

ACT コンポーネントシリーズ

ACT 距離計算コンポーネント

Version 1.0

プログラマーズ

リファレンス

ACT コンポーネントシリーズ

ACT 距離計算コンポーネント Version 1.0 プログラマーズ・リファレンス

2001年05月07日 初版発行

2001年08月01日 第2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) 2001 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 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 距離計算コンポーネント プログラマーズリファレンス 目次

第 1 部 ACT 距離計算コンポーネントの概要

1 . はじめに	1
2 . コンポーネントの利用制限	2
3 . インストール/アンインストール	3
4 . 動作環境	2
5 . 開発環境	2
6 . オブジェクト一覧	3
7 . 未対応機能	3

第 2 部 オブジェクトリファレンス

1 . はじめに	5
1.1 プログラム ID	5
1.2 データ型	5
1.3 Visual Basic での使用	5
2 . DistCalc オブジェクト	7
2.1 概要	7
2.2 インターフェース一覧	7
2.3 プロパティ	10
2.4 メソッド	27
3 . ENUM 定数一覧	51
3.1 計算種別(ENUM_CALCKIND 型)	51
3.2 高速道路の使用 / 非使用(ENUM_USEHIGHWAY 型)	51
3.3 ルート情報種別(ENUM_ROUTEKIND 型)	51
3.4 到達圏種別(ENUM_RANGEKIND 型)	52
3.5 速度フラグ(ENUM_SPEED 型)	52
3.6 エラー番号(ERR_ACTCALCM 型)	53
4 . コーディング例	57
4.1 2 点間計算	57
4.2 最短ルート計算	58
4.3 到達圏計算	59

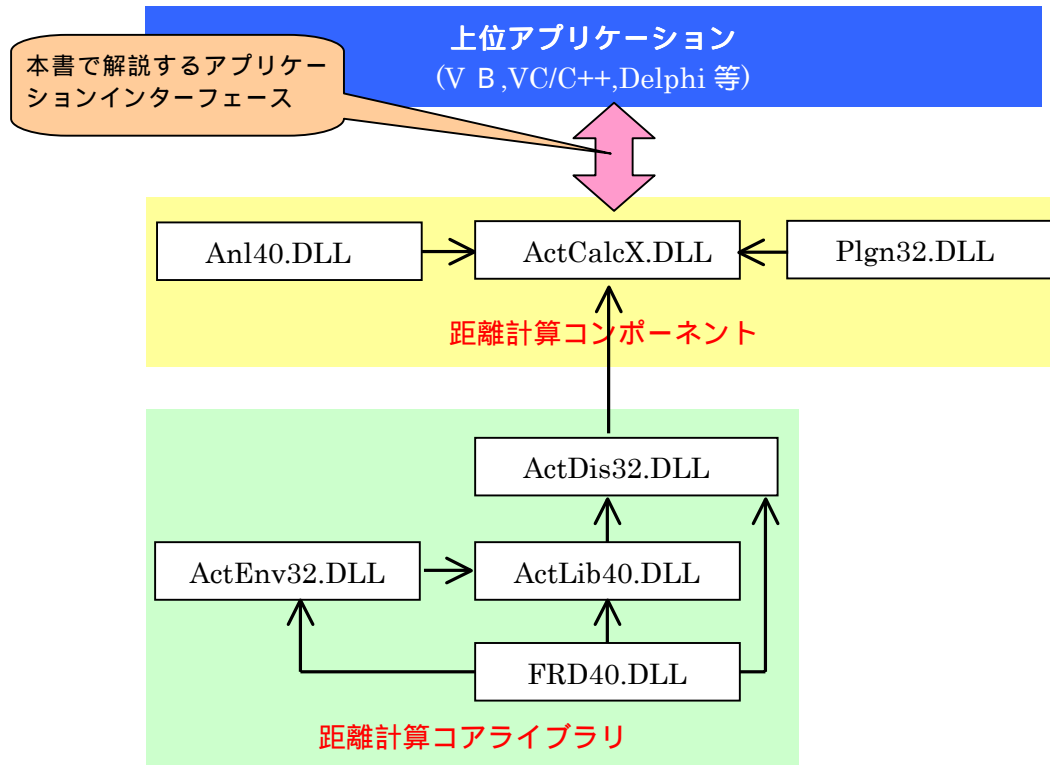
第1部 ACT 距離計算コンポーネントの概要

1. はじめに

ACT 距離計算コンポーネントは、距離計算を行うための COM (Component Object Model) コンポーネントです。COM を用いて開発が行える環境で容易に最短経路計算や到達圏計算を行うシステムを構築することが可能です。

ACT 距離計算コンポーネントは、下図の通り ACTCALCX.DLL、ANL40.DLL、PLGN32.DLL の3モジュールで構成され、距離計算コアライブラリの上位モジュールとして動作します。本書で解説するコンポーネントは、すべて ACTCALCX.DLL からエクスポートされています。

ACTCALCX.DLL が正常に機能するためには、これら3モジュールすべてと距離計算コアライブラリが正常に動作する必要があります。



2．コンポーネントの利用制限

ACT 距離計算コンポーネントは COMDLL 形式(*.DLL)で提供されます。提供されるモジュールをお客様が改変・変更することや、リバースエンジニアリング・逆コンパイル等を行うことはできません。また、これらのモジュールをアドバンスド・コア・テクノロジー株式会社との契約なしに再頒布及び販売することはできません。

3．インストール/アンインストール

ACT 距離計算コンポーネントは、単体でのインストール/アンインストールはできません。ACT 距離計算コンポーネントのインストール/アンインストールは、ACT 距離計算コンポーネントを含んでいる製品をインストール/アンインストールすることにより行います。

4．動作環境

ACT 距離計算コンポーネントは、次の OS で作動します。

- ・ Windows 95
- ・ Windows 98(SE を含む)
- ・ Windows NT 4.0(SP4 以上)
- ・ Windows 2000

いずれも IntelCPU のみ動作保証です。

5．開発環境

ACT 距離計算コンポーネントは、COMDLL 形式で提供されています。COM を用いて開発を行うことができる開発環境であれば、ACT 距離計算コンポーネントを用いたアプリケーションの開発が可能です。弊社では以下の開発環境でのみ動作確認を行っております。

- ・ Microsoft Visual Basic 6.0
- ・ Microsoft Visual C++ 6.0
- ・ Borland Delphi 5.0

