ACTIAN™

Distributed Tuning Interface Guide

Zen v16

Activate Your Data™

目次

このドキュメントについて

このドキュメントには、ソフトウェア開発キット（SDK）の一部として提供される Zen Distributed Tuning Interface コンポーネントに関する情報が含まれています。

このドキュメントの読者

このドキュメントは、Zen に精通し、Distributed Tuning Interface を使用して管理レベルのアプリケーションを開発したいユーザー向けにデザインされています。

本ソフトウェアの全般的な使用手順やほかのデータベース アクセス方法については説明していません。Ditributed Tuning Interface の使用法のリファレンスを提供します。

Distributed Tuning Interface Guide

以下のトピックでは、Zen Distributed Tuning Interface について紹介します。

- Distributed Tuning Interface の概要
- DTI の基本的な使用法
- DTI のサンプル プログラム
- DTI を使用する一般的なタスク

Distributed Tuning Interface のリファレンスへ直接移動して、Zen で DTI を使用する方法の詳細を参照することもできます。

Distributed Tuning Interface の概要

Distributed Tuning Interface（DTI）の目的は、Zen コンポーネントの設定、監視、および診断のためのアプリケーション プログラミング インターフェイスを提供することにあります。

メモ： 簡潔にするため、本ドキュメントのこれ以降、全体を通して Distributed Tuning Interface は DTI と記載します。

文字列引数のエンコード

ユーザー アプリケーションは、API レベルではクライアントの OS エンコードを使用します。DTI は、サーバーおよびクライアント上の OS エンコード間の相違について内部的に対処します。

古いバージョンのクライアントがサーバーと通信している場合、データベース エンジンでは、そのクライアントで使用しているエンコードとサーバーで使用可能なエンコードに互換性があるものと想定しています。

API カテゴリ

利用可能な API のカテゴリは DTI 関数グループに要約されています。

実行権限

通常、DTI アプリケーションでしたいことと言えば、DTI 関数のいずれかを呼び出すことと、すべての設定を表示したり変更したりすることでしょう。このフル アクセスを確保するには、サーバー マシン上で管理者レベルの権限を持つユーザーの名前とパスワードを提供することにより、サーバーに接続します。これは、DTI アプリケーションがターミナル サービス セッションを介してローカルで実行されている場合、あるいはリモートで実行されている場合に適用されます。ローカルで実行されているアプリケーションでは、ユーザー名とパスワードを省略できます。省略しても、任意の DTI 関数の呼び出しや、すべての設定の表示および変更は行えます。DTI を使ってサーバーへの接続を行うを参照してください。

管理者レベルの権限がない場合、ターミナル サーバー セッションを介してローカルで実行されているアプリケーションやリモートで実行されているアプリケーションは、ほとんどの DTI 関数に対してアクセス エラーを返します。関数のサブセットのみが動作します。たとえば、フル アクセスが許可されている場合に設定を変更できる関数の多くは、読み取り専用のアクセスに制限されます。

DTI の基本的な使用法

ヘッダー ファイル

DTI 関数は次のヘッダー ファイルに定義されています。

- btitypes.h
- catalog.h
- config.h
- connect.h
- ddf.h
- dticonst.h
- dtilicense.h
- dtisecurity.h
- monitor.h

リンク ライブラリ

次の表は、DTI のリンク ライブラリと、そのライブラリが Windows および Linux で最初に利用可能になったリリース バージョンの一覧です。表で定義されているように、アプリケーションを適切なライブラリにリンクします。

ライブラリ [1]	Windows	Linux	ライブラリを最初に利用可能なバージョン
w3dbav90.lib[2]	32 ビット		PSQL v9.0
w64dba.lib	64 ビット		PSQL v10.0
w3dbav80.lib[2]	32 ビット		Pervasive.SQL V8.0
w3dbav78.lib[2]	32 ビット		Pervasive.SQL 2000i (SP3)
w3dbav75.lib[2]	32 ビット		Pervasive.SQL 2000
libpsqldti.so		32 ビット	Pervasive.SQL V8.6
libpsqldti.so		64 ビット	PSQL 10.10

