ACT コンポーネントシリーズ

ACT 距離計算コンポーネント(マルチ版)
Version 2.0
プログラマーズ
リファレンス

アドバンスド・コア・テクノロジー株式会社

ACT コンポーネントシリーズ
ACT 距離計算コンポーネント(マルチ版) Version 2.0 プログラマーズリファレンス
２００１年０１月０４日　初版発行
２００２年０２月０１日　第２版発行
２００２年１２月１６日　第３版発行
２０１２年０９月２０日　第４版発行
２０１４年０４月０７日　第５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２階
電話　03-5512-9021　FAX　03-5512-9022

ACT 距離計算コンポーネント(マルチ版)
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(マルチ版)の概要

１．はじめに

ACT 距離計算コンポーネント(マルチ版)は、スレッドに対応した距離計算を行うための COM（Component Object Model）コンポーネントです。マルチ CPU を搭載したＰＣや、イントラネット／インターネット環境で距離計算を行う場合に最適なパフォーマンスを発揮します。

（注意）本ライブラリは互換性のために残されています。64bit 環境では SOAP 経由での距離計算プログラムの実装をお願い致します。

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(マルチ版)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(マルチ版)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(マルチ版)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(マルチ版)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(マルチ版)は、次の OS(64bit 版)で動作します。

・Windows 2003/2008/2008 R2/2012 サーバーシリーズ

５．開発環境

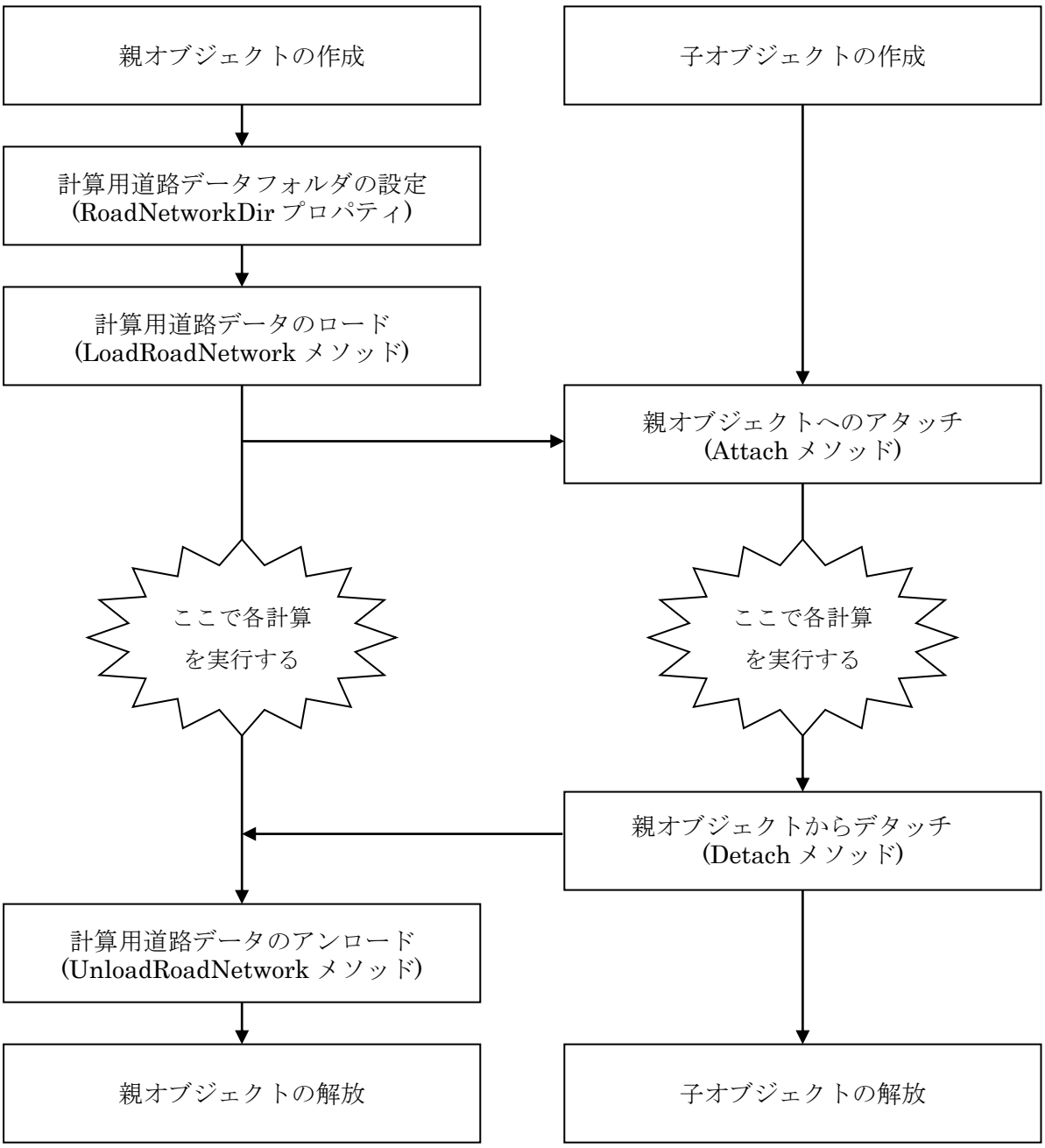
ACT 距離計算コンポーネント(マルチ版)は、COMDLL 形式で提供されています。COM に対応した開発環境であれば、***ACT*** 距離計算コンポーネント（マルチ版）を用いたアプリケーションの開発が可能です。