6. オブジェクト一覧

以下は、ACT 距離計算コンポーネントのオブジェクト一覧です。

オブジェクト名	解説
DistCalc	計算用道路地図管理機能や速度変更機能を持った距離計算オブジェクト

7. 未対応機能

ACT 距離計算コンポーネントでは、次の機能を提供していません。これらの機能を使用する場合には、距離計算コアライブラリ(ActDis32.DLL)を使用してください。

- ・文字ルート情報
- ・区間距離表
- ・詳細型ポリゴン

第 2 部 オブジェクトリファレンス

1. はじめに

1.1 プログラム ID

ACT 距離計算コンポーネントの各オブジェクトのプログラム ID(ProgID)は、つぎのとおりです。

オブジェクト名	プログラム ID
DistCalc オブジェクト	ActCalcX.DistCalc

1.2 データ型

本書ではデータ型の記述に IDL 形式を用いています。使用する開発言語のデータ型に読み替えてください。

1.3 Visual Basic での使用

ACT 距離計算コンポーネントを Visual Basic で使用する場合、参照設定ダイアログで「ACT 距離計算コンポーネント」と表示されるライブラリファイルにチェックを付けてください。

2 . DistCalc オブジェクト

2 . 1 概要

DistCalc オブジェクトで距離計算を行うには、RoadNetworkDir プロパティと LoadRoadNetwork メソッドを使用して計算用道路データをロードする必要があります。計算用道路データをロードした後は、各計算用メソッドを使用して計算が可能になります。計算が終了し、オブジェクトを解放する前に UnloadRoadNetwork メソッドを使用して、計算用道路データをアンロードする必要があります。また、道路速度変更機能を有しています。

2 . 2 インタフェース一覧

番号	名称	解 説	Ver. 1.0	Ver. x.x	Ver. x.x
1	Active プロパティ	距離計算が可能な状態か取得します	New		
2	RoadNetworkDir プロパティ	計算用道路データフォルダを格納します	New		
3	FromNodeCode プロパティ	発地ノードコードを格納します	New		
4	ToNodeCode プロパティ	着地ノードコードを格納します	New		
5	CalcKind プロパティ	計算種別(時間最短 or 距離最短)を格納します	New		
6	UseHighway プロパティ	高速道路の使用/非使用を格納します	New		
7	Time プロパティ	所要時間(分単位)を取得します	New		
8	DSecTime プロパティ	所要時間(1/10 秒単位)を取得します	New		
9	Distance プロパティ	道のり(m 単位)を取得します	New		
10	Toll_SS プロパティ	軽・自動二輪の通行料金(円単位)を取得します	New		
11	Toll_S プロパティ	普通車の通行料金(円単位)を取得します	New		
12	Toll_M プロパティ	中型車の通行料金(円単位)を取得します	New		
13	Toll_L プロパティ	大型車の通行料金(円単位)を取得します	New		
14	Toll_LL プロパティ	特大車の通行料金(円単位)を取得します	New		
15	IsTollExist プロパティ	通行料金が存在するか取得します	New		

DistCalc オブジェクト

16	IsDetailExist プロパティ	詳細ルートデータが存在するか取得します	New		
17	LastError プロパティ	最後に発生したエラー番号を取得します	New		
18	LoadRoadNetwork メソッド	計算用道路データをロードします	New		
19	UnloadRoadNetwork メソッド	計算用道路データをアンロードします	New		
20	CalcRoute メソッド	2点間ルート計算を行います	New		
21	CalcRoute2 メソッド	ルート計算を行います	New		
22	CalcOptRoute メソッド	最短ルート計算を行います	New		
23	GetRouteXY メソッド	簡易ルート情報(経度緯度)を取得します	New		
24	GetRouteXYToMMF メソッド	簡易ルート情報(経度緯度)を取得し、メモリマップドファイルに格納します	New		
25	GetRouteDetail メソッド	詳細ルート情報(経度緯度)を取得します	New		
26	GetRouteDetailToMMF メソッド	詳細ルート情報(経度緯度)を取得し、メモリマップドファイルに格納します	New		
27	GetRoute メソッド	ルート情報(ノードコードまたはリンクコード)を取得します	New		
28	CalcArea メソッド	到達圏計算を行います	New		
29	MakePolygon メソッド	到達圏計算を行い、簡易型(凸型)ポリゴンを取得します	New		
30	MakePolygonToMMF メソッド	到達圏計算を行い、簡易型(凸型)ポリゴンを取得し、メモリマップドファイルに格納します	New		
31	MakePolygon2 メソッド	到達圏計算を行い、標準型(凹凸型)ポリゴンを取得します	New		
32	MakePolygon2ToMMF メソッド	到達圏計算を行い、標準型(凹凸型)ポリゴンを取得し、メモリマップドファイルに格納します	New		
33	GetNodeXY メソッド	ノードの位置情報(経度緯度)を取得します	New		
34	GetNearestNode メソッド	指定位置(経度緯度)の近傍ノードコードを取得します	New		
35	GetSpeed メソッド	リンクの速度情報(片方向)を取得します	New		
36	GetSpeedEx メソッド	リンクの速度情報(両方向)を取得します	New		

DistCalc オブジェクト

37	ChangeBothSpeedEx メソッド	リンクの速度情報(両方向)を設定します	New		
38	CountTempSpeed メソッド	一時的速度が設定されているリンク数を取得します	New		
39	CommitTempSpeed メソッド	一時的速度を恒久的速度として保存します	New		
40	ClearTempSpeed メソッド	一時的速度を取り消します	New		
41	ClearLastError メソッド	エラー番号をクリアします	New		

2.3 プロパティ

Active プロパティ

距離計算が可能な状態か取得します。

構文

object.Active[=*Value*]

項目	データ型	内容
戻り値	VARIANT_BOOL	距離計算が可能な状態の場合 TRUE、計算不可な状態の場合 FALSE
<i>Value</i>	VARIANT_BOOL	距離計算が可能な状態にする場合 TRUE、計算不可な状態にする場合 FALSE を指定します。

解説

Active プロパティに値を設定することは、LoadRoadNetwork メソッドまたは UnloadRoadNetwork メソッドを呼び出すことと同じです。

RoadNetworkDir プロパティ

計算対象とする計算用道路データが存在するフォルダを格納します。

構文

object.**RoadNetworkDir**[=*Value*]

