

# **ACT** 距離計算コア

Version 5.2

距離計算コアライブラリ  
関数リファレンス

ACT 距離計算コア

Version 5.X

距離計算コアライブラリ・関数リファレンス

2001年01月04日 初版発行

2002年02月01日 第2版発行 (Version 5.1)

2002年12月01日 第3版発行 (Version 5.2)

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

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) 1996-2002 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 距離計算コア**  
Version 5.2

**距離計算コアライブラリ・関数リファレンス**  
目次

1 . はじめに . . . . .	1
2 . 距離計算コアライブラリの稼働環境 . . . . .	2
3 . 関数の種類 . . . . .	3
4 . 関数一覧 . . . . .	4
5 . 稼働環境設定関数 . . . . .	7
6 . 計算モード開始 / 終了関数 . . . . .	15
7 . 計算実行関数 . . . . .	19
8 . 計算結果取得関数 . . . . .	26
9 . リンク速度管理関数 . . . . .	44
10 . 通行料金計算関数 . . . . .	51
11 . その他の関数 . . . . .	56
12 . 定数一覧 . . . . .	62
13 . エラーログ . . . . .	64



## 1 . はじめに

本書は、距離計算コアライブラリ Version 5.2 の 32bit Windows DLL レベルでの関数コール手順を示したものです。

本書の関数および定数の表記は Win32 API の標準的な C 言語の表記に準じています。

### ( 1 ) Version 5.1 の新機能

Version 5.1 では、リンクの形状忠実なルート図形を求めることができるようになりました。追加された関数は次の 2 関数です。

```
ACT_ADMN_IsShapeFilesExist
ACT_DDIJ_GetCalcRouteDetail
```

### ( 2 ) Version 5.2 の新機能

Version 5.2 では、新しい計算ロジックの採用により計算速度が約 30% 向上したほか、経由交差点一覧の表示、通行料金の計算が可能になりました。

新しい計算ロジックでは次の点が改良されています。

時間最短計算で時間が等しい場合、距離の短い方のルートを選択します。同様に、距離最短計算で距離が等しい場合、時間の短い方のルートを選択します（以前のバージョンでは、キーが等しい場合、先に接続しているリンクが選択されます）。

高速道路を使用しない計算は、高速道路リンクを通過しないように計算されます（以前のバージョンでは、接続しているリンクが全て高速道路であるノードを通過しないように計算されます）。

経由交差点一覧関連では次の 2 関数が追加されています。

```
ACT_DDIJ_GetRouteNameFileName
ACT_DDIJ_OutputCalcRouteName
```

通行料金計算関連では次の 4 関数が追加されています。

```
ACT_ADMN_IsFareExist
ACT_DDIJ_CalcTollRoadFare
ACT_DDIJ_GetTollRoadFareFileName
ACT_DDIJ_CalcTollRoadFareDetail
```

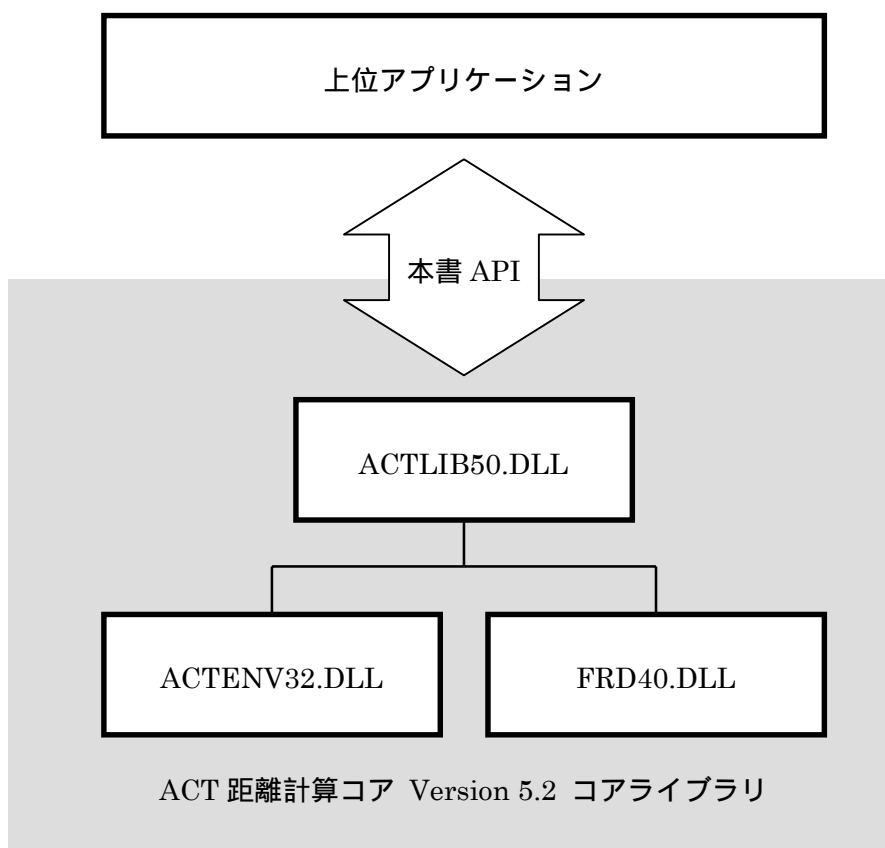
計算モード開始関数として下記の関数が追加されています。この関数でメモリ・オプションを指定することで、通行料金計算結果の取得、詳細ルートの取得に要する時間を短縮することができます。

```
ACT_DDIJ_LoadRoadNetworksEx
```

## 2 . 距離計算コアライブラリの稼動環境

距離計算コアライブラリは、ACTLIB50.DLL、ACTENV32.DLL、FRD40.DLL の3モジュールで構成されています。本書で解説する距離計算機能に関連する関数はすべて ACTLIB50.DLL からエクスポートされていますが、ACTLIB50.DLL が正常に機能するにはこれら3モジュールすべてとライセンス環境ファイル ( actdll.inf、 distcalc.lic ) が必要です。

また、ACT 距離計算コアが計算に使用する計算用道路データが事前に作成されていることが必要です (本ライブラリには、計算用道路データ作成機能はありません)。



ACT 距離計算コアライブラリを LoadLibrary 関数等を使用してプロセスのメモリスペースにマッピングする場合には、FRD40.DLL、ACTENV32.DLL、ACTLIB50.DLL の順番でロードするようにしてください。

### 3 . 関数の種類

本書では、距離計算コアライブラリの関数群を、以下の6つのカテゴリーに分けて解説しています。

項番	カテゴリー	解説
1	稼働環境設定関数	距離計算オブジェクトの生成 / 廃棄、計算用道路データの保存フォルダの指定を行う関数
2	計算モード開始 / 終了関数	距離計算を実行することができる状態（計算モード）を開始 / 終了する関数
3	計算実行関数	距離計算を実行する関数
4	計算結果取得関数	上記の計算実行関数の計算結果（所要時間、道のり、ルート情報）を取得する関数
5	リンク速度管理関数	道路速度の取得 / 変更を行う関数
6	通行料金計算関数	有料道路の通行料金を計算する関数
7	その他の関数	道路点の座標や、最寄道路点を取得する関数

距離計算コアライブラリの関数は、「稼働環境設定関数」で生成 / 廃棄される距離計算コアオブジェクトによって、機能と計算用道路データの連続性と独立性を保持する構造になっており、ほとんどの関数が下記のような関数定義となっています。

```
type WINAPI ACT_XXXX_Function( HANDLE hLibObject, ...other operators... )
HANDLE hLibObject          距離計算コアオブジェクトのハンドル
```

距離計算コアオブジェクトは、計算用道路データ、計算結果、計算モード等を保持しているため、関数に引き渡される距離計算コアオブジェクトのハンドルが不正である場合には、各関数の動作は保証されません。

## 4 . 関数一覧

下表は、カテゴリ別の関数一覧です。

(注) 【Ver 5.1 新規関数】【Ver 5.2 新規関数】と記された関数は、Version 5.1、Version 5.2 で新規に追加された関数です。

### ( 1 ) 稼働環境設定関数

関数名	解説
ACT_ADMN_CreateLibObject	距離計算コアオブジェクトを生成します
ACT_ADMN_AttachRoadNetwork	距離計算コアオブジェクトでロードされている計算用道路データを別の距離計算オブジェクトで使用できるようにします
ACT_ADMN_DeleteLibObject	距離計算コアオブジェクトを廃棄します
ACT_ADMN_SetWorkDir	距離計算コアオブジェクトに計算用道路データが保存されているフォルダを指定します
ACT_ADMN_GetWorkDir	距離計算コアオブジェクトに現在指定されている計算用道路データのフォルダのフルパスを取得します
ACT_ADMN_IsCalcNetworkExist	距離計算コアオブジェクトに現在指定されている計算用道路データのフォルダに計算用道路データが存在するかどうかチェックします

### ( 2 ) 計算モード開始 / 終了関数

関数名	解説
ACT_DDIJ_LoadRoadNetworks	計算用道路データの一部をメモリにロードし (ACT1 型)、距離計算を実行できる状態にします
ACT_DDIJ_LoadRoadNetworksEx	計算用道路データの一部をメモリにロードし、距離計算を実行できる状態にします。 【Ver 5.2 新規関数】
ACT_DDIJ_UnloadRoadNetworks	計算用道路データ用に確保されたメモリ領域を解放します (ACT1 型)
ACT_DDIJ_IsRoadNetworksLoaded	計算用道路データの一部がメモリにロードされ距離計算可能状態にあるかどうかチェックします

(注) 距離計算コア Version 5.2 では、右左折禁止を考慮する計算方法 (TRF1 型) は現在のところサポートしておりません。

## ( 3 ) 計算実行関数

関数名	解説
ACT_DDIJ_CalcRoute	指定された 2 点間の最短時間経路を計算します
ACT_DDIJ_CalcFullRoute	指定された起点からの最短時間経路を計算します
ACT_DDIJ_CalcFullRouteEx	指定された起点から、指定時間 ( 分単位 ) 以内の最短時間経路を計算します
ACT_DDIJ_CalcFullRouteExDSec	指定された起点から、指定時間 ( 1/10 秒単位 ) 以内の最短時間経路を計算します
ACT_DDIJ_CalcRouteDist	指定された 2 点間の最短距離経路を計算します
ACT_DDIJ_CalcFullRouteDist	指定された起点からの最短距離経路を計算します
ACT_DDIJ_CalcFullRouteDistEx	指定された起点から、距離 ( m単位 ) 以内の最短距離経路を計算します