６．オブジェクト一覧

以下は、***ACT*** 距離計算コンポーネント(マルチ版)のオブジェクト一覧です。

オブジェクト名	解説
DistCalcParent	計算用道路データ管理機能や速度変更機能を持った距離計算親オブジェクト
DistCalcChild	親(DistCalcParent)オブジェクトにアタッチして距離計算を行う子オブジェクト

距離計算コンポーネント(マルチ版)には、親・子の２種類のオブジェクトが存在します。
下図は計算用道路データをロードしている距離計算オブジェクト（親オブジェクト）と、アタッチ先の距離計算オブジェクト（子オブジェクト）の関係を示しています。子オブジェクトが親オブジェクトにアタッチする前に、親オブジェクトは計算用道路データのロードが完了しなくてはなりません。また、親オブジェクトの計算用道路データをアンロードする前に、子オブジェクトを解放しなくてはなりません。



７．Version 2.0 の変更点

Version 2.0 では以下の点が変更されました。

・通行料金に関するプロパティ、メソッドが動作するようになりました。

・DistCalcParent、DistCalcChild オブジェクトに経由交差点、経由有料道路に関するメソッドが追加されました。追加されたメソッドは次の通りです。

項番	オブジェクト	メソッド
1	DistCalcParent DistCalcChild	ClearRouteName メソッド
2	DistCalcParent DistCalcChild	AppendRouteName メソッド
3	DistCalcParent DistCalcChild	GetRouteName メソッド
4	DistCalcParent DistCalcChild	ClearTollRoadDetail メソッド
5	DistCalcParent DistCalcChild	AppendTollRoadDetail メソッド
6	DistCalcParent DistCalcChild	GetTollRoadDetail メソッド

・DistCalcParent オブジェクトに MemoryOption メソッドが追加されました。

８．旧 Version からの移行

(1)Version 1.5 からの移行

Version 1.5 から移行する場合、基本的にアプリケーションの再構築(再コンパイル)が不要です。安全性を確保するため、再構築(再コンパイル)を行うことをお勧めします。

(2)Version 1.0 からの移行

Version 1.0 から移行する場合、アプリケーションの再構築(再コンパイル)が必要です。
Toll プロパティが削除され、Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティに変更されています。

（メモ）

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

以下は、ACT 距離計算コンポーネント(マルチ版)の各オブジェクトのプログラム ID(ProgID)です。

オブジェクト名	プログラム ID
DistCalcParent オブジェクト	ActCalcM.DistCalcParent,2.0
DistCalcChild オブジェクト	ActCalcM.DistCalcChild,2.0

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。

２．DistCalcParent オブジェクト

２．１　概要

DistCalcParent オブジェクトは､距離計算コンポーネントのメインとなる親オブジェクトです。DistCalcParent 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。また子オブジェクトからのアタッチも可能になり、子オブジェクトでの計算も可能になります。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. 1.5	Ver. 2.0
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	AttachedChildCount プロパティ	アタッチしている子(DistCalcChild)オブジェクト数を取得します	New		
4	FromNodeCode プロパティ	発地ノードコードを格納します	New		
5	ToNodeCode プロパティ	着地ノードコードを格納します	New		
6	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		
7	UseHighway プロパティ	高速道路の使用/非使用を格納します	New		
8	Time プロパティ	所要時間(分単位)を取得します	New		
9	DSecTime プロパティ	所要時間(1/10 秒単位)を取得します	New		
１０	Distance プロパティ	道のり(m 単位)を取得します	New		
１１	Toll_SS プロパティ	軽・自動二輪の通行料金（円）を格納します		－	New
１２	Toll_S プロパティ	普通車の通行料金（円）を格納します		－	New
１３	Toll_M プロパティ	中型車の通行料金（円）を格納します		－	New