項目	データ型	内容
戻り値	BSTR	計算用道路データが存在するフォルダを取得します。
<i>Value</i>	BSTR	計算用道路データが存在するフォルダを指定します。

FromNodeCode プロパティ

発地となるノードのノードコードを格納します。

構文

object.**FromNodeCode**[=*Value*]

項目	データ型	内容
戻り値	double	発地のノードコードを取得します。
<i>Value</i>	double	発地のノードコードを指定します。

ToNodeCode プロパティ

着地となるノードのノードコードを格納します。

構文

object.**ToNodeCode**[=*Value*]

項目	データ型	内容
戻り値	double	着地のノードコードを取得します。
<i>Value</i>	double	着地のノードコードを指定します。

CalcKind プロパティ

距離計算の計算種別(時間最短 or 距離最短)を格納します。

構文

object.**CalcKind**[= *Value*]

項目	データ型	内容
戻り値	ENUM_CALCKIND	計算種別(時間最短 or 距離最短)を取得します。
<i>Value</i>	ENUM_CALCKIND	計算種別(時間最短 or 距離最短)を指定します。

解説

ENUM_CALCKIND 型は、ENUM 定数一覧を参照してください。

UseHighway プロパティ

高速道路の使用 / 非使用を格納します。

構文

object.**UseHighway**[= *Value*]

項目	データ型	内容
戻り値	ENUM_USEHIGHWAY	高速道路の使用 / 非使用を取得します。
<i>Value</i>	ENUM_USEHIGHWAY	高速道路の使用 / 非使用を設定します。

解説

ENUM_USEHIGHWAY 型は、ENUM 定数一覧を参照してください。

Time プロパティ

所要時間(分単位)を取得します。値の設定は出来ません。

構文

object.**Time**

項目	データ型	内容
戻り値	long	所要時間(分単位)を取得します。

解説

Time プロパティは、ToNodeCode プロパティで設定されているノードまでの所要時間(分単位)を取得します。

DSecTime プロパティ

所要時間(1/10 秒単位)を取得します。値の設定は出来ません。

構文

object.**DSecTime**

項目	データ型	内容
戻り値	long	所要時間(1/10 秒単位)を取得します。

解説

DSecTime プロパティは、ToNodeCode プロパティで設定されているノードまでの所要時間(1/10 秒単位)を取得します。

Distance プロパティ

道のり(m 単位)を取得します。値の設定は出来ません。

構文

object.Distance

項目	データ型	内容
戻り値	long	道のり(m 単位)を取得します。

解説

Distance プロパティは、ToNodeCode プロパティで設定されているノードまでの道のり(m 単位)を取得します。

Toll_SS プロパティ

軽・自動二輪の通行料金(円単位)を取得します。値の設定は出来ません。

構文

object.**Toll_SS**

項目	データ型	内容
戻り値	long	軽・自動二輪の通行料金(円単位)を取得します。

解説

Toll_SS プロパティは、ToNodeCode プロパティで設定されているノードまでの軽・自動二輪の通行料金(円単位)を取得します。

Toll_S プロパティ

普通車の通行料金(円単位)を取得します。値の設定は出来ません。

構文

object.Toll_S

項目	データ型	内容
戻り値	long	普通車の通行料金(円単位)を取得します。

解説

Toll_S プロパティは、ToNodeCode プロパティで設定されているノードまでの普通車の通行料金(円単位)を取得します。

Toll_M プロパティ

中型車の通行料金(円単位)を取得します。値の設定は出来ません。

構文

object.**Toll_M**

項目	データ型	内容
戻り値	long	中型車の通行料金(円単位)を取得します。

解説

Toll_M プロパティは、ToNodeCode プロパティで設定されているノードまでの中型車の通行料金(円単位)を取得します。

Toll_L プロパティ

大型車の通行料金(円単位)を取得します。値の設定は出来ません。

構文

object.Toll_L

項目	データ型	内容
戻り値	long	大型車の通行料金(円単位)を取得します。

解説

Toll_L プロパティは、ToNodeCode プロパティで設定されているノードまでの大型車の通行料金(円単位)を取得します。

Toll_LL プロパティ

特大車の通行料金(円単位)を取得します。値の設定は出来ません。

構文

object.**Toll_LL**

項目	データ型	内容
戻り値	long	特大車の通行料金(円単位)を取得します。

解説

Toll_LL プロパティは、ToNodeCode プロパティで設定されているノードまでの特大車の通行料金(円単位)を取得します。

IsTollExist プロパティ

通行料金が取得可能か取得します。値の設定は出来ません。

構文

object.IsTollExist

項目	データ型	内容
戻り値	VARIANT_BOOL	通行料金が取得可能な場合 TRUE、取得不可能な場合 FALSE。

解説

IsTollExist プロパティは、通行料金が取得可能か取得します。本プロパティは、RoadNetworkDir プロパティで指定されている計算用道路データについて通行料金が取得可能かを判定します。計算用道路データをロードしていない状態でも動作します。

IsDetailExist プロパティ

詳細ルートが取得可能か取得します。値の設定は出来ません。

構文

object.**IsDetailExist**

項目	データ型	内容
戻り値	VARIANT_BOOL	詳細ルートが取得可能な場合 TRUE、取得不可能な場合 FALSE。

解説

IsDetailExist プロパティは、詳細ルートが取得可能か取得します。本プロパティは、RoadNetworkDir プロパティで指定されている計算用道路データについて詳細ルートが取得可能かを判定します。計算用道路データをロードしていない状態でも動作します。

LastError プロパティ

最後に発生したエラー番号を取得します。値の設定は出来ません。

構文

object.LastError

項目	データ型	内容
戻り値	long	最後に発生したエラー番号を取得します。

解説

エラー番号については ENUM 定数一覧を参照してください。

2.4 メソッド

LoadRoadNetwork メソッド

計算用道路データをロードします。

構文

object.**LoadRoadNetwork**

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。

解説

LoadRoadNetwork メソッドは、RoadNetworkDir プロパティに設定されているフォルダ下に存在する計算用道路データをロードします。

UnloadRoadNetwork メソッド

計算用道路データをアンロードします。

構文

object.UnloadRoadNetwork

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。

CalcRoute メソッド

2 点間ルート計算を行います。

構文

object.CalcRoute

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。

解説

CalcRoute メソッドは、FromNodeCode プロパティで設定されているノードから ToNodeCode プロパティで設定されているノードへの 2 点間のルート計算を行います。計算結果は Time、DSecTime、Distance、Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティに設定されます。