[1] ライブラリはすべて Microsoft Visual Studio 2019 でコンパイルされています。

[2] 各 32 ビット ライブラリは、それ以前のライブラリの上位集合です。たとえば、w3dbav90.lib は w3dbav75.lib、w3dbav78.lib、および w3dbav80.lib の上位集合です。

DTI の関数については、Distributed Tuning Interface のリファレンスで説明します。

関数を呼び出す前に

DTI を呼び出す場合は、まず PvStart() 関数を呼び出す必要があります。その後、セッションを終了するまでに複数の DTI 関数を呼び出すことができます。

セッションを終了するときは、PvStop() 関数を呼び出してセッションを閉じる必要があります。

DTI のサンプル プログラム

DTI アクセス方法のためのランタイム ファイルは、デフォルトで Zen データベース エンジンおよび Zen クライアントと一緒にインストールされます。DTI アプリケーションを作成するためには、少なくとも Zen クライアントが必要です。

ヘッダー ファイルとサンプル ファイルは AG-TECH PSQL Library からダウンロードできます。特定の開発環境に関するサンプル ファイルは、次の表に示すように別々のディレクトリにインストールされます。

開発環境	場所
MS Visual C++ 8	＜インストール ディレクトリ＞ ¥SAMPLES¥ MSVC2005
MS Visual C++ 7	＜インストール ディレクトリ＞ ¥SAMPLES¥ MSVC2003
MS Visual C++ 6	＜インストール ディレクトリ＞ ¥SAMPLES¥MSVC
Delphi 5	＜インストール ディレクトリ＞ ¥SAMPLES¥ DELPHI5

詳細については、Zen データベース エンジンと一緒にインストールされる DTI 用のリリース ノート（readme_dti.htm）を参照してください。

DTI を使用する一般的なタスク

このトピックでは、DTI でよく使用される主要なタスクを説明します。

DTI を使ってサーバーへの接続を行う

サーバーへの接続ハンドルを取得する手順を説明します。多くの DTI 関数にとって、これは最初の手順です。

サーバーへの接続ハンドルを取得するには

1. DTI セッションを開始します。

```
// 返されるステータス コードを初期化する
```

```
BTI_LONG status = 0;
// 予約済みのパラメーターを使って PvStart 関数を
// 呼び出す
status = PvStart(0);
```

2. サーバーに接続します。

```
// 変数を初期化する
BTI_LONG status = 0;
BTI_CHAR_PTR uName = "jsmith";
BTI_CHAR_PTR pword = "123";
BTI_CHAR_PTR svrName = "myserver";
BTI_LONG hConn = 0xFFFFFFFF;
// 実行後、hConn にはほかの関数へ渡すための
// 接続ハンドルが格納される
status = PvConnectServer(svrName, uName, pword, &hConn);
// status が 0 でない場合は、ここでエラー処理を行う
```

接続ハンドルは多くの DTI 関数で必要となります。同時に複数の接続を開くことができます。ただし、それぞれの接続または関数について、PvDisconnect() 関数を呼び出してハンドルを解放する必要があります。

```
status = PvDisconnect(phConn);
```

DTI を使って設定 ID を取得する

設定関数の多くは、パラメーターとして設定 ID を使用します。次の手順は、設定 ID を取得するために必要不可欠な関数について説明します。

特定の設定の ID を取得するには

1. 接続ハンドルを取得するために、DTI を使ってサーバーへの接続を行うの手順を実行します。
2. PvConnectServer() で返された接続ハンドルを使って PvGetCategoryList() を呼び出し、カテゴリの一覧を取得します。
3. 各カテゴリについて、設定の一覧および設定数をそれぞれ PvGetSettingList() と PvGetSettingListCount() を使って取得します。