１４	Toll_L プロパティ	大型車の通行料金（円）を格納します		－	New
１５	Toll_LL プロパティ	特大車の通行料金（円）を格納します		－	New
１６	IsTollExist プロパティ	通行料金が存在するか取得します		－	New
１７	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	
１８	MemoryOption プロパティ	各種メソッドの処理高速化の為のメモリオプションを取得します			New
１９	Handle プロパティ	DLL でのハンドル値を取得します	New		
２０	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		
２１	LoadRoadNetwork メソッド	計算用道路データをロードします	New		
２２	UnloadRoadNetwork メソッド	計算用道路データをアンロードします	New		
２３	CalcRoute メソッド	２点間ルート計算を行います	New		
２４	CalcRoute2 メソッド	ルート計算を行います	New		
２５	CalcOptRoute メソッド	最短ルート計算を行います	New		
２６	GetRouteXY メソッド	ルート情報(経度緯度)を取得します	New		
２７	GetRouteDetail メソッド	詳細ルート情報(経度緯度)を取得します		New	
２８	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		
２９	CalcArea メソッド	到達圏計算を行います	New		
３０	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		
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	GetNodeXY メソッド	ノードの位置情報(経度緯度)を取得します	New		
３３	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		
３４	GetSpeed メソッド	リンクの速度情報(片方向)を取得します	New		
３５	GetSpeedEx メソッド	リンクの速度情報(両方向)を取得します	New		

３６	ChangeBothSpeedEx メソッド	リンクの速度情報(両方向)を設定します	New		
３７	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		
３８	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		
３９	ClearTempSpeed メソッド	一時的速度を取り消します	New		
４０	ClearLastError メソッド	エラー番号をクリアします	New		
４１	ClearRouteName メソッド	経由交差点情報をクリアします			New
４２	AppendRouteName メソッド	経由交差点情報を作成・追加します			New
４３	GetRouteName メソッド	経由交差点情報を取得します			New
４４	ClearTollRoadDetail メソッド	有料道路明細情報をクリアします			New
４５	AppendTollRoadDetail メソッド	有料道路明細情報を作成・追記します			New
４６	GetTollRoadDetail メソッド	有料道路明細情報を取得します			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
Value	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。
Value	BSTR	計算用道路データが存在するフォルダを指定します。

AttachedChildCount プロパティ

アタッチしている子(DistCalcChild)オブジェクト数を取得します。値の設定はできません。

構文

object.**AttachedChildCount**

項目	データ型	内容
戻り値	long	アタッチしている子(DistCalcChild)オブジェクト数を取得します。

解説

親オブジェクトが子オブジェクトにアタッチされている場合、次のメソッド実行出来ません。（各メソッドを呼び出した場合、エラーになります）

- 計算用道路データアンロード　UnloadRoadNetwork メソッド
- 道路速度変更関連　GetSpeed メソッド、GetSpeedEx メソッド、ChangeBothSpeedEx メソッド、CommitTempSpeed メソッド、ClearTempSpeed メソッド

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。
Value	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。
Value	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。
Value	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。
Value	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。

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。

MemoryOption プロパティ

各種メソッドの処理高速化の為のメモリオプションを取得します。

構文

object.**MemoryOption**[=*Value*]

項目	データ型	内容
戻り値	ENUM_MEMORYOPTION	メモリオプションを取得します。
Value	ENUM_MEMORYOPTION	メモリオプションを設定します。

解説

MemoryOption プロパティは、各種メソッドの処理高速化の為のメモリオプションを取得します。メモリオプションを有効にするには、本プロパティの値を設定後、LoadRoadNetwork メソッドを呼び出し計算用道路データのロードを行う必要があります。

メモリオプションを設定した場合、以下のプロパティ、メソッドの処理を高速化出来ます。

ENUM_MEMORYOPTION 値	高速化するプロパティ・メソッド
MEMORYOPTION_NORMAL	通常通り
MEMORYOPTION_HIGH	Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL、CalcRoute、CalcRoute2、CalcOptRoute、GetRouteXY 、GetRoute、AppendRouteName、AppendTollRoadDetail
MEMORYOPTION_FULL	GetRouteDetail

メモリオプションに MEMORYOPTION_HIGH を設定すると通常の約 1.5 倍、MEMORYOPTION_FULL を設定すると通常の約 1.8 倍メモリを多く消費します。

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

Handle プロパティ

距離計算コアライブラリ(ACTLIB50.DLL)でオブジェクトを識別するためのハンドル値を取得します。値の設定は出来ません。

構文

object.**Handle**

項目	データ型	内容
戻り値	long	DLL でのハンドル値を取得します。

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 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。