CalcRoute2 メソッド

ルート計算を行います。

構文

object.**CalcRoute2**(*Locs*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Locs</i>	ILocs	ルート計算を行う情報を格納した <i>Locs</i> コレクションオブジェクト

解説

CalcRoute2 メソッドは、*Locs* コレクションオブジェクト内の *Loc* オブジェクトで設定されている *NodeCode* プロパティのノード間ルート計算を行います。計算結果は各 *Loc* オブジェクトの *Time*、*DSecTime*、*Distance*、*Toll_SS*、*Toll_S*、*Toll_M*、*Toll_L*、*Toll_LL* プロパティに格納されます。

また、CalcRoute2 メソッドは、*Time*、*DSecTime*、*Distance*、*Toll_SS*、*Toll_S*、*Toll_M*、*Toll_L*、*Toll_LL* プロパティに設定されている値を加算します。これにより車輛の発時刻や各ノードでの待機時間等を考慮したルート計算が可能です。考慮をしないためには全ての *Loc* オブジェクトの *Time*、*DSecTime*、*Distance*、*Toll_SS*、*Toll_S*、*Toll_M*、*Toll_L*、*Toll_LL* プロパティに 0(ゼロ)を設定する必要があります。

CalcOptRoute メソッド

最短ルート(巡回セールスマン問題)計算を行います。

構文

object.**CalcOptRoute**(*Locs*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Locs</i>	ILocs	最短ルート計算を行う情報を格納した <i>Locs</i> コレクションオブジェクト

解説

CalcOptRoute メソッドは、Locs コレクションオブジェクト内の Loc オブジェクトで設定されている NodeCode プロパティのノード間の最短ルート(巡回セールスマン問題)計算を行います。計算の結果、Locs コレクション内の Loc オブジェクトの格納順が変更されます。但し、最初と最後の Loc オブジェクトの格納順は変更されません。計算結果は各 Loc オブジェクトの Time、DSecTime、Distance、Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティに格納されます。

また、CalcOptRoute メソッドは、Time、DSecTime、Distance、Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティに設定されている値を加算します。これにより車輛の発時刻や各ノードでの待機時間等を考慮したルート計算が可能です。考慮をしないためには全ての Loc オブジェクトの Time、DSecTime、Distance、Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティに 0(ゼロ)を設定する必要があります。

GetRouteXY メソッド

簡易ルート情報(経度緯度)を取得します。

構文

object.GetRouteXY(*Points*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Points</i>	IPoints	簡易ルート情報(経度緯度)を取得する Points コレクションオブジェクト

解説

GetRouteXY メソッドは、ToNodeCode プロパティで指定されているノードまでの簡易ルート情報(経度緯度)を取得します。簡易ルート情報(経度緯度)は Points のコレクションに追加されます。本メソッド呼び出し時に Points のコレクションに Point オブジェクトが存在する場合、削除されず追加されることに注意してください。

GetRouteXYToMMF メソッド

簡易ルート情報(経度緯度)を取得し、MMF(メモリマップドファイル)に格納します。

構文

object.**GetRouteXYToMMF**(*MMFFileName*)

項目	データ型	内容
戻り値	long	正常終了した場合 MMF に格納した簡易ルート情報のポイント数、異常終了した場合-1
<i>MMFFileName</i>	BSTR	MMF ファイル名称を指定

解説

GetRouteXYToMMF メソッドは、ToNodeCode プロパティで指定されているノードまで簡易ルート情報(経度緯度)を取得し、MMF(メモリマップドファイル)に格納します。MMF(メモリマップドファイル)使用後は、本メソッドを呼び出したアプリケーションが MMF(メモリマップドファイル)を解放しなければなりません。

GetRouteDetail メソッド

詳細ルート情報(経度緯度)を取得します。

構文

object.GetRouteDetail(*Points*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Points</i>	IPoints	詳細ルート情報(経度緯度)を取得する Points コレクションオブジェクト

解説

GetRouteDetail メソッドは、ToNodeCode プロパティで指定されているノードまでの詳細ルート情報(経度緯度)を取得します。詳細ルート情報(経度緯度)は Points のコレクションに追加されます。本メソッド呼び出し時に Points のコレクションに Point オブジェクトが存在する場合、削除されず追加されることに注意してください。

GetRouteDetailToMMF メソッド

詳細ルート情報(経度緯度)を取得し、MMF(メモリマップドファイル)に格納します。

構文

object.GetRouteDetailToMMF(*MMFileName*)

項目	データ型	内容
戻り値	long	正常終了した場合 MMF に格納した詳細ルート情報のポイント数、異常終了した場合-1
<i>MMFileName</i>	BSTR	MMF ファイル名称を指定

解説

GetRouteDetailToMMF メソッドは、ToNodeCode プロパティで指定されているノードまで詳細ルート情報(経度緯度)を取得し、MMF(メモリマップドファイル)に格納します。MMF(メモリマップドファイル)使用後は、本メソッドを呼び出したアプリケーションが MMF(メモリマップドファイル)を解放しなければなりません。

GetRouteNodeLink メソッド

ルート情報(ノードコードまたはリンクコード)を取得します。

構文

object.**GetRouteNodeLink**(*RouteKind*, *NLCodes*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>RouteKind</i>	ENUM_ROUTEKIND	取得するルート情報の種類を指定します。
<i>NLCodes</i>	INLCodes	ルート情報(ノードコードまたはリンクコード)を取得する NLCodes コレクションオブジェクト

解説

GetRouteNodeLink メソッドは、ToNodeCode プロパティで指定されているノードまでのルート情報(ノードコードまたはリンクコード)を取得します。ルート情報(ノードコードまたはリンクコード)は NLCodes のコレクションに追加されます。本メソッド呼び出し時に NLCodes のコレクションに NLCode オブジェクトが存在している場合、クリアされず追加されることに注意してください。

また、ENUM_ROUTEKIND 型は、ENUM 定数一覧を参照してください。

CalcArea メソッド

到達圏計算を行います。

構文

object.**CalcArea**(*RangeKind*, *Range*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>RangeKind</i>	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
<i>Range</i>	long	到達範囲を指定します。