4. 必要な設定を精査します。
5. PvGetSettingInfo() を使って、設定についての情報を取得します。
6. 完了したら、PvDisconnect() を呼び出してサーバーから切断します。
7. PvStop() を呼び出して DTI セッションを終了します。

DTI 構造体をパラメーターとして渡す

多くの関数で、関数呼び出しを行うときに DTI 構造体を渡すことが要求されます。以下のサンプル コードでは、構造体を含む関数呼び出しの例を示します。DTI 構造体の詳細については、DTI 構造体を参照してください。

```
WORD rValue = P_OK;
TABLEMAP* tableList;
WORD tableCount;
rValue = PvGetTableNames(m_DictHandle, &tableList, &tableCount);
```

Distributed Tuning Interface のリファレンス

DTI の目的は、Zen コンポーネントの設定、監視、および診断を行うためのインターフェイスを提供することにあります。DTI によって、お使いのアプリケーションから Zen ユーティリティの機能を利用できるようになります。

以下のトピックでは、インターフェイスとその使用法について説明します。

- DTI 関数リファレンスの使い方
- DTI 関数グループ
- DTI エラー メッセージ
- DTI 構造体
- DTI 呼び出しの順序
- DTI 関数の定義

DTI 関数リファレンスの使い方

各関数に関して、以下の情報を提供します。

- 簡単な説明 - 関数について簡単に説明します。
- 構文 - 関数の C プロトタイプ構文を示します。
- 引数 - 関数の引数について詳しく説明し、関数によって変更される値を示します。「in」とマークされているパラメーターは入力のみです。関数によって変更されません。「out」とマークされているパラメーターには、関数によって変更される値を格納します。「in/out」とマークされているパラメーターには、関数によって入力として使用され、かつ変更される値を格納します。
- 戻り値 - 返される可能性のある値とその意味を一覧表示します。
- 備考 - 関数のパラメーター、結果、使用法についての補足説明をします。
- 例 - 関数の使用法を説明するサンプル コードを示します。
- 関連項目 - 関連する関数とトピックの一覧です。

DTI 関数グループ

Distributed Tuning Interface はいくつかの関数グループに分けられています。グループ分けの要約については、次の表を参照してください。それぞれの関数の説明は、後述のセクションでアルファベット順に示します。

関数グループ	用途	関数の一覧
カタログ catalog.h	データベース カタログ情報を管理します。たとえば、名前付きデータベースを作成、開く、コピー、閉じることや、データ ソース名（DSN）を作成、変更、削除することなどを行います。	PvCheckDbInfo() PvCloseDatabase() PvCopyDatabase() PvCountDSNs() PvCreateDatabase() PvCreateDatabase2() PvCreateDSN()（非推奨） PvCreateDSN2()（非推奨） PvDeleteDSN()（非推奨） PvDropDatabase() PvFreeDbNamesData() PvGetDbCodePage() PvGetDbDataPath() PvGetDbDictionaryPath() PvGetDbFlags() PvGetDbName() PvGetDbNamesData() PvGetDbServerName() PvGetDSN()（非推奨） PvGetDSNEx()（非推奨） PvGetDSNEx2()（非推奨） PvGetEngineInformation() PvListDSNs()（非推奨） PvModifyDatabase() PvModifyDatabase2() PvModifyDSN()（非推奨） PvModifyDSN2()（非推奨） PvOpenDatabase()

関数グループ	用途	関数の一覧
構成 config.h	データベース エンジン、通信マネージャー、およびローカル リクエスター コンポーネントの設定を制御します。	PvCountSelectionItems() PvGetAllPossibleSelections() PvGetBooleanStrings() PvGetBooleanValue() PvGetCategoryInfo() PvGetCategoryList() PvGetCategoryListCount() PvGetLongValue() PvGetSelectionString() PvGetSelectionStringSize() PvGetSelectionValue() PvGetSettingHelp() PvGetSettingHelpSize() PvGetSettingInfo() PvGetSettingList() PvGetSettingListCount() PvGetSettingMap() PvGetSettingUnits() PvGetSettingUnitsSize() PvGetStringType() PvGetStringValue() PvGetStringValueSize() PvGetValueLimit() PvIsSettingAvailable() PvSetBooleanValue() PvSetLongValue() PvSetSelectionValue() PvSetStringValue()
接続 connect.h	DTI セッションの開始と終了、サーバーへの接続、接続されたサーバーの名前の取得、およびサーバーからの切断を行います。	PvConnectServer() PvDisconnect() PvGetServerName() PvStart() PvStop()