## ( 4 ) 計算結果取得関数

関数名	解説
ACT_DDIJ_GetCalcTime	指定された点までの所要時間 ( 分単位 ) を求めます
ACT_DDIJ_GetCalcDSecTime	指定された点までの所要時間 ( 1/10 秒単位 ) を求めます
ACT_DDIJ_GetCalcDistance	指定された点までの道のり ( m単位 ) を求めます
ACT_DDIJ_GetCalcRouteXY	指定された点までのルート情報 ( XY 座標の系列 ) を求めます
ACT_DDIJ_GetCalcRouteNode	指定された点までのルート情報 ( ノードコードの系列 ) を求めます
ACT_DDIJ_GetCalcRouteLink	指定された点までのルート情報 ( リンクコードの系列 ) を求めます
ACT_DDIJ_GetCalcRouteNodeLink	指定された点までのルート情報 ( ノードコードとリンクコードの系列 ) を求めます
ACT_DDIJ_GetAttribTimeDist	指定された区間種別部分の所要時間と道のりを求めます
ACT_ADMN_IsShapeFilesExist	詳細ルート情報取得が可能かどうかチェックします 【Ver 5.1 新規関数】
ACT_DDIJ_GetCalcRouteDetail	詳細ルート情報 ( X Y 座標の系列 ) を取得します 【Ver 5.1 新規関数】
ACT_DDIJ_GetRouteNameFileName	経路交差点名称ファイルのフルパスを取得します 【Ver 5.2 新規関数】
ACT_DDIJ_OutputCalcRouteName	経路交差点名称を出力します 【Ver 5.2 新規関数】

( 5 ) リンク速度管理関数

関数名	解説
ACT_ADMN_ChangeBothSpeedEx	指定されたリンクの速度（両方向）を設定します
ACT_ADMN_GetSpeed	指定されたリンクの速度（片方向）を取得します
ACT_ADMN_GetSpeedEx	指定されたリンクの速度（両方向）を取得します
ACT_ADMN_CountTempSpeed	一時的速度が設定されているリンク数を取得します
ACT_ADMN_CommitTempSpeed	一時的速度を恒久的速度として保存します
ACT_ADMN_ClearTempSpeed	一時的速度変更を取り消し元の速度（恒久的速度）に戻します

( 6 ) 通行料金計算関数

関数名	解説
ACT_ADMN_IsFareExist	通行料金計算が可能かどうかチェックします 【Ver 5.2 新規関数】
ACT_DDIJ_CalcTollRoadFare	通行料金を計算します 【Ver 5.2 新規関数】
ACT_DDIJ_GetTollRoadFareFileName	有料道路明細ファイルのフルパスを取得します 【Ver 5.2 新規関数】
ACT_DDIJ_CalcTollRoadFareDetail	有料道路明細ファイル（経由インターチェンジの一覧）を出力します 【Ver 5.2 新規関数】

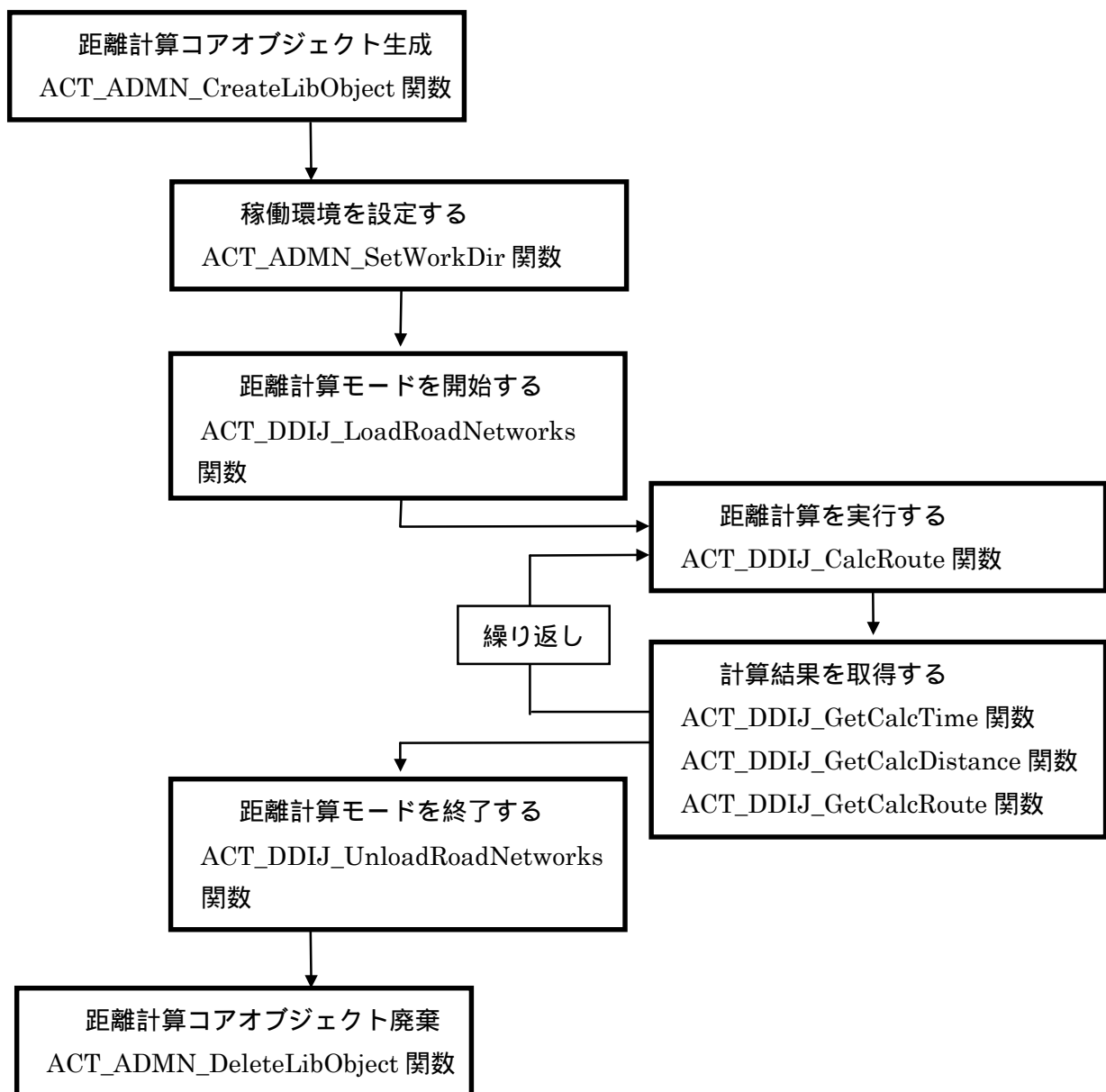
( 7 ) その他の関数

関数名	解説
ACT_DDIJ_GetNodeX	ノードコードで指定されたノードのX座標を取得します
ACT_DDIJ_GetNodeY	ノードコードで指定されたノードのY座標を取得します
ACT_DDIJ_IsHighwayNode	指定されたノードが高速道路上の点であるかどうかを判定します
ACT_DDIJ_GetNearestNodeEx	指定された座標に指定された範囲内で最も近い道路点を求めます
ACT_DDIJ_GetNodeNumber	計算用道路データのノード数を求めます
ACT_DDIJ_GetLinkNumber	計算用道路データのリンク数を求めます

## 5 . 稼働環境設定関数

距離計算コアライブラリの関数は、「稼働環境設定関数」で生成 / 廃棄される距離計算コアオブジェクトによって、各関数と計算用道路データの連続性と独立性を保持する構造になっています。距離計算を行うには、まず、ACT\_ADMN\_CreateLibObject 関数で距離計算オブジェクトを生成し、これに対して環境設定、計算実行を行います。また、不要になった距離計算コアオブジェクトは、ACT\_ADMN\_DeleteLibObject 関数で廃棄します。

下図は、簡単な2点間の距離計算を行い、所要時間、道のり、ルートを取り出すシーケンスを示したものです。



## **HANDLE WINAPI ACT\_ADMN\_CreateLibObject( void )**

本関数は、距離計算コアオブジェクトを生成します。

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

---

なし

### **戻り値**

距離計算コアオブジェクトのオブジェクトハンドル。  
オブジェクトの生成に失敗した場合は NULL。

### **備考**

本関数で生成された距離計算コアオブジェクトを、各関数に引数として引き渡します。  
本関数で生成された距離計算コアオブジェクトは、ACT\_ADMN\_DeleteLibObject 関数で廃棄します。

**BOOL WINAPI ACT\_ADMN\_AttachRoadNetwork( HANDLE hDestLibObject,  
HANDLE hSrcLibObject )**

HANDLE hDestLibObject	計算用道路ネットワークのアタッチ先距離計算コアオブジェクトのハンドル(サブオブジェクト)
HANDLE hSrcLibObject	アタッチする計算用道路ネットワークをロードしている距離計算コアオブジェクトのハンドル(ルートオブジェクト)

本関数は、距離計算コアオブジェクトにロードされている計算用道路ネットワークを、別の距離計算コアオブジェクトで使用できるようにします。

パラメータ	説明
hDestLibObject	(IN)計算用道路ネットワークのアタッチ先距離計算コアオブジェクトのハンドル(サブオブジェクト)
hSrcLibObject	(IN)アタッチする計算用道路ネットワークをロードしている距離計算コアオブジェクトのハンドル(ルートオブジェクト)