解説

CalcArea メソッドは、FromNodeCode プロパティで指定されているノードから到達範囲までの到達圏計算を行います。到達圏種別は、時間圏と距離圏があり、ENUM_RANGEKIND 型で定義されています。ENUM_RANGEKIND 型は、ENUM 定数一覧を参照してください。また、到達範囲に指定する値の単位は、時間圏計算の場合 1/10 秒単位、距離圏計算の場合 m 単位となります。

MakePolygon メソッド

到達圏計算を行い、簡易型(凸型)ポリゴンを取得します。

構文

object.**MakePolygon**(*RangeKind*, *Range*, *Points*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>RangeKind</i>	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
<i>Range</i>	long	到達範囲を指定します。
<i>Points</i>	IPoints	ポリゴンを格納する Points コレクションオブジェクト

解説

MakePolygon メソッドは、FromNodeCode プロパティで指定されているノードから到達範囲までの到達圏計算を行い、簡易型(凸型)ポリゴンを取得します。到達圏種別は、時間圏と距離圏があり、ENUM_RANGEKIND 型で定義されています。ENUM_RANGEKIND 型は、ENUM 定数一覧を参照してください。

また、到達範囲に指定する値の単位は、時間圏計算の場合 1/10 秒単位、距離圏計算の場合 m 単位となります。

MakePolygonToMMF メソッド

到達圏計算を行い、簡易型(凸型)ポリゴンを取得し、MMF(メモリマップドファイル)に格納します。

構文

object.**MakePolygonToMMF**(*RangeKind*, *Range*, *MMFileName*)

項目	データ型	内容
戻り値	long	正常終了した場合 MMF に格納した簡易型(凸型)ポリゴンのポイント数、異常終了した場合-1
<i>RangeKind</i>	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
<i>Range</i>	long	到達範囲を指定します。
<i>MMFileName</i>	BSTR	MMF ファイル名称を指定

解説

MakePolygonToMMF メソッドは、FromNodeCode プロパティで指定されているノードから到達範囲までの到達圏計算を行い、簡易型(凸型)ポリゴンを取得し、MMF(メモリマップドファイル)に格納します。

到達圏種別は、時間圏と距離圏があり、ENUM_RANGEKIND 型で定義されています。

ENUM_RANGEKIND 型は、ENUM 定数一覧を参照してください。

また、到達範囲に指定する値の単位は、時間圏計算の場合 1/10 秒単位、距離圏計算の場合 m 単位となります。

MMF(メモリマップドファイル)使用後は、本メソッドを呼び出したアプリケーションが MMF(メモリマップドファイル)を解放しなければなりません。

MakePolygon2 メソッド

到達圏計算を行い、標準型(凹凸型)ポリゴンを取得します。

構文

object.MakePolygon2(*RangeKind*, *Range*, *Level*, *Points*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>RangeKind</i>	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
<i>Range</i>	long	到達範囲を指定します。
<i>Level</i>	long	ポリゴンレベルを指定します。
<i>Points</i>	IPoints	ポリゴンを格納する Points コレクションオブジェクト

解説

MakePolygon2 メソッドは、FromNodeCode プロパティで指定されているノードから到達範囲までの到達圏計算を行い、標準型(凹凸型)ポリゴンを取得します。到達圏種別は、時間圏と距離圏があり、ENUM_RANGEKIND 型で定義されています。ENUM_RANGEKIND 型は、ENUM 定数一覧を参照してください。

また、到達範囲に指定する値の単位は、時間圏計算の場合 1/10 秒単位、距離圏計算の場合 m 単位となります。ポリゴンを作成する際のポリゴンレベルを 0~20 の間で指定が可能です。ポリゴンレベルの数値が大きいほどより細かく正確なポリゴンを作成できますが、ポリゴン作成に失敗することもあります。通常は 0~5 の値での使用をおすすめします。

MakePolygon2ToMMF メソッド

到達圏計算を行い、標準型(凹凸型)ポリゴンを取得し、MMF(メモリマップドファイル)に格納します。

構文

```
object.MakePolygon2ToMMF(RangeKind, Range, Level, MMFileName)
```

項目	データ型	内容
戻り値	long	正常終了した場合 MMF に格納した標準型(凹凸型)ポリゴンのポイント数、異常終了した場合-1
<i>RangeKind</i>	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
<i>Range</i>	long	到達範囲を指定します。
<i>Level</i>	long	ポリゴンレベルを指定します。
<i>MMFileName</i>	BSTR	MMF ファイル名称を指定

解説

MakePolygon2ToMMF メソッドは、FromNodeCode プロパティで指定されているノードから到達範囲までの到達圏計算を行い、標準型(凹凸型)ポリゴンを取得し、MMF(メモリマップドファイル)に格納します。

到達圏種別は、時間圏と距離圏があり、ENUM_RANGEKIND 型で定義されています。ENUM_RANGEKIND 型は、ENUM 定数一覧を参照してください。

また、到達範囲に指定する値の単位は、時間圏計算の場合 1/10 秒単位、距離圏計算の場合 m 単位となります。ポリゴンを作成する際のポリゴンレベルを 0~20 の間で指定が可能です。ポリゴンレベルの数値が大きいほどより細かく正確なポリゴンを作成できますが、ポリゴン作成に失敗することもあります。通常は 0~5 の値での使用をおすすめします。

MMF(メモリマップドファイル)使用後は、本メソッドを呼び出したアプリケーションが MMF(メモリマップドファイル)を解放しなければなりません。

GetNodeXY メソッド

ノードの位置情報(経度緯度)を取得します。

構文

object.GetNodeXY(*Loc*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Loc</i>	ILoc	ノードコードを設定した Loc オブジェクト

解説

GetNodeXY メソッドは、Loc オブジェクトの NodeCode プロパティに格納されているノードの位置情報(経度緯度)を取得し、Loc オブジェクトの Longitude、Latitude プロパティに設定します。

GetNearestNode メソッド

指定位置(経度緯度)の近傍ノードコードを取得します。

構文

object.GetNearestNode(*Loc*, *Range*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>Loc</i>	ILoc	指定位置(経度緯度)を設定した Loc オブジェクト
<i>Range</i>	double	探索レンジ(度単位)

解説

GetNearestNode メソッドは、Loc オブジェクトの Longitude、Latitude プロパティに格納されている位置から、探索レンジで指定された範囲内で最も近いノードを探索し、そのノードコードを Loc オブジェクトの NodeCode プロパティに設定します。

