に失敗しました
116		5101	プロキシ設定 : 設定を自動的に取得に失敗しました
117		5102	プロキシ設定 : 自動構成スクリプトの取得に失敗しました
118		5103	プロキシ設定 : 指定されたプロキシサーバーの取得に失敗しました
119		5200	接続ステータスエラー : 接続に失敗しました。エラーコード : %d
120		5201	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。接続に失敗しました。エラーコード : %d
121		5202	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。設定を自動的に検出するに失敗しました。接続に失敗しました。エラーコード : %d
122		5203	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。自動構成スクリプトを使用するに失敗しました。接続に失敗しました。エラーコード : %d
123		5204	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。プロキシサーバーの設定に失敗しました。接続に失敗しました。エラーコード : %d
124		5205	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。設定を自動的に検出するに失敗しました。自動構成スクリプトを使用するに失敗しました。接続に失敗しました。エラーコード : %d

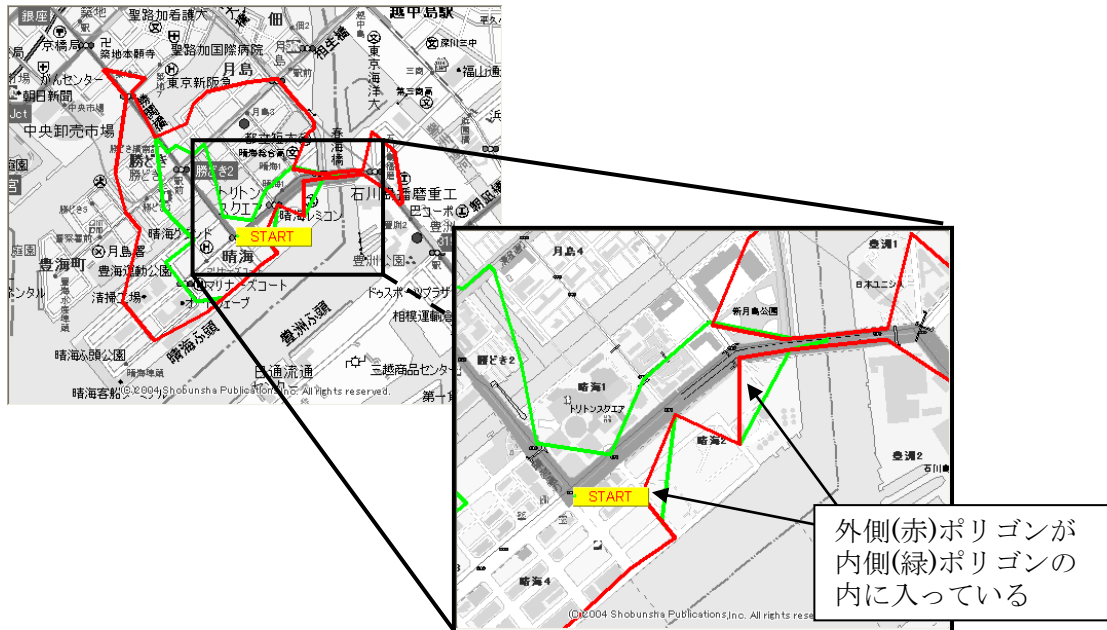
ブリッジ DLL を利用したプログラミング

125		5206	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。設定を自動的に検出するに失敗しました。プロキシサーバーの設定に失敗しました。接続に失敗しました。エラーコード : %d
126		5207	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。自動構成スクリプトを使用するに失敗しました。プロキシサーバーの設定に失敗しました。接続に失敗しました。エラーコード : %d
127		5208	接続ステータスエラー : Internet Explorer の設定取得に失敗しました。設定を自動的に検出するに失敗しました。自動構成スクリプトを使用するに失敗しました。プロキシサーバーの設定に失敗しました。接続に失敗しました。エラーコード : %d
128		5209	接続ステータスエラー:プロキシ設定:設定を自動的に検出。エラーコード:%d
129		5210	接続ステータスエラー:プロキシ設定:自動構成スクリプト。エラーコード:%d
130		5211	接続ステータスエラー:プロキシ設定:指定されたプロキシサーバー。エラーコード:%d
131		5212	接続ステータスエラー:プロキシ設定:デフォルトプロキシ。エラーコード:%d