解説

子(DistCalcChild)オブジェクトからアタッチされている場合、計算用道路データのアンロードは出来ません。

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 プロパティ、DSecTime プロパティ、Distance 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
Locs	ILocs	ルート計算を行う情報を格納した Locs 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 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。これにより車輌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
Locs	ILocs	最短ルート計算を行う情報を格納した Locs 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 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。これにより車輌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

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
Points	IPoints	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ルート情報(経度緯度)を取得します。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。

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
Points	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。

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。

構文

object.**GetRoute**(*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
RouteKind	ENUM_ROUTEKIND	取得するルート情報の種類を指定します。
NLCodes	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

GetRoute 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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	long	到達範囲を指定します。
Points	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。

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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	long	到達範囲を指定します。
Level	long	ポリゴンレベルを指定します。
Points	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。

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
Loc	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
Loc	ILoc	指定位置(経度緯度)を設定した Loc オブジェクト
Range	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
FromNodeCode	double	始点のノードコード
ToNodeCode	double	終点のノードコード
Flag	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。

本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

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
LinkCode	double	速度を取得するリンクのリンクコード
Flag	ENUM_SPEED	速度フラグ
SpeedAB	long	順方向速度を取得するポインタ
SpeedBA	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。

本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

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。
LinkCode	double	速度を設定するリンクのリンクコード
SpeedAB	long	順方向速度
SpeedBA	long	逆方向速度
Flag	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。

本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

リンクコードの上位 2 桁が 99 のリンクについて道路速度の設定を行わないでください。

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。本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

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。本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

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。本メソッドは子(DistCalcChild)オブジェクトにアタッチされている場合は使用できません。また、複数の親オブジェクトが同一の計算用道路データに対して同時に速度変更関連メソッドを実行した場合、エラーとなる場合があります。

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。

ClearRouteName メソッド

経由交差点情報をクリアします。

構文

object.**ClearRouteName**

項目	データ型	内容
なし		

解説

経由交差点情報を取得する前に、必ず本メソッドをコールし、経由交差点情報のクリアを行ってください。

AppendRouteName メソッド

経由交差点情報を作成・追加します。

構文

object.**AppendRouteName**(*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。
Flag	ENUM_ROUTENAME	経由交差点フラグの組み合わせ

解説

AppendRouteName メソッドは、ToNodeCode プロパティで設定されているノードへの２点間の経由交差点情報を作成・追加します。本メソッドをコールする前に CalcRoute メソッドを呼び出しルート計算を行ってください。作成・追加された経由交差点情報は、コンポーネント内部に蓄積されます。経由交差点情報を取得するには、GetRouteName メソッドを使用してください。

経由交差点の情報は、引数で指定する経由交差点フラグにより次のように異なります。

経由交差点フラグ	意味
なし(標準)	発着地、中継点を出力します。
ROUTENAME_NODE	交差点名称を出力します。
ROUTENAME_LINK	道路名称を出力します。
ROUTENAME_FARE	インターチェンジ名称と通行料金を出力します。
ROUTENAME_XY	交差点とインターチェンジの経度緯度を出力します。

また、経由交差点フラグによって RouteName オブジェクトの更新されるプロパティも次のように異なります。

プロパティ	経由交差点フラグ				
	なし (標準)	ROUTE NAME _NODE	ROUTE NAME _LINK	ROUTE NAME _FARE	ROUTE NAME _XY
PointFlag	○(1)	○(0)	○(0)		
LinkFlag	○(0)	○(0)	○(1)		
NLCode	○	○	○		
Name		○	○		
Time	○	○	×		
DSecTime	○	○	×		
Distance	○	○	×		
ICType				○	
Toll_SS				○	
Toll_S				○	
Toll_M				○	
Toll_L				○	
Toll_LL				○	
Longitude			×		○
Latitude			×		○

上表の○印は、プロパティが更新されることを意味します。×印は、他のフラグにかかわらずプロパティは更新されないことを意味します。（ROUTENAME_LINK フラグが設定されたことにより出力された RouteName オブジェクトの Time、DSecTime、Distance プロパティは、更新されず必ず 0 になります）

経由交差点フラグは、ENUM_ROUTENAME 型として定義されています。ENUM_ROUTENAME は、ENUM 定数一覧を参照してください。

GetRouteName メソッド

経由交差点情報を取得します。

構文

object.**GetRouteName**(*RouteNames*)

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。
RouteNames	IRouteNames	経由交差点情報を格納する RouteNames コレクションオブジェクト

解説

GetRouteName メソッドは、AppendRouteName メソッドを使用し、作成・追加した経由交差点情報を RouteNames コレクションに格納します。