関数グループ	用途	関数の一覧
辞書 ddf.h	辞書（DDF）の作成およびクローズと、テーブル、インデックス、ユーザー、グループの作成および削除を行います。	PvAddIndex() PvAddTable() PvAddUserToGroup() PvAlterUserName() PvAlterUserPassword() PvCloseDictionary() PvCreateDictionary()（非推奨） PvCreateGroup() PvCreateUser() PvDropGroup() PvDropIndex() PvDropIndexByName() PvDropTable() PvDropUser() PvFreeTable() PvFreeTableNames() PvGetError() PvGetTable() PvGetTableNames() PvGetTableStat() PvGetTableStat2() PvGetTableStat3() PvOpenDictionary()（非推奨） PvRemoveUserFromGroup()
ライセンス管理 dtilicense.h	キーの認証や認証解除、キーに関する情報の取得など、ライセンス管理を行います。	PvAddLicense() PvValidateLicenses() PvDeleteLicense() PvGetProductsInfo()

関数グループ	用途	関数の一覧
監視と診断 monitor.h	ファイル、クライアント、および SQL 接続について、MicroKernel エンジンの以下のような情報を監視します。 アクティブ ファイル – 開いているファイルの数と一覧の取得、ファイルが開いているかどうかの照会、ファイルを開いた、またはロックしたユーザーの照会、ページ サイズ、リードオンリー フラグ、レコード ロック、トランザクション ロック、およびハンドル数の取得、各ハンドルのハンドル情報の取得。 アクティブ クライアント – クライアントの数と一覧の取得、アクティブ ハンドルの照会、クライアント情報の取得、ハンドル情報の取得、クライアントおよび全クライアント機能の切断。 リソース使用状況 – ファイル数、ハンドル数、クライアント数、ワーカ スレッド数、使用中ライセンス数、トランザクション数、ロック数を含む、データの現在値、ピーク値、および最大値の取得。 通信統計情報 – すべての通信統計情報、つまり、通信に関するデータの合計値、増加値、現在値、ピーク値、最大値の取得、増加値のリセット機能。	PvDisconnectMkdeClient() PvDisconnectSQLConnection() PvFreeMkdeClientsData() PvFreeOpenFilesData() PvFreeSQLConnectionsData() PvGetFileHandlesData() PvGetFileHandleInfo() PvGetFileInfo() PvGetMkdeClientId() PvGetMkdeClientInfo() PvGetMkdeClientHandlesData() PvGetMkdeClientHandleInfo() PvGetMkdeClientsData() PvGetMkdeCommStat() PvGetMkdeCommStatEx() PvGetMkdeUsage() PvGetMkdeUsageEx() PvGetMkdeVersion() PvGetOpenFilesData() PvGetOpenFileName() PvGetSQLConnectionsData() PvGetSQLConnectionInfo()
セキュリティ dtisecurity.h	データベース セキュリティの有効化 / 無効化、またはセキュリティ状態の照会	PvIsDatabaseSecured() PvSecureDatabase() PvUnSecureDatabase()

DTI エラー メッセージ

定義されているステータス コードについては、*dticonst.h* および *ddfstrct.h* を参照してください。

DTI 構造体

DTI で使用される構造体について、以下に説明します。各構造体グループでは、含まれる構造体のタイプ、および必要となる設定や引数について詳細に説明します。構造体は以下のファイルに格納されています。