**戻り値**

計算用道路ネットワークが正常にアタッチされた場合は TRUE。  
アタッチ中にエラーが発生した場合は FALSE。

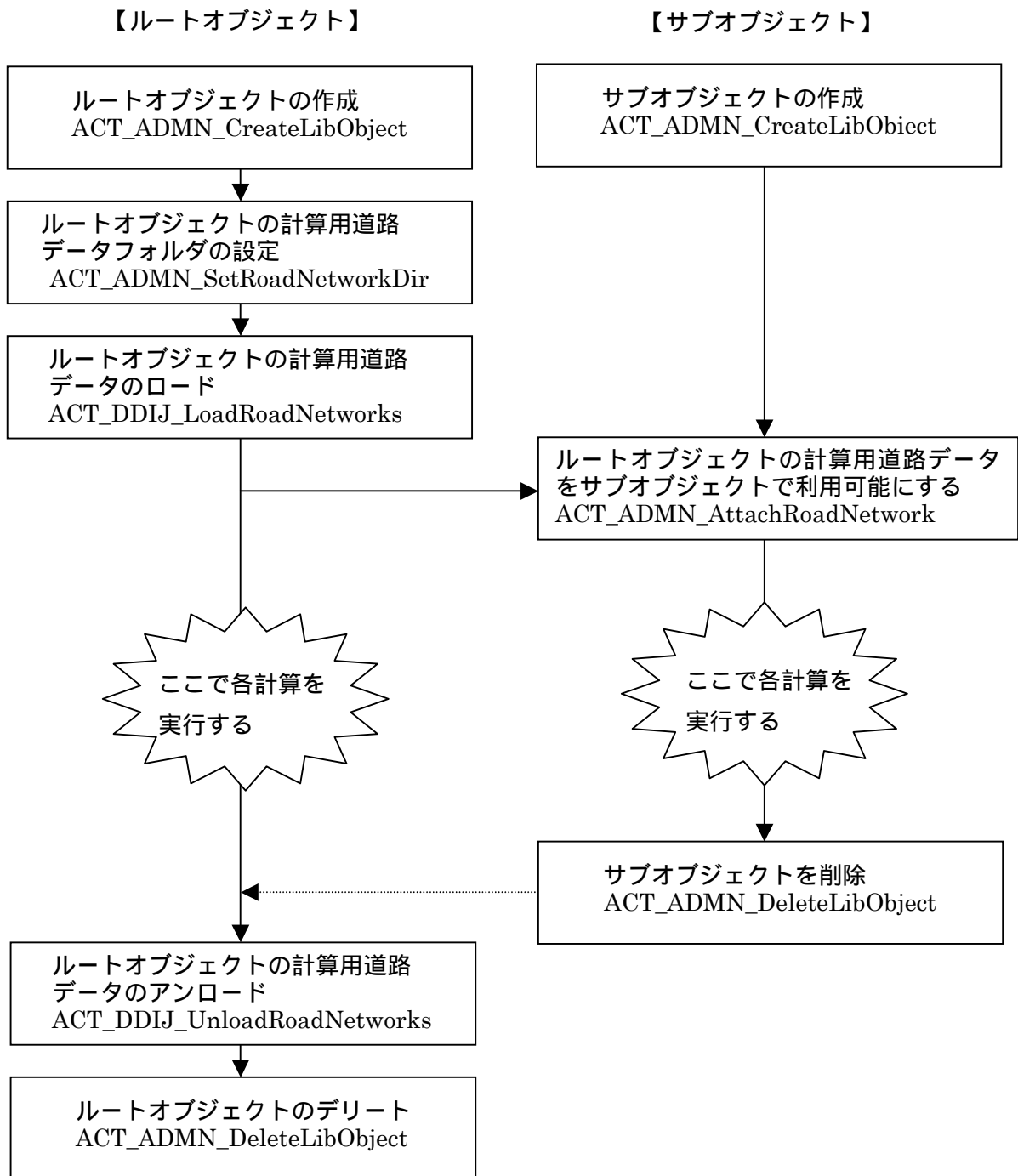
**備考**

hDestLibObject、hSrcLibObject とともに事前に ACT\_ADMN\_CreateLibObject 関数で生成された距離計算オブジェクトでなくてはなりません。

サブオブジェクト(hDestLibObject)に計算用道路データをアタッチする前に、ルートオブジェクト(hSrcLibObject)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、ルートオブジェクト(hSrcLibObject)の計算用道路データをアンロードする前に、サブオブジェクト(hDestLibObject)を廃棄しなくてはなりません。尚、サブオブジェクト(hDestLibObject)を廃棄しても、アタッチされているルートオブジェクト(hSrcLibObject)の計算用道路データはアンロードされません。

ルートオブジェクトとサブオブジェクトを上手く使うことで、計算用道路データのロードの時間と消費メモリを節約することができます。

次頁は、ルートオブジェクトとサブオブジェクトの関係を示したものです。



**BOOL WINAPI ACT\_ADMN\_DeleteLibObject( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

本関数は、距離計算コアオブジェクトを廃棄します。

**パラメータ****説明**

パラメータ	説明
hLibObject	(IN)廃棄する距離計算コアオブジェクトのハンドル

**戻り値**

距離計算コアオブジェクトが廃棄された場合は TRUE。

廃棄中にエラーが発生した場合は FALSE。

**備考**

本関数コール後は hLibObject は正常なハンドルではなくなります。本関数が失敗した場合は hLibObject が正常なハンドルであるかどうかは保証されません。

**BOOL WINAPI ACT\_ADMN\_SetWorkDir( HANDLE hLibObject, LPSTR lpszWorkDir )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

LPSTR lpszWorkDir      計算用道路データが保存されているフォルダのフルパス

本関数は、距離計算コアオブジェクトに、計算対象の計算用道路データが保存されているフォルダを設定します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
lpszWorkDir	(IN)計算用道路データが保存されているフォルダのフルパスを格納した文字列へのポインタ

**戻り値**

設定が正常に完了した場合は TRUE。正常に完了しない場合は FALSE。

**備考**

距離計算コアライブラリのすべての関数は、本関数で指定されたフォルダ下にある計算用道路データに対して演算を行います。尚、この指定は、距離計算コアオブジェクトが廃棄されると消去されます。

**BOOL WINAPI ACT\_ADMN\_GetWorkDir( HANDLE hLibObject, LPSTR lpszWorkDir )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル  
LPSTR lpszWorkDir      計算用道路データが保存されているフォルダのフルパスを取得する文字列へのポインタ

本関数は、距離計算コアオブジェクトに指定されている計算用道路データのフォルダのフルパスを取得します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
lpszWorkDir	(OUT)計算用道路データが保存されているフォルダのフルパスを取得する文字列へのポインタ。文字列のサイズは最低でも MAX_PATH 以上の長さが必要です。

**戻り値**

正常に取得した場合は TRUE。正常に取得できない場合は FALSE。

**備考**

本関数は、現在指定されている計算用道路データのフォルダのフルパスを取得します

**BOOL WINAPI ACT\_ADMN\_IsCalcNetworkExist( HANDLE hLibObject )**

HANDLE hLibObject     距離計算コアオブジェクトのハンドル

本関数は、距離計算コアオブジェクトに指定されている計算用道路データのフォルダに計算用道路データが存在するかどうかチェックします。

**パラメータ**

**説明**

---

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

**戻り値**

計算用道路データが存在する場合は TRUE。存在しない場合は FALSE。

**備考**

本関数は、ACT\_ADMN\_SetWorkDir 関数で設定されているフォルダに、計算用道路データが存在するかどうかをチェックします。ACT\_ADMN\_SetWorkDir 関数による計算用道路データ保管フォルダの指定が完了していない場合には、本関数は FALSE を返します。

## 6 . 計算モード開始 / 終了関数

距離計算を実際に行うには、距離計算コアオブジェクトに対して、ACT\_DDIJ\_LoadRoadNetworks 関数をコールして、ACT\_DDIJ\_SetWorkDir 関数で指定したフォルダ内の計算用道路データの一部をメモリにロードする必要があります。

所定の計算作業が終了したら、ACT\_DDIJ\_UnloadRoadNetworks 関数をコールしてメモリを解放します。なお、この関数をコールしてメモリを解放すると、メモリ内に保持されている計算結果は失われます。

**BOOL WINAPI ACT\_DDIJ\_LoadRoadNetworks( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

本関数は、計算用道路データの一部をメモリにロードし (ACT1 型)、距離計算を実行できる状態にします。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル

### 戻り値

正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

### 備考

右左折禁止を考慮しない方法 (ACT1 型) で計算をするために、計算用道路データの一部をメモリにロードします。計算終了後は、ACT\_DDIJ\_UnloadRoadNetworks 関数をコールしてメモリを解放してください。

尚、距離計算コア Version 5.2 では、右左折禁止を考慮する計算方法 (TRF1 型) は現在のところサポートしておりません。

**BOOL WINAPI ACT\_DDIJ\_LoadRoadNetworksEx( HANDLE hLibObject , UINT unModeFlag )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル  
 UINT unModeFlag      計算モード、メモリ・オプションを指定するフラグ

本関数は、計算用道路データの一部をメモリにロードし、距離計算を実行できる状態にします。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
unModeFlag	(IN)計算モード、メモリ・オプションを指定するフラグ。ゼロまたは下記のいずれかの値を指定します。ゼロは通常のメモリ消費量を示します。 ACTLIB_MODE_HIMEMORY(0x0080)：通常の約 1.5 倍のメモリ消費 ACTLIB_MODE_FULLMEMORY(0x0800)：通常の約 1.8 倍のメモリ消費

**戻り値**

正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

**備考**

右左折禁止を考慮しない方法 (ACT1 型) で計算をするために、計算用道路データの一部をメモリにロードします。計算終了後は、ACT\_DDIJ\_UnloadRoadNetworks 関数をコールしてメモリを解放してください。

unModeFlag = 0 のとき、通常のメモリ消費量となります。

unModeFlag = ACTLIB\_MODE\_HIMEMORY のとき、通常の約 1.5 倍のメモリを消費しますが、通常の場合に比べ下記の関数の計算結果取得時間が約半分になります。

ACT_DDIJ_GetCalcRouteLink	ACT_DDIJ_GetCalcRouteNodeLink
ACT_DDIJ_OutputCalcRouteName	ACT_DDIJ_CalcTollRoadFare
ACT_DDIJ_CalcTollRoadFareDetail	

unModeFlag = ACTLIB\_MODE\_FULLMEMORY のとき、通常の約 1.8 倍のメモリを消費しますが、通常の場合に比べ、上記の各関数の計算結果取得時間が約半分になるほか、下記の関数の計算結果取得時間が約 1 / 3 になります。

ACT\_DDIJ\_GetCalcRouteDetail

尚、距離計算コア Version 5.2 では、右左折禁止を考慮する計算方法 (TRF1 型) は現在のところサポートしていません。

**void WINAPI ACT\_DDIJ\_UnloadRoadNetworks( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

計算用道路データ用に確保されたメモリ領域を解放します (ACT1 型)。

#### パラメータ

#### 説明

---

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

#### 戻り値

なし。

#### 備考

本関数は、ACT\_DDIJ\_LoadRoadNetworks 関数または ACT\_DDIJ\_LoadRoadNetworksEx 関数でメモリにロードした計算用道路データを、メモリから解放します。これによって、メモリ内に保持されていた計算結果は消去されます。