UseHighway プロパティの値が USEHIGHWAY_YES の場合、高速道路のノードを含めて探索し、USEHIGHWAY_NO の場合、(高速道路以外の)一般道路のノードのみを探索します。

GetSpeed メソッド

リンクの速度情報(片方向)を取得します。

構文

object.**GetSpeed**(*FromNodeCode*, *ToNodeCode*, *Flag*)

項目	データ型	内容
戻り値	long	正常終了した場合 速度情報(1/10km/h 単位)、異常終了した場合 -1
<i>FromNodeCode</i>	double	始点のノードコード
<i>ToNodeCode</i>	double	終点のノードコード
<i>Flag</i>	ENUM_SPEED	速度フラグ

解説

GetSpeed メソッドは、始点(*FromNodeCode*)から終点(*ToNodeCode*)方向へのリンクの速度情報を取得します。始点と終点が1リンクの両端のノードでない場合、メソッドは失敗します。速度フラグは、ENUM_SPEED 型として定義されています。ENUM_SPEED 型は、ENUM 定数一覧を参照してください。

一時的速度を取得するには、計算用道路データがロードされている必要があります。また、計算用道路データがロードされている場合でも、SPEED_PARAM 値を使用することで恒久的速度を取得することが出来ます。SPEED_TEMP と SPEED_PARAM を両方指定した場合、一時的速度が優先されます。SPEED_ZERORESERVE は指定しないでください。

GetSpeedEx メソッド

リンクの速度情報(両方向)を取得します。

構文

object.**GetSpeedEx**(*LinkCode*, *Flag*, *SpeedAB*, *SpeedBA*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE
<i>LinkCode</i>	double	速度を取得するリンクのリンクコード
<i>Flag</i>	ENUM_SPEED	速度フラグ
<i>SpeedAB</i>	long	順方向速度を取得するポインタ
<i>SpeedBA</i>	long	逆方向速度を取得するポインタ

解説

GetSpeedEx メソッドは、指定したリンクの速度情報を取得します。

速度フラグは、ENUM_SPEED 型として定義されています。ENUM_SPEED 型は、ENUM 定数一覧を参照してください。

一時的速度を取得するには、計算用道路データがロードされている必要があります。また、計算用道路データがロードされている場合でも、SPEED_PARAM 値を使用することで恒久的速度を取得することが出来ます。SPEED_TEMP と SPEED_PARAM を両方指定した場合、一時的速度が優先されます。SPEED_ZERORESERVE は指定しないでください。

ChangeBothSpeedEx メソッド

リンクの速度情報(両方向)を設定します。

構文

object.**ChangeBothSpeedEx**(*LinkCode*, *SpeedAB*, *SpeedBA*, *Flag*)

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。
<i>LinkCode</i>	double	速度を設定するリンクのリンクコード
<i>SpeedAB</i>	long	順方向速度
<i>SpeedBA</i>	long	逆方向速度
<i>Flag</i>	ENUM_SPEED	速度フラグ

解説

ChangeBothSpeedEx メソッドは、指定したリンクの速度情報を設定します。

速度に負値を指定した場合、当該方向の速度は変更されません。

速度フラグは、ENUM_SPEED 型として定義されています。ENUM_SPEED 型は、ENUM 定数一覧を参照してください。

一時的速度を取得するには、計算用道路データがロードされている必要があります。SPEED_TEMP 値を指定しない場合、一時的速度と恒久的速度の両方の速度が変更されます。SPEED_ZERORESERVE を設定した場合、変更前の速度が 0 に設定されている方向の速度は変更されません。これによって、一方通行に設定されている道路に誤って速度を設定してしまい一方通行を解除してしまうことを防ぐことができます。

CountTempSpeed メソッド

一時的速度が設定されているリンク数を取得します。

構文

object.CountTempSpeed

項目	データ型	内容
戻り値	long	正常終了した場合、一時的速度が設定されているリンク数、異常終了した場合 -1。

解説

一時的速度が設定されているリンク数を取得するには、計算用道路データがロードされている必要があります。

CommitTempSpeed メソッド

一時的速度を恒久的速度として保存します。

構文

object.CommitTempSpeed

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。

解説

一時的速度の保存するには、計算用道路データがロードされている必要があります。

ClearTempSpeed メソッド

一時的速度を取り消し恒久的速度に戻します。

構文

object.ClearTempSpeed

項目	データ型	内容
戻り値	VARIANT_BOOL	正常終了した場合 TRUE、異常終了した場合 FALSE。

解説

一時的速度の取り消しには、計算用道路データがロードされている必要があります。

ClearLastError メソッド

エラー番号をクリアします。

構文

object.ClearLastError

項目	データ型	内容
----	------	----

なし

解説

LastError プロパティをエラーなし(ERR_None)に設定します。

3 . ENUM 定数一覧

3 . 1 計算種別(ENUM_CALCKIND 型)

ENUM_CALCKIND 型は、計算種別を表します。

デフォルト値は CALCKIND_TIME(時間最短)です。

値	内容
CALCKIND_TIME	時間最短
CALCKIND_DIST	距離最短

3 . 2 高速道路の使用 / 非使用(ENUM_USEHIGHWAY 型)

ENUM_USEHIGHWAY 型は、高速道路の使用 / 非使用を表します。

デフォルト値は、USEHIGHWAY_YES (高速道路を使用する) です。

値	内容
USEHIGHWAY_YES	高速道路を使用する
USEHIGHWAY_NO	高速道路を使用しない

3 . 3 ルート情報種別(ENUM_ROUTEKIND 型)

ENUM_ROUTEKIND 型は、ルート情報種別を表します。

値	内容
ROUTEKIND_NODE	ノードコードを取得する
ROUTEKIND_LINK	リンクコードを取得する
ROUTEKIND_NODELINK	ノードコード、リンクコードを双方交互に取得する

3.4 到達圏種別(ENUM_RANGEKIND 型)

ENUM_RANGEKIND 型は、到達圏種別を表します。

値	内容
RANGEKIND_TIME	時間圏を計算する
RANGEKIND_DIST	距離圏を計算する

3.5 速度フラグ(ENUM_SPEED 型)