## Appendix A : ポリゴン修正機能

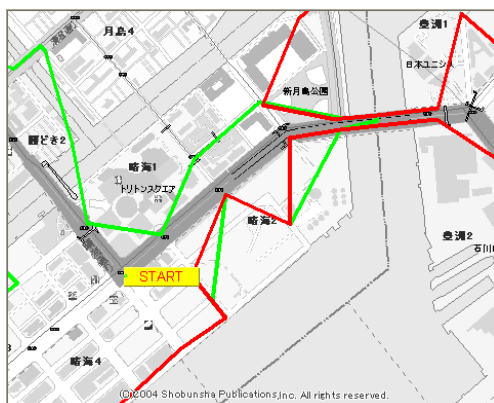
### (1) ポリゴン逆転現象

距離計算サービスの到達圏／流入圏計算では、パラメータの設定により非常に複雑なポリゴンを作成することができます。このため、同一地点からの到達範囲の異なるポリゴンを作成すると、下図のように外側のポリゴンの一部が内側ポリゴンの中に進入するという「ポリゴン逆転現象」が発生する場合があります。

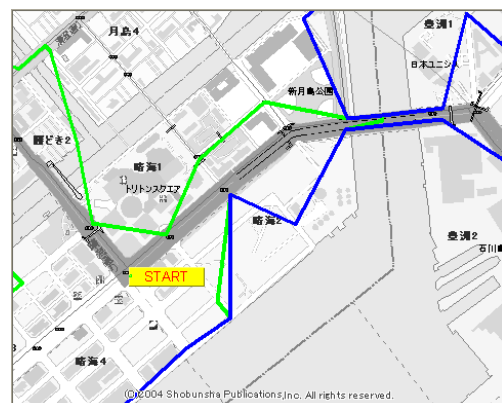


「**ACT** 距離計算サービス」では、ノード（交差点）毎に正確に距離計算を行っています。ポリゴンは、到達範囲の外側のノード（交差点）を一定の規則で選択し直線で結んで作成されるため、このような現象が発生します。この問題に対応するため「ポリゴン修正機能」が存在します。ポリゴン修正機能を使用すると下図右側の青線のような修正されたポリゴンを取得することが可能です。

【ポリゴン修正機能 Off 時】



【ポリゴン修正機能 On 時】



## (2) ポリゴン修正機能の使用方法

ポリゴン修正機能を使用するには、距離計算サービスを呼び出す際に使用する距離計算コマンド構造体 (DCCCommand) を下表のように設定してください。

項番	構造体	メンバ	設定内容
1	Param	RnsOutput	1(RNSOUTPUT.OUTPUT)を指定することによりポリゴン修正機能が有効になる
2	Pnts	—	内側のポリゴン情報 (前回計算した Pnts)