## BOOL WINAPI ACT\_DDIJ\_IsRoadNetworksLoaded( HANDLE hLibObject )

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

本関数は、計算用道路データの一部がメモリにロードされ距離計算可能状態にあるかどうかチェックします。

### パラメータ

### 説明

---

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

### 戻り値

計算用道路データの一部がメモリにロードされている場合は TRUE。ロードされていない場合は FALSE。

### 備考

計算用道路データの一部がメモリにロードされ距離計算可能状態にあるかどうかチェックします。ACT1 型、TRF1 型どちらでも利用できます。

尚、距離計算コア Version 5.2 では、右左折禁止を考慮する計算方法 (TRF1 型) は現在のところサポートしておりません。

## 7 . 計算実行関数

距離計算を実行するには、ACT\_DDIJ\_LoadRoadNetworks 関数をコールして、距離計算モードに移行した後に、以下の計算実行関数をコールします。

距離計算の計算方法には、「所要時間」を最短とする最短時間経路を求める方法と、「道のり」を最短とする最短距離経路を求める方法があり、それぞれの場合について別々の関数が用意されています。

計算の結果は、次節の「計算結果取得関数」で取得します。計算実行関数で計算した結果は、次に計算実行関数をコールして計算を行った場合、または、ACT\_DDIJ\_UnloadRoadNetworks 関数をコールして計算モードを終了するか、ACT\_ADMN\_DeleteLibObject 関数をコールして当該距離計算オブジェクトが廃棄されるまで、メモリ上に保持されます。

```

BOOL WINAPI ACT_DDIJ_CalcRoute( HANDLE hLibObject,
                                double ldFromNodeCode, double ldToNodeCode,
                                BOOL bNotUseHighWay )
HANDLE hLibObject           距離計算コアオブジェクトのハンドル
double ldFromNodeCode      発地点のノードコード
double ldToNodeCode       着地点のノードコード
BOOL bNotUseHighWay       高速道路非使用フラグ
    
```

本関数は、指定された 2 点間の最短時間経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldFromNodeCode	(IN)発地点のノードコード
ldToNodeCode	(IN)着地点のノードコード
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点または着地点の道路点を示すノードコードに不備がある場合、これらの点が計算用道路データに存在しない場合、発地点から着地点に到達できなかった場合には、FALSE を返します。

### 備考

指定された 2 点間の最短時間経路を計算します。計算結果は、計算結果取得関数で取得します。

**BOOL WINAPI ACT\_DDIJ\_CalcFullRoute( HANDLE hLibObject, double ldStartNodeCode, BOOL bNotUseHighWay )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル  
 double ldStartNodeCode                発地点のノードコード  
 BOOL bNotUseHighWay                  高速道路非使用フラグ

指定された起点からの最短時間経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldStartNodeID	(IN)発地点のノードコード
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

#### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点の道路点を示すノードコードに不備がある場合、この点が計算用道路データに存在しない場合、この点からどの道路点にも到達できない場合には、FALSE を返します。

#### 備考

指定された発地点から、計算用道路データ内のすべての道路点までの最短時間経路を計算します。計算結果は、計算結果取得関数で取得します。

**BOOL WINAPI ACT\_DDIJ\_CalcFullRouteEx( HANDLE hLibObejct,  
double ldStartNodeCode,  
long lnThreshold, BOOL bNotUseHighWay )**

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldStartNodeCode	発地点のノードコード
long lnThreshold	到達時間
BOOL bNotUseHighWay	高速道路非使用フラグ

指定された起点から、指定時間（分単位）以内の最短時間経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldStartNodeCode	(IN)発地点のノードコード
lnThreshold	(IN)到達時間。分単位。
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

#### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点の道路点を示すノードコードに不備がある場合、この点が計算用道路データに存在しない場合、この点からどの道路点にも到達できない場合には、FALSE を返します。

#### 備考

指定された発地点から、指定された時間内に到達できるすべての道路点までの最短時間経路を計算します。計算結果は、計算結果取得関数で取得します。

**BOOL WINAPI ACT\_DDIJ\_CalcFullRouteExDSec(HANDLE hLibObject,  
double ldStartNodeCode, long lnThreshold,  
BOOL bNotUseHighWay)**

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldStartNodeCode	発地点のノードコード
long lnThreshold	到達時間
BOOL bNotUseHighWay	高速道路非使用フラグ

指定された起点から、指定時間（1/10 秒単位）以内の最短時間経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldStartNodeCode	(IN)発地点のノードコード
lnThreshold	(IN)到達時間。1/10 秒単位。
bNotUseHighWay	(IN)高速道路使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

#### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点の道路点を示すノードコードに不備がある場合、この点が計算用道路データに存在しない場合、この点からどの道路点にも到達できない場合には、FALSE を返します。

#### 備考

指定された発地点から、指定された時間内に到達できるすべての道路点までの最短時間経路を計算します。計算結果は、計算結果取得関数で取得します。

```

BOOL WINAPI ACT_DDIJ_CalcRouteDist( HANDLE hLibObject,
                                double ldFromNodeCode, double ldToNodeCode,
                                BOOL bNotUseHighWay )
HANDLE hLibObject                距離計算コアオブジェクトのハンドル
double ldFromNodeCode            発地点のノードコード
double ldToNodeCode              着地点のノードコード
BOOL bNotUseHighWay              高速道路非使用フラグ

```

本関数は、指定された 2 点間の最短距離経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldFromNodeCode	(IN)発地点のノードコード
ldToNodeCode	(IN)着地点のノードコード
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点または着地点の道路点を示すノードコードに不備がある場合、これらの点が計算用道路データに存在しない場合、発地点から着地点に到達できなかった場合には、FALSE を返します。

### 備考

指定された 2 点間の最短距離経路を計算します。計算結果は、計算結果取得関数で取得します。

**BOOL WINAPI ACT\_DDIJ\_CalcFullRouteDist( HANDLE hLibObject,  
double ldStartNodeCode,  
BOOL bNotUseHighWay )**

HANDLE hLibObject                    距離計算コアオブジェクトのハンドル  
double ldStartNodeCode                起点のノードコード  
BOOL bNotUseHighWay                  高速道路非使用フラグ

指定された起点からの最短距離経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldStartNodeCode	(IN)起点のノードコード
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

#### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点の道路点を示すノードコードに不備がある場合、この点が計算用道路データに存在しない場合、この点からどの道路点にも到達できない場合には、FALSE を返します。

#### 備考

指定された発地点から、計算用道路データ内のすべての道路点までの最短距離経路を計算します。計算結果は、計算結果取得関数で取得します。

```

BOOL WINAPI ACT_DDIJ_CalcFullRouteDistEx( HANDLE hLibObject,
double ldStartNodeCode,
long lnThreshold, BOOL bNotUseHighWay )

```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldStartNodeCode	起点のノードコード
long lnThreshold	到達時間
BOOL bNotUseHighWay	高速道路非使用フラグ

指定された起点から、指定距離（m単位）以内の最短距離経路を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldStartNodeCode	(IN)起点のノードコード
lnThreshold	(IN)到達距離。m単位。
bNotUseHighWay	(IN)高速道路非使用フラグ。高速道路を使用しない場合には TRUE(1)を、使用する場合には FALSE(0)を設定します。

#### 戻り値

計算が正常に終了した場合は TRUE。正常に終了しなかった場合は FALSE。

発地点の道路点を示すノードコードに不備がある場合、この点が計算用道路データに存在しない場合、この点からどの道路点にも到達できない場合には、FALSE を返します。

#### 備考

指定された発地点から、指定された距離内にあるすべての道路点までの最短時間経路を計算します。計算結果は、計算結果取得関数で取得します。

## 8 . 計算結果取得関数

前節の「計算実行関数」で計算された結果は、次に計算実行関数をコールするか、計算モードを終了する (ACT\_DDIJ\_UnloadRoadNetworks 関数) まで、メモリ内に保持されています。本節の「計算結果取得関数」をコールすることで、この計算結果を取得することができます。

**long WINAPI ACT\_DDIJ\_GetCalcTime( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル  
 double ldNodeCode                      着地点のノードコード

本関数は、指定された点までの所要時間 (分単位) を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード

### 戻り値

値が取得できた場合は発地点から到達地点までの所要時間 (分単位) を返します。値が取得できなかった場合は ACTLIB\_ERROR\_TIME 以上の値を返します。

着地点を示すノードコードが不備である場合、指定された点に到達できない場合、着地点が計算用道路データに存在しない場合は ACTLIB\_ERROR\_TIME 以上の値を返します。

### 備考

本関数をコールする前に、距離計算実行関数が正常に終了していることが必要です。

**long WINAPI ACT\_DDIJ\_GetCalcDSecTime( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル

double ldNodeCode                      着地点のノードコード

本関数は、指定された点までの所要時間（1/10 秒単位）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード

### 戻り値

値が取得できた場合は発地点から到達地点までの所要時間（1/10 秒単位）を返します。値が取得できなかった場合は ACTLIB\_ERROR\_DECTIME を返します。

着地点を示すノードコードが不備である場合、指定された点に到達できない場合、着地点が計算用道路データに存在しない場合は ACTLIB\_ERROR\_DSECTIME を返します。

### 備考

本関数をコールする前に、距離計算実行関数が正常に終了していることが必要です。

**long WINAPI ACT\_DDIJ\_GetCalcDistance( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル

double ldNodeCode                      着地点のノードコード

本関数は、指定された点までの道のり（m単位）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード

### 戻り値

値が取得できた場合は発地点から到達地点までの道のり（m単位）を返します。値が取得できなかった場合は ACTLIB\_ERROR\_DISTANCE を返します。

着地点を示すノードコードが不備である場合、指定された点に到達できない場合、着地点が計算用道路データに存在しない場合は ACTLIB\_ERROR\_DISTANCE を返します。