- CONFIG.H
- DDFSTRCT.H
- MONITOR.H

各構造体に固有の情報については、その構造体に対応するヘッダー ファイルを参照してください。

CONFIG.H 構造体

以下に CONFIG.H に含まれる構造体の一覧を示します。これらの構造体の詳細については、config ヘッダー ファイルを参照してください。

- PVCATEGORYINFO
- PVSETTINGINFO

DDFSTRCT.H 構造体

以下に DDFSTRCT.H に含まれる構造体の一覧を示します。これらの構造体の詳細については、ddf ヘッダー ファイルを参照してください。

- TABLEMAP
- TABLEINFO
- TABLEINFO フラグ

```
B_FLAG_TRUE_NULLABLE = 64
```

  テーブルは真のヌル値を許可します。テーブルが作成されるとき、ヌル値を許可する各列の前に 1 バイトのヌル インジケーターが追加されます。

- TABLESTAT

systemDataKey（以前の systemData）フィールドの値は、システム データが存在しない場合は 0、システム データまたはシステム データ v2 が存在する場合は 1 になります。

- TABLESTAT2

  TABLESTAT2 と TABLESTAT の相違点を参照してください。

- TABLESTAT3

  TABLESTAT3 と TABLESTAT2 の相違点を参照してください。

- COLUMNMAP
- COLUMNMAP フラグ

```
B_FLAG_CASE_SENSITIVE = 1
```

列の値は、比較する際、つまりインデックス セグメントの一部として比較する際に大文字と小文字が区別されます。

```
B_FLAG_NULLABLE = 4
```

真のヌル値を許可するようにテーブルを作成すると、列の値がヌルであるかどうかを示すために、列の値の前に 1 バイトのヌル インジケーターが追加されます。

```
B_FLAG_NTEXT = 2048
```

列が B_TYPE_BLOB として作成されている場合、データは文字データではなくワイド文字として扱われます。

```
B_FLAG_BINARY = 4096
```

列が B_TYPE_STRING または B_TYPE_BLOB として作成されている場合、データは文字データではなくバイナリとして扱われます。

- COLUMNMAP データ型

  COLUMNMAP データ型は以下の値を取ります。

```
B_TYPE_STRING = 0,
B_TYPE_INTEGER = 1,
B_TYPE_FLOAT = 2,
B_TYPE_DATE = 3,
B_TYPE_TIME = 4,
B_TYPE_DECIMAL = 5,
B_TYPE_MONEY = 6,
B_TYPE_LOGICAL = 7,
B_TYPE_NUMERIC = 8,
B_TYPE_BFLOAT = 9,
B_TYPE_LSTRING = 10,
B_TYPE_ZSTRING = 11,
B_TYPE_NOTE = 12,
```

```
B_TYPE_LVAR = 13,
B_TYPE_BINARY = 14,
B_TYPE_AUTOINC = 15,
B_TYPE_BIT = 16,
B_TYPE_NUMERSTS = 17,
B_TYPE_NUMERSA = 18,
B_TYPE_CURRENCY = 19,
B_TYPE_TIMESTAMP = 20,
B_TYPE_BLOB = 21,
B_TYPE_GDECIMAL = 22,
B_TYPE_WSTRING = 25,
B_TYPE_WZSTRING = 26,
B_TYPE_GUID = 27,
B_TYPE_DATETIME = 30
```

- INDEXMAP
- INDEXMAP フラグ

```
B_FLAG_DUPLICATES = 1
```

インデックスでの重複を許可します。

```
B_FLAG_MODIFIABLE = 2
```

インデックスの変更が可能です。

```
B_FLAG_SORT_DESCENDING = 64
```

インデックスを降順にソートします。

```
B_FLAG_PARTIAL = 512
```

インデックスは部分的です。セグメント上の部分インデックス フラグがインデックスの最後のセグメントではない場合、その部分インデックス フラグは無視されます。部分インデックスが適用されるのは、インデックスの最後のセグメントのみです。