ENUM_SPEED 型は、速度フラグを表します。

値	内容
SPEED_TEMP	一時的速度を取得する
SPEED_PARAM	恒久的速度を取得する(*1)
SPEED_ZERORESERVE	速度 0 のリンク速度は、速度設定を行わない(*2)

*1 ChangeBothSpeedEx メソッドでは使用不可

*2 GetSpeed、GetSpeedEx メソッドでは使用不可

3.6 エラー番号(ERR_ACTCALCM 型)

ERR_ACTCALCM 型は、エラー番号を表します。

値	内容
ERR_None	エラーなし
ERR_Get_RoadNetworkDir	RoadNetworkDir プロパティの取得に失敗
ERR_Set_RoadNetworkDir	RoadNetworkDir プロパティの設定に失敗
ERR_NotExistsDir	RoadNetworkDir プロパティで設定しようとしたフォルダが存在しない
ERR_LoadRoadNetwork	計算用道路地図のロードに失敗
ERR_AlreadyRoadNetworkLoaded	すでに Load 済みのオブジェクトが再度ロードしようとした
ERR_NotExistRoadNetwork	計算用道路地図が存在しない
ERR_UnloadRoadNetwork	計算用道路地図のアンロードに失敗
ERR_NotLoadedRoadNetwork	計算用道路地図をロードしていない状態で、アンロードしようとした
ERR_Set_CalcKind	CalcKind プロパティの設定に失敗
ERR_InvalidCalcKind	CalcKind プロパティに無効な値を設定しようとした
ERR_Set_UseHighway	UseHighway プロパティの設定に失敗
ERR_InvalidUseHighway	UseHighway プロパティに無効な値を設定しようとした
ERR_InvalidFromNodeCode	FromNodeCode プロパティに無効な値を設定しようとした
ERR_InvalidToNodeCode	ToNodeCode プロパティに無効な値を設定しようとした
ERR_CalcRoute	CalcRoute メソッド異常終了
ERR_CalcRouteFailure	CalcRoute メソッド失敗(DLL レベル)
ERR_CalcRoute2	CalcRoute2 メソッド異常終了
ERR_CalcRoute2LocsCount	Locs コレクション内の Loc オブジェクト数が 2 未満のため計算不能
ERR_CalcRoute2Failure	CalcRoute2 メソッド失敗(DLL レベル)

ENUM 定数一覧

ERR_CalcOpt	CalcOptRoute メソッド異常終了
ERR_CalcOptLocsCount	Locs コレクション内の Loc オブジェクト数が 2 未満のため計算不能
ERR_CalcOptFailure	CalcOptRoute メソッド失敗(DLL レベル)
ERR_GetRouteXY	GetRouteXY メソッド異常終了
ERR_GetRouteXYFailure	GetRouteXY メソッド失敗(DLL レベル)
ERR_GetRoute	GetRoute メソッド異常終了
ERR_GetRouteNode	GetRoute(Node)メソッド失敗(DLL レベル)
ERR_GetRouteLink	GetRoute(Link)メソッド失敗(DLL レベル)
ERR_GetRouteNodeLink	GetRoute(NodeLink)メソッド失敗(DLL レベル)
ERR_CalcArea	CalcArea メソッド異常終了
ERR_CalcAreaFailure	CalcArea メソッド失敗(DLL レベル)
ERR_MakePolygon	MakePolygon メソッド異常終了
ERR_InvalidRangeKind	RangeKind に無効な値を設定しようとした
ERR_InvalidRange	Range に無効な値を設定しようとした
ERR_MakePolygon2	MakePolygon2 メソッド異常終了
ERR_InvalidRangeKind2	RangeKind に無効な値を設定しようとした
ERR_InvalidRange2	Range に無効な値を設定しようとした
ERR_InvalidLevel2	Level に無効な値を設定しようとした
ERR_CreateObjectFailure	オブジェクトの作成に失敗
ERR_AlreadyAttached	すでにアタッチ済みである <ul style="list-style-type: none"> ・アタッチされている親オブジェクトでアンロードしようとした ・アタッチされている親オブジェクトで速度変更メソッドを実行しようとした ・アタッチしている子オブジェクトがさらにアタッチしようとした
ERR_ParentNotLoaded	計算用道路地図をロードしていない親オブジェクトにアタッチしようとした
ERR_AttachFailure	アタッチに失敗(DLL レベル)
ERR_License_Limit	ライセンス期限切れ

ERR_CreateMMF	MMF(メモリマップドファイル)作成に失敗
ERR_MapViewOfFile	MMF(メモリマップドファイル)のマッピングに失敗
ERR_GetRouteXYToMMFFailure	GetRouteXY メソッド失敗(DLL レベル)
ERR_GetRouteXYToMMF	GetRouteXY メソッド異常終了
ERR_NotExistDetail	詳細ルートデータが存在しない
ERR_GetRouteDetailFailure	GetRouteDetail メソッド(DLL レベル)
ERR_GetRouteDetail	GetRouteDetail 異常終了
ERR_GetRouteDetailToMMFFailure	GetRouteDetailToMMF メソッド(DLL レベル)
ERR_GetRouteDetailToMMF	GetRouteDetailToMMF 異常終了
ERR_MakePolygonToMMF	MakePolygonToMMF 異常終了
ERR_MakePolygon2ToMMF	MakePolygon2ToMMF 異常終了