### 備考

本関数をコールする前に、距離計算実行関数が正常に終了していることが必要です。

```
int WINAPI ACT_DDIJ_GetCalcRouteXY( HANDLE hLibObject, double ldNodeCode
                                   long * lpBufSize, double * lpRouteBuf,
                                   BOOL bContinue )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpBufSize	ルートバッファのサイズ
double * lpRouteBuf,	ルートバッファへのポインタ
BOOL bContinue	継続フラグ

指定された点までのルート情報（XY座標の系列）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
LdNodeCode	(IN)着地点のノードコード
lpBufSize	(IN/OUT)ルートバッファサイズを指定する long 値へのポインタ。関数コール時に lpRouteBuf が格納できる最大のルートの要素数を指定します。関数リターン時には、ルートバッファにセットされたルートの要素数がセットされます。
lpRouteBuf,	(OUT)ルートバッファを示すポインタ
bContinue	(IN)継続フラグ。初回コール時には FALSE。継続コール時には TRUE を指定します。

### 戻り値

ACTLIB\_GETROUTEOK(0) : 正常終了。すべてのルートデータがバッファにセットされた。

ACTLIB\_GETROUTECONT(1) : 正常終了。途中までのルートデータがバッファにセットされた。

ACTLIB\_GETROUTEERROR(2) : エラー。到着地点に到達するルートが存在しない場合。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数を最初にコールする場合 bContinue の値は、FALSE を設定してください。ルートのシーケンスを格納するバッファ（lpRouteBuf）のサイズが実際のルートが辿るノード数より小さい場合は、バッファのサイズまでルートのシーケンスが格納され、戻り値 = 1 となります。この場合、bContinue に TRUE を設定して、本関数を再度コールしてください。

尚、2回目以降に本関数をコールして取得したルートデータの最初の要素は、前回コール時に取得したルートデータの最後の要素に等しくなります。

ルートバッファに格納されるデータは、ルートが辿るノードの X 座標（経度）、Y 座標（緯度）の系列で、それぞれ度単位、小数点以下 6 桁の倍精度浮動小数点数値となります。尚、ルートデータが格納されていないルートバッファの領域の値は不定となります。

尚、本関数で使用している、ルートの「要素」とは、一組のX座標とY座標を示しており、ルートバッファの構造体の要素とは異なります。

すなわち、ルートバッファの実際のサイズは、

ルートデータの要素数 × 8【sizeof(double)】 × 2【経度と緯度】  
となります。

```
int WINAPI ACT_DDIJ_GetCalcRouteNode( HANDLE hLibObject, double ldNodeCode,
                                     long * lpBufSize, double * lpRouteBuf,
                                     BOOL bContinue )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpBufSize	ルートバッファのサイズ
double * lpRouteBuf,	ルートバッファへのポインタ
BOOL bContinue	継続フラグ

指定された点までのルート情報（ノードコードの系列）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード
lpBufSize	(IN/OUT)ルートバッファサイズを指定する long 値へのポインタ。関数コール時に lpRouteBuf が格納できる最大のノード数を指定します。関数リターン時には、ルートバッファにセットされたノード数がセットされます。
lpRouteBuf,	(OUT)ルートバッファを示すポインタ
bContinue	(IN)継続フラグ。初回コール時には FALSE。継続コール時には TRUE を指定します。

### 戻り値

ACTLIB\_GETROUTEOK(0)：正常終了。すべてのルートデータがバッファにセットされた。

ACTLIB\_GETROUTECONT(1)：正常終了。途中までのルートデータがバッファにセットされた。

ACTLIB\_GETROUTEERROR(2)：エラー。到着地点に到達するルートが存在しない場合。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数を最初にコールする場合 bContinue の値は、FALSE を設定してください。ルートのシーケンスを格納するバッファ（lpRouteBuf）のサイズが実際のルートが辿るノード数より小さい場合は、バッファのサイズまでルートのシーケンスが格納され、戻り値 = 1 となります。この場合、bContinue に TRUE を設定して、本関数を再度コールしてください。

尚、2 回目以降に本関数をコールして取得したルートデータの最初の要素は、前回コール時に取得したルートデータの最後の要素に等しくなります。

ルートバッファに格納されるデータは、ルートが辿るノードコードの系列で、14 桁の倍精度浮動小数点数値（小数点以下桁数ゼロ）となります。

尚、ルートデータが格納されていないルートバッファの領域の値は不定となります。

X Y 座標の系列を返す関数の場合と異なり、本関数のルートバッファのサイズは、  
ルートデータの要素数 × 8 【sizeof(double)】  
となります。

```
int WINAPI ACT_DDIJ_GetCalcRouteLink( HANDLE hLibObject, double ldNodeCode,
                                     long * lpBufSize, double * lpRouteBuf,
                                     BOOL bContinue )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpBufSize	ルートバッファのサイズ
double * lpRouteBuf,	ルートバッファへのポインタ
BOOL bContinue	継続フラグ

指定された点までのルート情報（リンクコードの系列）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード
lpBufSize	(IN/OUT)ルートバッファサイズを指定する long 値へのポインタ。関数コール時に lpRouteBuf が格納できる最大のリンク数を指定します。関数リターン時には、ルートバッファにセットされたリンク数がセットされます。
lpRouteBuf,	(OUT)ルートバッファを示すポインタ
bContinue	(IN)継続フラグ。初回コール時には FALSE。継続コール時には TRUE を指定します。

### 戻り値

ACTLIB\_GETROUTEOK(0)：正常終了。すべてのルートデータがバッファにセットされた。

ACTLIB\_GETROUTECONT(1)：正常終了。途中までのルートデータがバッファにセットされた。

ACTLIB\_GETROUTEERROR(2)：エラー。到着地点に到達するルートが存在しない場合。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数を最初にコールする場合 bContinue の値は、FALSE を設定してください。ルートのシーケンスを格納するバッファ（lpRouteBuf）のサイズが実際のルートが辿るノード数より小さい場合は、バッファのサイズまでルートのシーケンスが格納され、戻り値 = 1 となります。この場合、bContinue に TRUE を設定して、本関数を再度コールしてください。

尚、2 回目以降に本関数をコールして取得したルートデータの最初の要素は、前回コール時に取得したルートデータの最後の要素に等しくなります。

ルートバッファに格納されるデータは、ルートが辿るリンクコードの系列で、14 桁の倍精度浮動小数点数値（小数点以下桁数ゼロ）となります。

尚、ルートデータが格納されていないルートバッファの領域の値は不定となります。

X Y 座標の系列を返す関数の場合と異なり、本関数のルートバッファのサイズは、  
ルートデータの要素数 × 8 【sizeof(double)】  
となります。

```
int WINAPI ACT_DDIJ_GetCalcRouteNodeLink( HANDLE hLibObject,
                                          double ldNodeCode,
                                          long * lpBufSize, double * lpRouteBuf,
                                          BOOL bContinue )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpBufSize	ルートバッファのサイズ
double * lpRouteBuf,	ルートバッファへのポインタ
BOOL bContinue	継続フラグ

指定された点までのルート情報（ノードコードとリンクコードの系列）を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード
lpBufSize	(IN/OUT)ルートバッファサイズを指定する long 値へのポインタ。関数コール時に lpRouteBuf が格納できる最大のリンク数を指定します。関数リターン時には、ルートバッファにセットされたノード数とリンク数の合計値がセットされます。
lpRouteBuf,	(OUT)ルートバッファを示すポインタ
bContinue	(IN)継続フラグ。初回コール時には FALSE。継続コール時には TRUE を指定します。

### 戻り値

ACTLIB\_GETROUTEOK(0)：正常終了。すべてのルートデータがバッファにセットされた。

ACTLIB\_GETROUTECONT(1)：正常終了。途中までのルートデータがバッファにセットされた。

ACTLIB\_GETROUTEERROR(2)：エラー。到着地点に到達するルートが存在しない場合。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数を最初にコールする場合 bContinue の値は、FALSE を設定してください。ルートのシーケンスを格納するバッファ（lpRouteBuf）のサイズが実際のルートが辿るノード数より小さい場合は、バッファのサイズまでルートのシーケンスが格納され、戻り値 = 1 となります。この場合、bContinue に TRUE を設定して、本関数を再度コールしてください。

尚、2 回目以降に本関数をコールして取得したルートデータの最初の要素は、前回コール時に取得したルートデータの最後の要素に等しくなります。

ルートバッファに格納されるデータは、ルートが辿るノードコードとリンクコードの系列で、14桁の倍精度浮動小数点数値(小数点以下桁数ゼロ)となります。ルートデータの最初と最後はノードコードとなります(初回コール時、継続コール時とも同様)。

尚、ルートデータが格納されていないルートバッファの領域の値は不定となります。

本関数のルートバッファのサイズは、

( ルートデータのノード数 + ルートデータのリンク数 ) × 8 【sizeof(double)】

但し、ルートデータのノード数 = ルートデータのリンク数 + 1

となります。

```

BOOL WINAPI ACT_DDIJ_GetAttribTimeDist( HANDLE hLibObject, double ldNodeCode,
long * lpTime, long * lpDistance,
UINT unFlag,
int nNumAttribute, int * lpAttributeCode )

```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpTime	所要時間 ( 1/10 秒単位 )
long * lpDistance	道のり ( m 単位 )
UINT unFlag	計算フラグ ( 常に 0 )
int nNumAttribute	指定区間種別コードの数
int * lpAttributeCode	指定区間種別コードバッファ

指定された点までの経路から指定区間種別部分の所要時間と道のりを抜き出します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード
lpTime	(OUT)所要時間 ( 1/10 秒単位 ) を受け取るバッファへのポインタ
lpDistance	(OUT)道のり ( m 単位 ) を受け取るバッファへのポインタ
unFlag	(IN)計算フラグ ( 常に 0 )
nNumAttribute	(IN)lpAttributeCode に格納されている指定区間種別コードの数
lpAttributeCode	(IN)指定区間種別コードを格納するバッファへのポインタ

### 戻り値

関数が正常に終了した場合は TRUE。失敗した場合は FALSE。

計算が終了していない場合、着地点までのルートが存在しない場合には、関数は FALSE を返します。

### 備考

本関数は、lpAttributeCode バッファに格納されている区間種別コードに合致する区間種別部分の所要時間と道のりを抜き出し、lpTime と lpDistance が示すバッファに値を設定します。

本関数の実行前に、距離計算関数が正常に終了している必要があります。

**BOOL WINAPI ACT\_ADMN\_IsShapeFilesExist( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

詳細ルート表示が可能かどうかチェックします。

**パラメータ****説明**

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル

**戻り値**

詳細ルート表示が可能な場合は TRUE。不可能な場合は FALSE。

**備考**

本関数は、ACT\_ADMN\_SetWorkDir 関数で指定されている計算用道路データフォルダ内の計算用道路データに詳細ルートデータが存在するかどうかをチェックします。