コレクション内の最後の RouteName オブジェクト(最終着地)の Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティは、それぞれの通行料金の合計金額が格納されます。

ClearTollRoadDetail メソッド

有料道路明細情報をクリアします。

構文

object.**ClearTollRoadDetail**

項目	データ型	内容

なし

解説

有料道路明細情報を取得する前に、必ず本メソッドをコールし、有料道路明細情報のクリアを行ってください。

AppendTollRoadDetail メソッド

有料道路明細情報を作成・追加します。

構文

object.**AppendTollRoadDetail**(*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。
Flag	ENUM_TOLLROADDETAIL	有料道路明細フラグ(拡張用パラメータ)

解説

AppendTollRoadDetail メソッドは、ToNodeCode プロパティで設定されているノードへの 2 点間の有料道路明細情報を作成・追加します。本メソッドをコールする前に CalcRoute メソッドを呼び出しルート計算を行ってください。作成・追加された有料道路明細情報は、コンポーネント内部に蓄積されます。経由交差点情報を取得するには、GetTollRoadDetail メソッドを使用してください。

また、現バージョンでは、有料道路明細フラグには TOLLROADDETAIL_NONE を設定してください。

GetTollRoadDetail メソッド

有料道路明細情報を取得します。

構文

object.**GetTollRoadDetail**(*TollRoadDetails*)

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。
TollRoadDetails	ITollRoadDetails	有料道路明細情報を格納する TollRoadDetails コレクションオブジェクト

解説

GetTollRoadDetail メソッドは、TollRoadDetail メソッドを使用し、作成・追加した有料道路明細情報を TollRoadDetails コレクションに格納します。

３．DistCalcChild オブジェクト

３．１　概要

DistCalcChild オブジェクトは、親(DistCalcParent)オブジェクトに従属した距離計算オブジェクトです。DistCalcChild オブジェクトが計算可能になるためには、計算用道路データをロードした親(DistCalcParent)オブジェクトにアタッチする必要があります。親(DistCalcParent)オブジェクトにアタッチするには Attach メソッドを使用します。処理が終了し、オブジェクトを解放する前に Detach メソッドを使用して、アタッチを解除する必要があります。

３．２　インタフェース一覧

番号	名 称	解　　説	Ver. 1.0	Ver 1.5	Ver. 2.0
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	FromNodeCode プロパティ	発地ノードコードを格納します	New		
3	ToNodeCode プロパティ	着地ノードコードを格納します	New		
4	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		
5	UseHighway プロパティ	高速道路の使用/非使用を格納します	New		
6	Time プロパティ	所要時間(分単位)を取得します	New		
7	DSecTime プロパティ	所要時間(1/10 秒単位)を取得します	New		
8	Distance プロパティ	道のり(m 単位)を取得します	New		
9	Toll_SS プロパティ	軽・自動二輪の通行料金（円）を格納します		－	New
１０	Toll_S プロパティ	普通車の通行料金（円）を格納します		－	New
１１	Toll_M プロパティ	中型車の通行料金（円）を格納します		－	New
１２	Toll_L プロパティ	大型車の通行料金（円）を格納します		－	New
１３	Toll_LL プロパティ	特大車の通行料金（円）を格納します		－	New
１４	IsTollExist プロパティ	通行料金が存在するか取得します		－	New
１５	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	

１６	Handle プロパティ	DLL でのハンドル値を取得します	New		
１７	Parent プロパティ	アタッチしている親(DistCalcParent)オブジェクトを取得します	New		
１８	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		
１９	Attach メソッド	親(DistCalcParent)オブジェクトにアタッチします	New		
２０	Detach メソッド	親(DistCalcParent)オブジェクトからデタッチします	New		
２１	CalcRoute メソッド	２点間ルート計算を行います	New		
２２	CalcRoute2 メソッド	ルート計算を行います	New		
２３	CalcOptRoute メソッド	最短ルート計算を行います	New		
２４	GetRouteXY メソッド	ルート情報(経度緯度)を取得します	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	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		
２７	CalcArea メソッド	到達圏計算を行います	New		
２８	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		
２９	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		
３０	GetNodeXY メソッド	ノードの位置情報(経度緯度)を取得します	New		
３１	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		
３２	ClearLastError メソッド	エラー番号をクリアします	New		
３３	ClearRouteName メソッド	経由交差点情報をクリアします			New
３４	AppendRouteName メソッド	経由交差点情報を作成・追加します			New
３５	GetRouteName メソッド	経由交差点情報を取得します			New