4 . コーディング例

距離計算オブジェクトの Visual Basic 6.0 を用いたコーディング例を示します。

4 . 1 2 点間計算

```
' DistCalc オブジェクトの CalcRoute メソッドにより
' 2 点間ルート計算を実行します。
' 実行後は Time,DSecTime,Distance,Toll_S プロパティの値および
' 詳細ルート情報を MsgBox で表示します。
Private Sub Command1_Click()
    Dim DistCalc As ActCalcX.DistCalc      ' DistCalc オブジェクト
    Dim Points As ACTDataX.Points        ' Points オブジェクト(詳細ルート情報用)
    Dim i As Long

    ' オブジェクトを初期化します
    Set DistCalc = New ActCalcX.DistCalc
    Set Points = New ACTDataX.Points

    ' 計算用道路データフォルダを設定します (例)
    DistCalc.RoadNetworkDir = "e:\Sample\計算地図"

    ' 計算用道路データをメモリ上にロードします
    DistCalc.LoadRoadNetwork

    ' プロパティを設定します
    DistCalc.CalcKind = CALCKIND_TIME      ' 時間最短
    DistCalc.UseHighway = USEHIGHWAY_YES  ' 高速道路使用
    DistCalc.FromNodeCode = 13000000120780# ' 始点ノードコード (例)
    DistCalc.ToNodeCode = 13000000122582#  ' 終点ノードコード (例)

    ' 2 点間ルート計算を実行します
    DistCalc.CalcRoute

    ' 結果を表示します
    MsgBox "Time =" + Str$(DistCalc.Time) + Chr$(13) + _
        "DSecTime =" + Str$(DistCalc.DSecTime) + Chr$(13) + _
        "Distance =" + Str$(DistCalc.Distance)
    If DistCalc.IsTollExist Then
        MsgBox "Toll_S =" + Str$(DistCalc.Toll_S)
    End If
    If DistCalc.IsDetailExist Then
        DistCalc.GetRouteDetail Points
        MsgBox "詳細ルート情報は" + Str(Points.Count) + "点です"
        For i = 0 To Points.Count - 1
            MsgBox Str(i + 1) + "点目" + _
                " Longitude =" + Str(Points.Item(i).Longitude) + _
                " Latitude =" + Str(Points.Item(i).Latitude)
        Next
    End If

    ' 計算用道路データをメモリ上から解放します
    DistCalc.UnloadRoadNetwork

    ' オブジェクトを解放します
    Set DistCalc = Nothing
    Set Points = Nothing
End Sub
```

4.2 最短ルート計算

```

' DistCalc オブジェクトの CalcOptRoute メソッドにより
' 最短ルート計算を実行します。
' 実行後は求めたルート順に Locs オブジェクトの
' Tag,Time,DSecTime,Distance プロパティの値を MsgBox で表示します。
Private Sub Command1_Click()
    Dim i As Integer           ' ループカウンタ
    Dim ldNodeCode(5) As Double ' ノードコード
    Dim szBuff As String       ' 表示用
    Dim DistCalc As ActCalcX.DistCalc ' DistCalc オブジェクト
    Dim DistLocs As ACTDataX.Locs ' Locs オブジェクト
    Dim DistLoc As ACTDataX.Loc ' Loc オブジェクト

    ' オブジェクトを初期化します
    Set DistCalc = New ActCalcX.DistCalc
    Set DistLocs = New ACTDataX.Locs
    Set DistLoc = New ACTDataX.Loc

    ' 計算用道路データフォルダを設定します (例)
    DistCalc.RoadNetworkDir = "E:\Sample\計算地図"

    ' 計算用道路データをメモリ上にロードします
    DistCalc.LoadRoadNetwork

    ' プロパティを設定します
    DistCalc.CalcKind = CALCKIND_TIME ' 時間最短
    DistCalc.UseHighway = USEHIGHWAY_YES ' 高速道路使用

    ' 中継点を設定します (例)
    ldNodeCode(0) = 13000000036205#
    ldNodeCode(1) = 13000000076996#
    ldNodeCode(2) = 13000000124377#
    ldNodeCode(3) = 13000000122692#
    ldNodeCode(4) = 13000000036205#

    ' Locs オブジェクトにデータを設定します
    DistLocs.Clear
    For i = 0 To 4
        DistLoc.NodeCode = ldNodeCode(i) ' ノードコード (例) を設定します
        DistLoc.Tag = DistLocs.Count ' Tag を設定します
        DistCalc.GetNodeXY DistLoc ' ノードコードの座標を求めます
        DistLocs.Add DistLoc ' Locs オブジェクトに追加します
    Next

    ' 最短ルート計算を実行します
    DistCalc.CalcOptRoute DistLocs

    ' 結果を表示します
    szBuff = "Locs=" + Str$(DistLocs.Count)
    For i = 0 To 4
        szBuff = szBuff + Chr$(13) _
            + "Tag=" + Str$(DistLocs.Item(i).Tag) + " : " _
            + "Time=" + Str$(DistLocs.Item(i).Time) + " : " _
            + "DSecTime=" + Str$(DistLocs.Item(i).DSecTime) + " : " _
            + "Distance=" + Str$(DistLocs.Item(i).Distance)

    Next
    MsgBox szBuff

    ' 計算用道路データをメモリ上から解放します
    DistCalc.UnloadRoadNetwork

    ' オブジェクトを解放します
    Set DistCalc = Nothing
    Set DistLocs = Nothing
    Set DistLoc = Nothing
End Sub

```

4.3 到達圏計算

```

' DistCalc オブジェクトの CalcArea メソッドのにより到達圏計算を実行します。
' 実行後は求めた到達圏ポリゴンの頂点座標を MsgBox で表示します。
Private Sub Command1_Click()
    Dim DistCalc As ActCalcX.DistCalc      ' DistCalc オブジェクト
    Dim AreaPoints As ACTDataX.Points     ' Points オブジェクト
    Dim i As Integer                       ' ループカウンタ
    Dim szBuff As String                   ' 表示用

    ' オブジェクトを初期化します
    Set DistCalc = New ActCalcX.DistCalc
    Set AreaPoints = New ACTDataX.Points

    ' 計算用道路データフォルダを設定します (例)
    DistCalc.RoadNetworkDir = "E:\Sample\計算地図"

    ' 計算用道路データをメモリ上にロードします
    DistCalc.LoadRoadNetwork

    ' プロパティを設定します
    DistCalc.CalcKind = CALCKIND_TIME      ' 時間最短
    DistCalc.UseHighway = USEHIGHWAY_YES  ' 高速道路使用
    DistCalc.FromNodeCode = 13000000036205# ' スタート点ノードコード (例)

    ' 到達圏ポリゴンを求めます (1分圏)
    DistCalc.MakePolygon RANGEKIND_TIME, 600, AreaPoints

    ' 結果を表示します
    szBuff = "Points =" + Str$(AreaPoints.Count)
    For i = 0 To AreaPoints.Count - 1
        szBuff = szBuff + Chr$(13) + Str$(i) + " : " _
            + "Lon =" + Str$(AreaPoints.Item(i).Longitude) + " : " _
            + "Lat =" + Str$(AreaPoints.Item(i).Latitude)

    Next
    MsgBox szBuff

    ' 計算用道路データをメモリ上から解放します
    DistCalc.UnloadRoadNetwork

    ' オブジェクトを解放します
    Set DistCalc = Nothing
    Set AreaPoints = Nothing
End Sub

```