詳細ルートデータを使用する関数 ( ACT\_DDIJ\_GetCalcRouteDetail 関数 ) をコールする前に、本関数を用いて詳細ルートデータの有無をチェックしてください。

```
int WINAPI ACT_DDIJ_GetCalcRouteDetail ( HANDLE hLibObject, double ldNodeCode,
                                         long * lpBufSize, double * lpRouteBuf,
                                         BOOL bContinue )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldNodeCode	着地点のノードコード
long * lpBufSize	ルートバッファのサイズ
double * lpRouteBuf,	ルートバッファへのポインタ
BOOL bContinue	継続フラグ

指定された点までの詳細ルート情報 (XY 座標の系列) を求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
LdNodeCode	(IN)着地点のノードコード
lpBufSize	(IN/OUT)ルートバッファサイズを指定する long 値へのポインタ。関数コール時に lpRouteBuf が格納できる最大のルートの要素数を指定します。関数リターン時には、ルートバッファにセットされたルートの要素数がセットされます。
lpRouteBuf,	(OUT)ルートバッファを示すポインタ
bContinue	(IN)継続フラグ。初回コール時には FALSE。継続コール時には TRUE を指定します。

### 戻り値

ACTLIB\_GETROUTEOK(0) : 正常終了。すべてのルートデータがバッファにセットされた。

ACTLIB\_GETROUTECONT(1) : 正常終了。途中までのルートデータがバッファにセットされた。

ACTLIB\_GETROUTEERROR(2) : エラー。到着地点に到達するルートが存在しない場合。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数を最初にコールする場合 bContinue の値は、FALSE を設定してください。ルートのシーケンスを格納するバッファ (lpRouteBuf) のサイズが実際のルートが辿るノード数より小さい場合は、バッファのサイズまでルートのシーケンスが格納され、戻り値 = 1 となります。この場合、bContinue に TRUE を設定して、本関数を再度コールしてください。

尚、2 回目以降に本関数をコールして取得したルートデータの最初の要素は、前回コール時に取得したルートデータの最後の要素に等しくなります。

ルートバッファに格納されるデータは、ルートが辿るノードの X 座標 (経度)、Y 座標 (緯度) の系列で、それぞれ度単位、小数点以下 6 桁の倍精度浮動小数点数値となります。尚、ルートデータが格納されていないルートバッファの領域の値は不定となります。

**void WINAPI ACT\_DDIJ\_GetRouteNameFileName( HANDLE hLibObject,  
LPSTR szFileName )**

HANDLE hLibObject 距離計算コアオブジェクトのハンドル  
LPSTR szFileName 経由交差点名称ファイルへのフルパス

経由交差点名称出力ファイルのフルパスを求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
szFileName	(OUT)経由交差点名称ファイルへのフルパスを格納する文字列へのポインタ。バッファのサイズは MAX_PATH 以上必要です。

#### 戻り値

なし。

#### 備考

本関数は ACT\_DDIJ\_OutputCalcRouteName 関数が経由交差点名称を出力するファイルへのフルパスを取得します。経由交差点名称ファイルは、距離計算コアオブジェクト固有のファイルで、距離計算コアオブジェクト作成時( ACT\_ADMN\_CreateLibObject 関数コール時)に Windows のテンポラリ・フォルダ下に「ACTxx.TMP」( xx は 16 進数)という名称で作成されます。また、距離計算コアオブジェクトが廃棄された時( ACT\_ADMN\_DeleteLibObject 関数コール時)に削除されません。

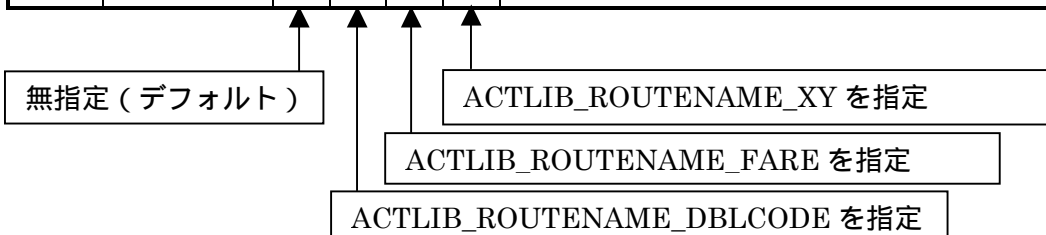


経路交差点名称出力ファイルは、カンマ区切りテキストファイルで、1行目はカラム名称となっています。2行目以降がデータ行で、データの最初の行は発地点ノードデータ、最終行は終点ノードデータです。連続して経路交差点名称を出力した場合は、前回の終点ノードが中継点ノードとなります。

出力カラムは unFlag に指定する値によって異なり、それぞれのフラグが指定された場合、下表の「出力」欄に 印が付されたカラムが出力され、×印のカラムは出力されません。

時間、距離についてはレコードがノードの場合のみ有効です。通行料金、インターチェンジタイプはレコードがインターチェンジの場合のみ有効です。最終レコード（終点ノード）の時間、距離、通行料金はルート全体の合計値となっています。

項番	カラム名称	出力				内容
1	SE					地点識別フラグ 1=計算の発地点、中継点、または、終点 0=上記以外
2	LINK					ノード/リンク識別フラグ 0=ノード、1=リンク、インターチェンジ
3	CODE					ノードまたはリンクの14桁コード ACTLIB_ROUTENAME_DBLCODE を指定した場合に出力。
4	PREF		×			都道府県コード。ノードまたはリンクコードの上2桁。 ACTLIB_ROUTENAME_DBLCODE を指定しない場合に出力。
5	CODE		×			道路点IDまたは道路区間ID。ノードまたはリンクコードの下9桁。 ACTLIB_ROUTENAME_DBLCODE を指定しない場合に出力。
6	NAME					交差点名称、道路名称、インターチェンジ名称。最大20バイト
7	TIME					ノードまでの所要時間。単位：1/10秒
8	DIST					ノードまでの距離。単位：m
9	FareSS					通行料金（軽・二輪）。単位：円
10	FareS					通行料金（普通車）。単位：円
11	FareM					通行料金（中型車）。単位：円
12	FareL					通行料金（大型車）。単位：円
13	FareLL					通行料金（特大車）。単位：円
14	ICType					インターチェンジタイプ 0=インターチェンジではない 1~7=インターチェンジ（詳細は定数一覧を参照）
15	X					経度（X座標）。単位：度
16	Y					緯度（Y座標）。単位：度



下記は、ノード 18000000021574 からノード 19000000016496 までの経由交差点を出力した例です。  
unFlag には、全てのフラグ( ACTLIB\_ROUTENAME\_DBLCODE | ACTLIB\_ROUTENAME\_NODE  
| ACTLIB\_ROUTENAME\_LINK | ACTLIB\_ROUTENAME\_FARE | ACTLIB\_ROUTENAME\_XY )  
を指定しています。



## 9 . リンク速度管理関数

計算に使用する道路（リンク）の速度は、計算モード終了までメモリ上に保持されている「一時的速度」を、ディスク上に保存されている「恒久的速度」の2種類の速度があり、距離計算コアライブラリは、メモリ上に保持されている「一時的速度」を用いて演算を行います。

本節のリンク速度管理関数は、この「一時的速度」「恒久的速度」のどちらに対しても取得/変更操作が行えるようになっています。

尚、計算モードを開始した直後の状態では「一時的速度」 = 「恒久的速度」の関係が成立していません。

ACT 距離計算コアでは、各リンクについて「順方向」と「逆方向」2方向の速度を持っています。道路区間データの「接続点ID1」から「接続点ID2」への方向が「順方向」で、その逆が「逆方向」となります。

尚、ACT\_ADMN\_AttachRoadNetwork 関数を用いて同じメモリ領域にロードされている計算用道路データを参照している複数の距離計算オブジェクトがあるとき、一方で道路速度を変更すると同時に、他方で計算を実行した場合には、計算結果は不定となります。

```

BOOL WINAPI ACT_ADMN_ChangeBothSpeedEx( HANDLE hLibObject,
double ldLinkCode,
int nSpeedAB, int nSpeedBA, UINT unFlag )

```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldLinkCode	対象道路のリンクコード
int nSpeedAB	順方向の速度
int nSpeedBA	逆方向の速度
UINT unFlag	変更フラグ

指定されたリンクの速度（両方向）を設定します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldLinkCode	(IN)対象道路のリンクコード
nSpeedAB	(IN)順方向の速度。1/10km/h 単位で指定します。e.g.100=10km/h
nSpeedBA	(IN)逆方向の速度。1/10km/h 単位で指定します。e.g.100=10km/h
unFlag	(IN)変更フラグ。速度変更方法および変更対象を指定します。下記の2つの値のいずれかまたは両方を指定することができます。 ACTLIB_SPEED_TEMP(0x0010) : 一時的速度を変更する。 ACTLIB_SPEED_ZERORESERVE(0x0040) : 速度ゼロの方向の速度は変更しない。

### 戻り値

速度変更が正常に行われた場合は TRUE。失敗した場合は FALSE。

### 備考

nSpeedAB、nSpeedBA に負値を指定した場合には、当該方向の速度は変更されません。たとえば、nSpeedAB=300、nSpeedBA = - 1 と指定すると、この道路区間の順方向の速度は 30km/h に変更されますが、逆方向の速度は変更されません。

ACTLIB\_SPEED\_ZERORESERVE を指定すると、変更対象の道路区間で現在速度がゼロに指定されている方向の速度は変更しません。これによって、一方通行が指定されている道路の速度の変更を変更する際に誤って一方通行情報を解除してしまうのを防ぐことができます。

一時的速度の変更は、ACT\_DDIJ\_LoadRoadNetworks 関数がコールされて計算用道路データの一部がメモリにロードされている状態でのみ可能です。従って、ACTLIB\_SPEED\_TEMP フラグを指定する場合には、計算用道路データの一部がメモリにロードされている状態ではありません。

ACTLIB\_SPEED\_TEMP フラグを指定しない場合には、一時的速度と恒久的速度の両方の速度が変更されます。