３６	ClearTollRoadDetail メソッド	有料道路明細情報をクリアします			New
３７	AppendTollRoadDetail メソッド	有料道路明細情報を作成・追記します			New
３８	GetTollRoadDetail メソッド	有料道路明細情報を取得します			New

３．３　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。値の設定は出来ません。

構文

object.**Active**

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

解説

親(DistCalcParent)オブジェクトの Active プロパティと異なり、値を設定すること出来ません。
計算可能な状態にするには Attach メソッドを使用して親(DistCalcParent)オブジェクトにアタッチします。

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。
Value	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。
Value	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。
Value	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。
Value	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。

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。

Handle プロパティ

距離計算コアライブラリ(ACTLIB50.DLL)でオブジェクトを識別するためのハンドル値を取得します。値の設定は出来ません。

構文

object.**Handle**

項目	データ型	内容
戻り値	long	DLL でのハンドル値を取得します。

Parent プロパティ

アタッチしている親(DistCalcParent)オブジェクトを取得します。値の設定は出来ません。

構文

object.**Parent**

項目	データ型	内容
戻り値	IDistCalcParent	親(DistCalcParent)オブジェクトを取得します。アッタチしていない場合、NULL ポインタを返します。

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 一覧を参照してください。

３．４　メソッド

Attach メソッド

親(DistCalcParent)オブジェクトにアタッチします。

構文

object.**Attach**(*DistCalcParent*)

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
DistCalcParent	IDistCalcParent	アタッチする親(DistCalcParent)オブジェクトを指定します

解説

アタッチする親(DistCalcParent)オブジェクトが計算用道路データをロードしていない場合、エラーになります。また、すでにアタッチ済みでデタッチせずにアタッチした場合もエラーになります。

Detach メソッド

親(DistCalcParent)オブジェクトからデタッチします。

構文

object.**Detach**

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

子(DistCalcChild)オブジェクトのインスタンスを解放する前に必ず親(DistCalcParent)オブジェクトからデタッチを行ってください。

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 プロパティ、DSecTime プロパティ、Distance 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
Locs	ILocs	ルート計算を行う情報を格納した Locs 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 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。これにより車輌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
Locs	ILocs	最短ルート計算を行う情報を格納した Locs 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 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。これにより車輌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

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
Points	IPoints	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ルート情報(経度緯度)を取得します。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。

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
Points	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。

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。

構文

object.**GetRoute**(*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
RouteKind	ENUM_ROUTEKIND	取得するルート情報の種類を指定します。
NLCodes	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

GetRoute 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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	long	到達範囲を指定します。
Points	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。

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
RangeKind	ENUM_RANGEKIND	到達圏種別(時間圏 or 距離圏)を指定します。
Range	long	到達範囲を指定します。
Level	long	ポリゴンレベルを指定します。
Points	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。

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
Loc	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
Loc	ILoc	指定位置(経度緯度)を設定した Loc オブジェクト
Range	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。

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。

ClearRouteName メソッド

経由交差点情報をクリアします。

構文

object.**ClearRouteName**

項目	データ型	内容

なし

解説

経由交差点情報を取得する前に、必ず本メソッドをコールし、経由交差点情報のクリアを行ってください。

AppendRouteName メソッド

経由交差点情報を作成・追加します。

構文

object.**AppendRouteName**(*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。
Flag	ENUM_ROUTENAME	経由交差点フラグの組み合わせ

解説

AppendRouteName メソッドは、ToNodeCode プロパティで設定されているノードへの２点間の経由交差点情報を作成・追加します。本メソッドをコールする前に CalcRoute メソッドを呼び出しルート計算を行ってください。作成・追加された経由交差点情報は、コンポーネント内部に蓄積されます。経由交差点情報を取得するには、GetRouteName メソッドを使用してください。

経由交差点の情報は、引数で指定する経由交差点フラグにより次のように異なります。

経由交差点フラグ	意味
なし(標準)	発着地、中継点を出力します。
ROUTENAME_NODE	交差点名称を出力します。
ROUTENAME_LINK	道路名称を出力します。
ROUTENAME_FARE	インターチェンジ名称と通行料金を出力します。
ROUTENAME_XY	交差点とインターチェンジのＸＹ座標を出力します。

また、経由交差点フラグによって RouteName オブジェクトの更新されるプロパティも次のように異なります。

プロパティ	経由交差点フラグ				
	なし (標準)	ROUTE NAME _NODE	ROUTE NAME _LINK	ROUTE NAME _FARE	ROUTE NAME _XY
PointFlag	○(1)	○(0)	○(0)		
LinkFlag	○(0)	○(0)	○(1)		
NLCode	○	○	○		
Name		○	○		
Time	○	○	×		
DSecTime	○	○	×		
Distance	○	○	×		
ICType				○	
Toll_SS				○	
Toll_S				○	
Toll_M				○	
Toll_L				○	
Toll_LL				○	
Longitude			×		○
Latitude			×		○