例えば 5 分圏、10 分圏、15 分圏のような連続したポリゴンを作成する場合、次のような手順で距離計算サービスを呼び出してください。

手順① Param.RnsOutput に 0、ポリゴン情報を設定せずに、5 分圏計算。

→出力として 5 分ポリゴンを取得

手順② Param.RnsOutput に 1、Pnts に①で得られた 5 分ポリゴンを設定し、10 分圏を計算

→出力として (5 分ポリゴンの内側に進入しない) 10 分ポリゴンを取得

手順③ Param.RnsOutput に 1、Pnts に②で得られた 10 分ポリゴンを設定し、15 分圏を計算

→出力として (10 分ポリゴンの内側に進入しない) 15 分ポリゴンを取得

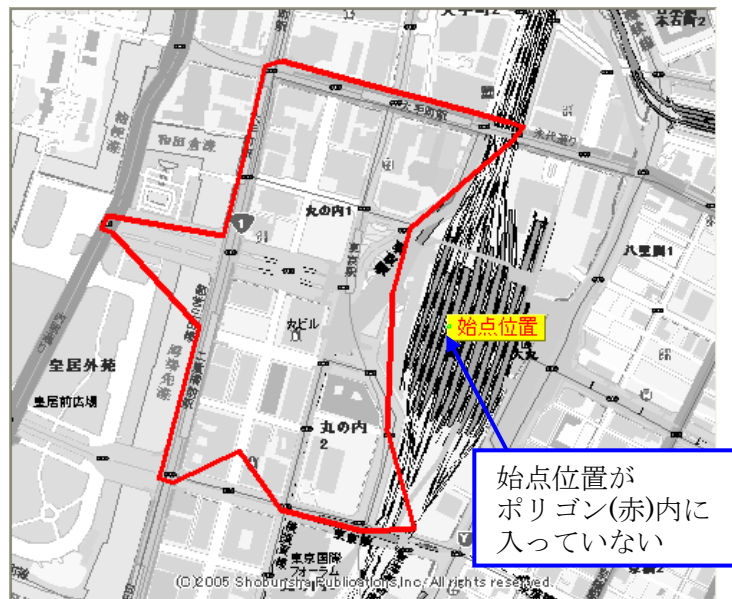
## (3) 注意事項

- Pnts に設定するポリゴンは、必ず内側のポリゴンを設定してください。(内側でない場合、ポリゴンが出力されない、または不正確なポリゴンを出力する場合があります)
- ポリゴン修正機能は、ポリゴン逆転現象を 100%抑制することを保証する機能ではありません。ポリゴン逆転現象を発生させたくない場合には、ポリゴンレベルを-1(凸型ポリゴン)に設定するようにしてください。

## Appendix B：ポリゴン始点包含機能

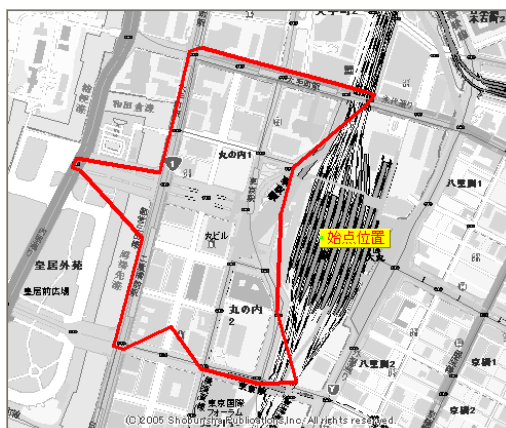
### (1) ポリゴン始点除外現象

距離計算サービスの到達圏／流入圏計算では、パラメータにより与えられた位置（経緯度）に最も近い交差点から到達圏／流入圏計算を行っています。駅周辺や湾岸地域など道路が少ない地域で、到達範囲の狭い計算を行うと下図のように与えられた位置（始点）を含まないポリゴンを作成する場合があります。到達圏／流入圏計算サービスでは必ず始点が含まれます。

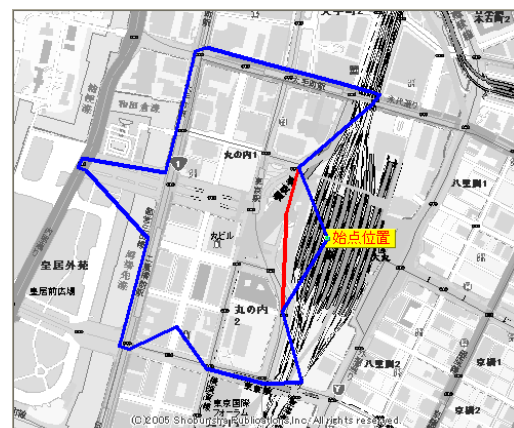


「**ACT** 距離計算サービス」では、ノード（交差点）毎に正確に距離計算を行っています。ポリゴンは、与えられた位置（始点）から最も近い交差点から計算し、作成されるため、このような現象が発生します。この問題に対応するため「ポリゴン始点包含機能」が存在します。ポリゴン始点包含機能を使用すると下図右側の青線のような修正されたポリゴンを取得することが可能です。

【ポリゴン始点包含機能 Off 時】



【ポリゴン始点包含機能 On 時】





(2) ポリゴン始点包含機能の使用方法

ポリゴン始点包含機能を使用するには、距離計算サービスを呼び出す際に使用する距離計算コマンド構造体 (DCCommand) を下表のように設定してください。

項番	構造体	メンバ	設定内容
1	Param	TrdsOutput	1(TRDSOUTPUT.OUTPUT)を指定することによりポリゴン始点包含機能が有効になる

以上