```
int WINAPI ACT_ADMN_GetSpeed( HANDLE hLibObject,
                             double ldFromNodeCode, double ldToNodeCode,
                             UINT unFlag )
```

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldFromNodeCode	起点の道路点のノードコード
double ldToNodeCode	終点の道路点のノードコード
UINT unFlag	取得対象フラグ

指定されたリンクの速度（片方向）を取得します。

パラメータ	説明
LibObject	(IN)距離計算コアオブジェクトのハンドル
ldFromNodeCode	(IN)起点の道路点のノードコード
ldToNodeCode	(IN)終点の道路点のノードコード
unFlag	(IN)取得対象フラグ。一時的速度または恒久的速度を取得することを指定します。次の値のいずれかまたは両方を指定することができます。 ACTLIB_SPEED_TEMP(0x0010)：一時的速度を取得します。 ACTLIB_SPEED_PARAM(0x0020)：恒久的速度を取得します。

### 戻り値

関数が正常に終了した場合は取得した速度（1/10km/h 単位）。失敗した場合は - 1。

### 備考

ldFromNodeCode で指定された起点と、ldToNodeCode で指定された終点を結ぶ道路区間の、起点から終点方向の速度を取得します。起点と終点が道路区間の両端にない場合は関数は失敗します。一時的速度を取得するには、ACT\_DDIJ\_LoadRoadNetworks 関数がコールされて計算用道路データの一部がメモリにロードされている状態でのみ可能です。従って、ACTLIB\_SPEED\_TEMP フラグのみを指定する場合には、計算用道路データの一部がメモリにロードされている状態でなくてはなりません。一方、計算用道路データの一部がメモリにロードされている状態でも、ACTLIB\_SPEED\_PARAM フラグを指定することによって、恒久的速度を取得することができます。

ACTLIB\_SPEED\_TEMP と ACTLIB\_SPEED\_PARAM の両方をフラグに指定した場合は、一時的速度の取得が優先されます。一時的速度が取得できない場合にのみ、恒久的速度が取得されます。

**DWORD WINAPI ACT\_ADMN\_GetSpeedEx( HANDLE hLibObject,  
double ldLinkCode, UINT unFlag )**

HANDLE hLibObject 距離計算コアオブジェクトのハンドル  
double ldLinkCode 対象リンクコード  
UINT unFlag 取得対象フラグ

指定されたリンクの速度（両方向）を取得します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldLinkCode	(IN)対象リンクコード
unFlag	(IN)取得対象フラグ。一時的速度または恒久的速度を取得することを指定します。次の値のいずれかまたは両方を指定することができます。 ACTLIB_SPEED_TEMP(0x0010)：一時的速度を取得します。 ACTLIB_SPEED_PARAM(0x0020)：恒久的速度を取得します。

#### 戻り値

関数が正常に終了した場合は両方向の速度。戻り値の LOWORD が順方向、HIWORD が逆方向。  
失敗した場合は - 1。

#### 備考

一時的速度を取得するには、ACT\_DDIJ\_LoadRoadNetworks 関数がコールされて計算用道路データの一部がメモリにロードされている状態でのみ可能です。従って、ACTLIB\_SPEED\_TEMP フラグのみを指定する場合には、計算用道路データの一部がメモリにロードされている状態ではなくてはなりません。一方、計算用道路データの一部がメモリにロードされている状態でも、ACTLIB\_SPEED\_PARAM フラグを指定することによって、恒久的速度を取得することができます。

ACTLIB\_SPEED\_TEMP と ACTLIB\_SPEED\_PARAM の両方をフラグに指定した場合は、一時的速度の取得が優先されます。一時的速度が取得できない場合にのみ、恒久的速度が取得されます。

**long WINAPI ACT\_ADMN\_CountTempSpeed( HANDLE hLibObject,  
LPSTR lpszTempSpeedFile )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル  
LPSTR lpszTempSpeedFile              一時的速度の一覧を保存するファイルへのフルパスを保存する文字列へのポインタ

一時的速度が設定されているリンク数を取得します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
lpszTempSpeedFile	(IN)一時的速度の一覧を保存するファイルへのフルパスを保存する文字列へのポインタ。ファイルを出力しない場合は NULL。

### 戻り値

一時的速度が設定されている道路区間の本数。関数が失敗した場合は - 1。

### 備考

本関数は一時的速度が設定されている道路区間の数を取得します。

また、lpszTempSpeedFile が NULL 以外の場合は、指定されたファイルに下記の形式で一時的速度が設定されているリンクの速度情報が保存されます。尚、SPEEDA は順方向を、SPEEDB は逆方向を示し、速度値は 1/10km/h 単位です。速度値が -1 の場合は当該方向は一時的速度が設定されていないことを示します。

```
“LINK”, “SPEEDA”, “SPEEDB”
13000000012345,300,300
13000000012346,300,-1
13000000012399,300,-1
...
```

一時的速度は、ACT\_ADMN\_LoadRoadNetworks 関数がコールされて計算用道路データの一部がメモリにロードされている状態でのみ有効な速度です。従って、計算用道路データの一部がメモリにロードされていない状態で本関数を実行すると、関数はエラー ( - 1 ) を返します。

**BOOL WINAPI ACT\_ADMN\_CommitTempSpeed( HANDLE hLibObject )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル

一時的速度を恒久的速度として保存します。

**パラメータ****説明**

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル

**戻り値**

関数が正常に終了した場合は TRUE。関数が失敗した場合は FALSE。

**備考**

本関数は一時的速度を恒久的速度としてディスクに保存します。

一時的速度は、ACT\_ADMN\_LoadRoadNetworks 関数がコールされて計算用道路データの一部がメモリにロードされている状態でのみ有効な速度です。従って、計算用道路データの一部がメモリにロードされていない状態で本関数を実行すると、関数は FALSE を返します。

尚、複数の距離計算オブジェクトが同じフォルダに保存されている計算用道路データをロードしている状態で、複数の距離計算オブジェクトが同時に本関数をコールすると、エラーが発生します。

**BOOL WINAPI ACT\_ADMN\_ClearTempSpeed( HANDLE hLibObject )**

HANDLE hLibObject                      距離計算コアオブジェクトのハンドル

一時的速度変更を取り消し元の速度（恒久的速度）に戻します。

**パラメータ**

**説明**

---

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

**戻り値**

関数が正常に終了した場合は TRUE。関数が失敗した場合は FALSE。

**備考**

本関数は一時的速度をすべて取り消し、恒久的速度に変更します。

一時的速度は、ACT\_ADMN\_LoadRoadNetworks 関数がコールされて計算用道路データの一部分がメモリにロードされている状態でのみ有効な速度です。従って、計算用道路データの一部分がメモリにロードされていない状態で本関数を実行すると、関数は FALSE を返します。

## 10 . 通行料金計算関数

本節では、高速道路と有料道路の通行料金を計算する関数を説明します。

通行料金を計算するには、計算用道路データ作成時にデジタル道路地図データの種別とバージョンに合致した通行料金データが計算用道路データに設定されている必要があります。

【Ver 5.2 新規関数】

**BOOL WINAPI ACT\_ADMN\_IsFareExist( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

通行料金計算が可能かどうかチェックします。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル

### 戻り値

通行料金計算が可能な場合には TRUE。それ以外の場合は FALSE。

### 備考

本関数は hLibObject に指定されている計算用道路データに通行料金データが設定されているかどうかをチェックします。計算用道路データの保存場所が指定されていない場合、通行料金データが設定されていない場合には、本関数は FALSE を返します。

```

BOOL WINAPI ACT_DDIJ_CalcTollRoadFare( HANDLE hLibObject
                                double ldToNodeCode,
                                LPTOLLROADFARE lpTollRoadFare );

```

HANDLE hLibObject      距離計算コアオブジェクトのハンドル  
 double ldNodeCode      着地点のノードコード  
 LPTOLLROADFARE lpTollRoadFare      通行料金構造体へのポインタ

指定されたノードまでの通行料金を計算します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)着地点のノードコード
lpTollRoadFare	(OUT)計算結果を格納する通行料金構造体へのポインタ。通行料金構造体の定義は下記のとおりです。 <pre> typedef struct tagTOLLROADFARE{     long    InFareSS;        // 軽自動車料金（単位：円）     long    InFareS;        // 普通車料金（単位：円）     long    InFareM;        // 中型車料金（単位：円）     long    InFareL;        // 大型車料金（単位：円）     long    InFareLL;       // 特大車料金（単位：円）     long    InFareReg;      // 予備     double  IdDistance;    // 予備 } TOLLROADFARE; typedef TOLLROADFARE *    LPTOLLROADFARE; </pre>

### 戻り値

通行料金の計算が正常に行われた場合には TRUE。それ以外の場合は FALSE。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

着地点までの距離計算関数が実行されていない場合、着地点までのルートが存在しない場合、通行料金データが設定されていない場合には、本関数は FALSE を返します。

```
void WINAPI ACT_DDIJ_GetTollRoadFareFileName( HANDLE hLibObject,
                                             LPSTR szFileName )
```

HANDLE hLibObject     距離計算コアオブジェクトのハンドル  
 LPSTR    szFileName    経由有料道路明細ファイルへのフルパス

経由有料道路明細ファイルのフルパスを求めます。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
szFileName	(OUT) 経由有料道路明細ファイルへのフルパスを格納する文字列へのポインタ。バッファのサイズは MAX_PATH 以上必要です。

#### 戻り値

なし。

#### 備考

本関数は ACT\_DDIJ\_CalcTollRoadFareDetail 関数が経由する有料道路のインターチェンジ名を出力するデフォルトファイルへのフルパスを取得します。経由有料道路明細ファイルは、距離計算コアオブジェクト固有のファイルで、距離計算コアオブジェクト作成時( ACT\_ADMN\_CreateLibObject 関数コール時)に Windows のテンポラリ・フォルダ下に「ACTxx.TMP」(xx は 16 進数)という名称で作成されます。また、距離計算コアオブジェクトが廃棄された時( ACT\_ADMN\_DeleteLibObject 関数コール時)に削除されます。

```

BOOL WINAPI ACT_DDIJ_CalcTollRoadFareDetail( HANDLE hLibObject,
double ldToNodeCode,
LPSTR lpszOutputFile, UINT unOutputFlag );
    