上表の○印は、プロパティが更新されることを意味します。×印は、他のフラグにかかわらずプロパティは更新されないことを意味します。（ROUTENAME_LINK フラグが設定されたことにより出力された RouteName オブジェクトの Time、DSecTime、Distance プロパティは、更新されず必ず 0 になります）

経由交差点フラグは、ENUM_ROUTENAME 型として定義されています。ENUM_ROUTENAME は、ENUM 定数一覧を参照してください。

GetRouteName メソッド

経由交差点情報を取得します。

構文

object.**GetRouteName**(*RouteNames*)

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。
RouteNames	IRouteNames	経由交差点情報を格納する RouteNames コレクションオブジェクト

解説

GetRouteName メソッドは、AppendRouteName メソッドを使用し、作成・追加した経由交差点情報を RouteNames コレクションに格納します。

コレクション内の最後の RouteName オブジェクト(最終着地)の Toll_SS、Toll_S、Toll_M、Toll_L、Toll_LL プロパティは、それぞれの通行料金の合計金額が格納されます。

ClearTollRoadDetail メソッド

有料道路明細情報をクリアします。

構文

object.**ClearTollRoadDetail**

項目	データ型	内容

なし

解説

有料道路明細情報を取得する前に、必ず本メソッドをコールし、有料道路明細情報のクリアを行ってください。

AppendTollRoadDetail メソッド

有料道路明細情報を作成・追加します。

構文

object.**AppendTollRoadDetail**(*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。
Flag	ENUM_TOLLROADDETAIL	有料道路明細フラグ(拡張用パラメータ)

解説

AppendTollRoadDetail メソッドは、ToNodeCode プロパティで設定されているノードへの 2 点間の有料道路明細情報を作成・追加します。本メソッドをコールする前に CalcRoute メソッドを呼び出しルート計算を行ってください。作成・追加された有料道路明細情報は、コンポーネント内部に蓄積されます。経由交差点情報を取得するには、GetTollRoadDetail メソッドを使用してください。

また、現バージョンでは、有料道路明細フラグには TOLLROADDETAIL_NONE を設定してください。

GetTollRoadDetail メソッド

有料道路明細情報を取得します。

構文

object.**GetTollRoadDetail**(*TollRoadDetails*)

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。
TollRoadDetails	ITollRoadDetails	有料道路明細情報を格納する TollRoadDetails コレクションオブジェクト

解説

GetTollRoadDetail メソッドは、TollRoadDetail メソッドを使用し、作成・追加した有料道路明細情報を TollRoadDetails コレクションに格納します。

４．ENUM 定数一覧

４．１　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

４．２　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

４．３　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

４．４　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

４．５　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

４．６　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	計算用道路データのロードに失敗
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	計算用道路データが存在しない
ERR_UnloadRoadNetwork	計算用道路データのアンロードに失敗
ERR_NotLoadedRoadNetwork	計算用道路データ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_CalcRouteLocsCount2	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_CalcRouteFailure2	CalcRoute2 メソッド失敗(DLL レベル)

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_GetRouteDetail	GetRouteDetail メソッド異常終了
ERR_GetRouteDetailFailure	GetRouteDetail 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_MakePolygonFailure	MakePolygon メソッド失敗(DLL レベル)
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_MakePolygonFailure2	MakePolygon2 メソッド失敗(DLL レベル)
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 ・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	計算用道路データ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

４．７　経由交差点フラグ(ENUM_ROUTENAME 型)

ENUM_ROUTENAME 型は、経由交差点フラグを表します。

値	内容
ROUTENAME_NODE	交差点名称を出力します
ROUTENAME_LINK	道路名称を出力します
ROUTENAME_MORELINK	将来予約用定数(R.F.U.)
ROUTENAME_XY	交差点とインターチェンジの経度緯度を出力します
ROUTENAME_FARE	インターチェンジ名称と通行料金を出力します
ROUTENAME_LONGNAME	将来予約用定数(R.F.U.)
ROUTENAME_TRF1	将来予約用定数(R.F.U.)
ROUTENAME_DISTKEY	将来予約用定数(R.F.U.)
ROUTENAME_DETAIL	将来予約用定数(R.F.U.)

４．８　有料道路明細フラグ(ENUM_TOLLROADDETAIL 型)

ENUM_TOLLROADDETAIL 型は、有料道路明細フラグを表します。

値	内容
TOLLROADDETAIL_NODE	将来予約用定数(R.F.U.)