```

HANDLE hLibObject      距離計算コアオブジェクトのハンドル  
 double ldToNode          着地点のノードコード  
 LPSTR lpszOutputFile    出力先ファイルのフルパス  
 UINT unOutputFlag      出力指定フラグ（未使用、常に 0 を指定）

指定された点までの経由有料道路明細（インターチェンジ名称）を出力します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldToNode	(IN)着地点のノードコード
lpszOutputFile	(IN)出力先ファイルのフルパスを格納する文字列へのポインタ。指定しない場合は NULL。出力先ファイルを指定しない場合の出力先はデフォルトの経由有料道路明細ファイル（ACT_DDIJ_GetTollRoadFareFileName 関数で取得することができます）となります。
unOutputFlag	(IN)出力指定フラグ。現在は使用されていません。常にゼロ（0）を設定してください。

### 戻り値

経由有料道路明細を出力できた場合には TRUE。出力できなかった場合には FALSE。

### 備考

本関数の実行前に、距離計算関数が正常に終了している必要があります。

本関数は、指定されたファイルに指定ノード点までの経由インターチェンジの一覧を CSV 形式で出力します。

本関数はファイルに追加書き込みを行いますので新しい経由有料道路明細を出力する直前にファイルをトランケート（長さをゼロにする）してください。尚、ファイルを削除してしまうと、同じ一時ファイル名称が別の距離計算コアオブジェクトで使用される可能性がありますので、注意してください。

出力されるファイルはカンマ区切りの CSV ファイルで、最初の行はカラム名称となっています。2 行目以降がデータ行で、最終行には合計金額値が出力されます。

出力されるカラムは下表のとおりです。

項番	カラム名	内容
1	入道路名	入口インターチェンジ、または課金インターチェンジの道路名称。 最大30バイト。
2	入口IC	入口インターチェンジ、または課金インターチェンジ名称。 最大30バイト。
3	出道路名	出口インターチェンジの道路名称。最大30バイト。
4	出口IC	出口インターチェンジ名称。最大30バイト。
5	二輪・軽	二輪・軽の通行料金。単位：円
6	普通車	普通車の通行料金。単位：円
7	中型車	中型車の通行料金。単位：円
8	大型車	大型車の通行料金。単位：円
9	特大車	特大車の通行料金。単位：円
10	距離	インターチェンジ間の距離。単位：km、小数点以下1桁 本項目の値はデータの整備状況により不正確な場合があります。

下記は、ノード 18000000021574 からノード 19000000016496 までの経由有料道路明細を出力した例です。

"入道路名","入口IC","出道路名","出口IC","二輪・軽","普通車","中型車","大型車","特大車","距離"

"東海北陸自動車道","白鳥","東海北陸自動車道","飛驒清見",1000,1200,1450,1900,3050,40.9

"安房峠道路","平湯","","",600,750,900,1250,2050,0.0

"長野自動車道","松本","中央道・富士吉田線","上野原",3100,3850,4600,6250,10350,157.6

","", "", "", "", 4700,5800,6950,9400,15450,198.5

入口と出口

課金料金所

合計行

## 1 1 . その他の関数

本節では、ノードの座標を取得する関数と、指定された X Y 座標に最も近い道路点を検索する関数を説明します。

ノードの座標を取得する関数は、指定されたノードが計算用道路データに存在するかどうかを判断する用途にも利用できます。

**double WINAPI ACT\_DDIJ\_GetNodeX( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

double ldNodeCode      道路点のノードコード

指定されたノードの X 座標を取得します。

パラメータ	説明
hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldNodeCode	(IN)道路点のノードコード

### 戻り値

関数が正常に終了した場合は、X 座標（度単位）。失敗した場合は - 1。

### 備考

指定された道路点の X 座標を取得します。

**double WINAPI ACT\_DDIJ\_GetNodeY( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

double ldNodeCode      道路点のノードコード

指定されたノードのY座標を取得します。

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

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

ldNodeCode	(IN)道路点のノードコード
------------	----------------

### 戻り値

関数が正常に終了した場合は、Y座標（度単位）。失敗した場合は - 1。

### 備考

指定された道路点のY座標を取得します。

**BOOL WINAPI ACT\_DDIJ\_IsHighwayNode( HANDLE hLibObject, double ldNodeCode )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

double ldNodeCode      道路点のノードコード

指定された道路点が高速道路上の点であるかどうかを判定します。

#### パラメータ

#### 説明

---

hLibObject              (IN)距離計算コアオブジェクトのハンドル

ldNodeCode              (IN)道路点のノードコード

#### 戻り値

指定道路点が高速道路上の点である場合は TRUE。それ以外の場合は FALSE。

#### 備考

指定された道路点が計算用道路データに存在しない場合は FALSE を返します。

高速道路上の点とは、当該道路点に接続するすべての道路区間が高速道路（通常は区間種別="01"および"02"）である点です。

**double WINAPI ACT\_DDIJ\_GetNearestNodeEx( HANDLE hLibObject,  
double ldX, double ldY,  
double ldRange, int nFlag )**

HANDLE hLibObject	距離計算コアオブジェクトのハンドル
double ldX	検索する地点のX座標(経度)
double ldY	検索する地点のY座標(緯度)
double ldRange	検索範囲
int nFlag	高速道路上道路点除外フラグ

指定されたX Y座標に最も近い道路点を検索します。

### パラメータ

### 説明

hLibObject	(IN)距離計算コアオブジェクトのハンドル
ldX	(IN)検索する地点のX座標(度単位)
ldY	(IN)検索する地点のY座標(度単位)
ldRange	(IN)検索範囲(度単位)
nFlag	(IN)高速道路上道路点除外フラグ。 0 = 全ての点を検索対象とする。 1 = 高速道路上の道路点は検索しない。

### 戻り値

指定座標に最も近い道路点のノードコード。道路点が見つからない場合は - 1。

### 備考

本関数は指定されたX Y座標の±ldRange度の範囲にある道路点を検索します。  
nFlag=1のときは高速道路上の点を検索対象から除外します。

**long WINAPI ACT\_DDIJ\_GetNodeNumber( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

距離計算オブジェクトにロードされている計算用道路データのノード数を求めます。

---

**パラメータ**

**説明**

hLibObject                      (IN)距離計算コアオブジェクトのハンドル

**戻り値**

距離計算オブジェクトにロードされている計算用道路データのノード数。

エラーが発生した場合は - 1。

**備考**

本関数は、計算用道路データがロードされていない場合は - 1 を返します。

**long WINAPI ACT\_DDIJ\_GetLinkNumber( HANDLE hLibObject )**

HANDLE hLibObject      距離計算コアオブジェクトのハンドル

距離計算オブジェクトにロードされている計算用道路データのリンク数を求めます。

#### パラメータ

#### 説明

---

hLibObject	(IN)距離計算コアオブジェクトのハンドル
------------	-----------------------

#### 戻り値

距離計算オブジェクトにロードされている計算用道路データのリンク数。

エラーが発生した場合は - 1。

#### 備考

本関数は、計算用道路データがロードされていない場合は - 1 を返します。

## 1 2 . 定数一覧

所要時間、道のりのエラー値は下表の値となっています。

項番	定数	値 ( 10 進 )	意味
1	ACTLIB_ERROR_TIME	1789569	分単位でのエラー値
2	ACTLIB_ERROR_DSECTIME	1073741824	1 / 10 秒単位でのエラー値
3	ACTLIB_ERROR_DISTANCE	1073741824	m単位でのエラー値

ルート情報取得関数の戻り値は下表の値となっています。

項番	定数	値 ( 10 進 )	意味
1	ACTLIB_GETROUTEOK	0	すべてのルートデータを出力
2	ACTLIB_GETROUTECONT	1	継続ルートデータあり
3	ACTLIB_GETROUTEERROR	2	ルートデータ出力失敗

メモリ・オプション指定フラグには下表のいずれかの値を使用します。

項番	定数	値 ( 16 進 )	意味
1	( なし )	0x0000	通常のメモリ消費
2	ACTLIB_MODE_HIMEMORY	0x0080	通常の約 1.5 倍のメモリ消費
3	ACTLIB_MODE_FULLMEMORY	0x0800	通常の約 1.8 倍のメモリ消費

経由交差点名称出力関数のフラグには下記の値を使用します。

項番	定数	値( 1 6 進 )	意味
1	ACTLIB_ROUTENAME_DBLCODE	0x0400	ノードコードとリンクコードに 14 桁コードを出力します。本フラグを指定しない場合は「都道府県コード、道路点 I D」の形式で出力されます。
2	ACTLIB_ROUTENAME_NODE	0x0001	交差点名称を出力します。
3	ACTLIB_ROUTENAME_LINK	0x0002	道路名称を出力します。
4	ACTLIB_ROUTENAME_FARE	0x0010	インターチェンジ名称と通行料金を出力します。
5	ACTLIB_ROUTENAME_XY	0x0008	交差点とインターチェンジの X Y 座標を出力します。

リンク速度設定・取得関数のフラグには下記の値を使用します。

項番	定数	値 (16 進)	意味
1	ACTLIB_SPEED_TEMP	0x0010	一時的速度 (メモリ上) 指定
2	ACTLIB_SPEED_PARAM	0x0020	恒久的速度 (ディスク上) 指定
3	ACTLIB_SPEED_ZERORESERVE	0x0040	ゼロ方向速度変更禁止

インターチェンジ種別は下表のように定義されています。

項番	定数	値 (10 進)	意味
1	ACT_TOLL_NONE	0	不明
2	ACT_TOLL_GETCARD	1	通行券を受け取る入口
3	ACT_TOLL_PAYOUT	2	通行券を渡して通行料金を支払う出口
4	ACT_TOLL_PAYGATE	3	通過時に通行料金を支払うゲート
5	ACT_TOLL_FREECARD	5	乗継券を受け取るゲート
6	ACT_TOLL_FREEPAY	6	乗継券があれば無料、なければ通行料金を支払うゲート
7	ACT_TOLL_INOUT	7	通行券を受け取る入口と通行券を渡して通行料金を支払う出口を兼用している出入口

インターチェンジ種別名称は下表のように定義されています。

項番	定数	値 (文字列)
1	ACT_TOLL_NONE_TEXT	(NULL)
2	ACT_TOLL_GETCARD_TEXT	入口
3	ACT_TOLL_PAYOUT_TEXT	出口
4	ACT_TOLL_PAYGATE_TEXT	課金
5	ACT_TOLL_FREECARD_TEXT	乗継所
6	ACT_TOLL_FREEPAY_TEXT	乗継料金所
7	ACT_TOLL_INOUT_TEXT	出入口

### 1 3 . エラーログ

距離計算コアライブラリの関数内部でエラーが発生した場合には、エラー状況がエラーログに書き出されます。

エラーログファイルは、距離計算コアライブラリがインストールされているフォルダ下に ACTLIB50.LOG という名称で保存されています。尚、本ファイルは、距離計算オブジェクトが生成されたときに、ファイルの直近書き込み日付が前日以前である場合に削除されます。

ログファイルの1行の書式は下表のとおりです。

項番	書式	内容
1	XXXXXX	エラーが発生した距離計算オブジェクトのハンドル(16進数)
2	YYYYMMDD	エラーが発生した日付
3	HH:MM:SS	エラーが発生した時刻
4	文字列	エラーが発生した関数等の表記
5	文字列	エラーの具体的内容

以上