４．９　メモリオプションフラグ(ENUM_MEMORYOPTION 型)

ENUM_MEMORYOPTION 型は、メモリオプションフラグを表します。

値	内容
MEMORYOPTION_NORMAL	メモリ使用量：通常
MEMORYOPTION_HIGH	メモリ使用量：約 1.5 倍
MEMORYOPTION_FULL	メモリ使用量：約 1.8 倍

５．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。

５．１　２点間計算

```
'DistCalcParent 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 プロパティの値を MsgBox で表示します。
Private Sub Command1_Click()
    Dim DCParent As ActCalcM.DistCalcParent 'DistCalcParent オブジェクト

    'オブジェクトを初期化します
    Set DCParent = New ActCalcM.DistCalcParent

    '計算用道路データフォルダを設定します（例）
    DCParent.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
    DCParent.LoadRoadNetwork

    'プロパティを設定します
    DCParent.CalcKind = CALCKIND_TIME           '時間最短
    DCParent.UseHighway = USEHIGHWAY_YES        '高速道路使用
    DCParent.FromNodeCode = 13000000036205#     '始点ノードコード（例）
    DCParent.ToNodeCode = 13000000076996#       '終点ノードコード（例）

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

    '結果を表示します
    MsgBox "Time=" + Str$(DCParent.Time) + Chr$(13) + _
           "DSecTime=" + Str$(DCParent.DSecTime) + Chr$(13) + _
           "Distance=" + Str$(DCParent.Distance)

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

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

５．２　最短ルート計算

```
'DistCalcParent 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 DCParent        As ActCalcM.DistCalcParent 'DistCalcParent 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 DCParent = New ActCalcM.DistCalcParent
    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）
    DCParent.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
    DCParent.LoadRoadNetwork

    'プロパティを設定します
    DCParent.CalcKind = CALCKIND_TIME          '時間最短
    DCParent.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
        DCParent.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
    DCParent.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
    DCParent.UnloadRoadNetwork

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

５．３　到達圏計算

```
'DistCalcParent オブジェクトの CalcArea メソッド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 i           As Integer                  'ループカウンタ
    Dim szBuff      As String                   '表示用
    Dim DCParent    As ActCalcM.DistCalcParent  'DistCalcParent オブジェクト
    Dim AreaPoints  As ActDataX.Points          'Points オブジェクト

    'オブジェクトを初期化します
    Set DCParent = New ActCalcM.DistCalcParent
    Set AreaPoints = New ActDataX.Points

    '計算用道路データフォルダを設定します（例）
    DCParent.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
    DCParent.LoadRoadNetwork

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

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

５．４　子オブジェクトでの２点間計算

```
'DistCalcParent オブジェクトで計算用道路データをロードし、
'DistCalcChild 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 プロパティの値を MsgBox で表示します。
Private Sub Command1_Click()
    Dim DCParent    As ActCalcM.DistCalcParent    'DistCalcParent オブジェクト
    Dim DCChild     As ActCalcM.DistCalcChild     'DistCalcChild オブジェクト


    '親オブジェクトを初期化します
    Set DCParent = New ActCalcM.DistCalcParent

    '計算用道路データフォルダを設定します（例）
    DCParent.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
    DCParent.LoadRoadNetwork

    '子オブジェクトを初期化します
    Set DCChild = New ActCalcM.DistCalcChild

    '親オブジェクトにアタッチします
    DCChild.Attach DCParent

    'プロパティを設定します
    DCChild.CalcKind = CALCKIND_TIME          '時間最短
    DCChild.UseHighway = USEHIGHWAY_YES       '高速道路使用
    DCChild.FromNodeCode = 13000000036205#    '始点ノードコード（例）
    DCChild.ToNodeCode = 13000000076996#      '終点ノードコード（例）

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

    '結果を表示します
    MsgBox "Time=" + Str$(DCChild.Time) + Chr$(13) + _
           "DSecTime=" + Str$(DCChild.DSecTime) + Chr$(13) + _
           "Distance=" + Str$(DCChild.Distance)

    '親オブジェクトからデタッチします
    DCChild.Detach

    '子オブジェクトを解放します
    Set DCChild = Nothing

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

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

前ページでのコーディング例では、Command1_Click イベントで親オブジェクトの操作と子オブジェクトの計算を同時に行っています。実際の運用システムではスレッドを作成し、その中で子オブジェクトを作成し、計算を実行するようなコーディングを行うべきです。そうすることで複数クライアントからの計算要求の処理や、マルチプロセッサやメモリといったリソースを最大限に活用することが出来るようになります。

（メモ）