TABLESTAT2 と TABLESTAT の相違点

TABLESTAT2 構造体と TABLESTAT 構造体との間の以下の相違点に注意してください。

- **tableName** フィールドおよび **tableLocation** フィールドではより多くの文字数を使用できます。
- **numberOfRecords** フィールドは 16 ビットから 32 ビットに増加します。
- ファイル属性フィールドは、以前は "Y" または "N" の値を持つ文字フィールドで、属性が存在するかしないかを示していました。現在の属性フィールドは、1 または 0 の値を持つ 1 バイトの整数です。値 1 は属性が存在することを意味します。
- **freespaceThreshold** フィールドは整数データ型になりました。

- **fileVersion** フィールドは、浮動小数点数型ではなくなりました。現在は 1 バイトの整数で、Btrieve の Stat（15）オペレーションが返すのと同じ値を保持します。9.5 ファイル形式の場合、返される値は 0x95 です。
- 新しいフィールドの **pageCompression** は、テーブルに関連付けられている物理ファイルに圧縮ページがあるかないかを示します。
- 以前の **dataCompression** および **systemDataKey** は、それぞれ **recordCompression** および **systemData** という名前に変更されました。

TABLESTAT3 と TABLESTAT2 の相違点

TABLESTAT3 構造体と TABLESTAT2 構造体との間の以下の相違点に注意してください。

- **numberOfRecords** フィールドは 32 ビットから 64 ビットに増加します。

以前のバージョンとの互換性

Zen クライアントは、今までどおりデータベース エンジンへの PvGetTableStat 呼び出しを行うことができます。データベース エンジンは、クライアントのバージョンに基づいて、応答メッセージを TABLESTAT2 構造体または TABLESTAT 構造体に変換します。

Zen クライアントは、接続しているデータベース エンジンのバージョンを特定します。データベース エンジンのバージョンが現在のリリースより前の場合、PvGetTableStat2 は TABLESTAT 構造体を返し、pageCompression が返す値に 0 を設定します。

MONITOR.H 構造体

以下に MONITOR.H に含まれる構造体の一覧を示します。これらの構造体の詳細については、monitor ヘッダー ファイルを参照してください。

- PVDATETIME
- PVFILEINFO
- PVFILEHDLINFO
- PVCLIENTID
- PVMKDECLIENTINFO
- PVMKDECLIENTHDLINFO

- PVMKDEUSAGE
- PVMKDEUSAGEEX
- PVVERSION
- PVCOMMSTAT
- PVCOMMSTATEX
- PVCOMMPROTOCOLSTAT
- PVSQLCONNINFO
- PVSQLCONNID

DTI 呼び出しの順序

すべての Distributed Tuning Interface 呼び出しは、まず PvStart() を呼び出すことによって DTI セッションを初期化しておく必要があります。

```
status = PvStart(0);

// ここに、複数の DTI 関数呼び出しを挿入します

status = PvStop (0);
```

各関数の備考には、その特定の関数に対し、実行前および実行後に必要となる条件が挙げられています。

DTI 関数の定義

このトピックでは、DTI 関数のリファレンスをアルファベット順で提供します。

PvAddIndex()

indexList で指定されるインデックスを、既存のテーブルおよび基となるデータ ファイルに追加します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvAddIndex(
    WORD            dictHandle,
    LPCSTR          tableName,
  INDEXMAP*         indexList,
    WORD            indexCount);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	インデックスが追加されるテーブル名。
In	*indexList*	インデックス定義の配列。
In	*indexCount*	*indexList* 配列内のインデックスの数。

戻り値

PCM_Success	操作は正常に終了しました。
PCM_errFailed	操作は正常に終了しませんでした。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。
PCM_errTableNotFound	指定されたテーブルが見つかりません。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidIndexName	指定されたインデックス名が無効です。
PCM_errColumnNotFound	指定された列がテーブル内に見つかりません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableName によって指定されたテーブルが、*dictHandle* で指定された辞書の中になければなりません。

インデックスの説明に使用する INDEXMAP 配列を割り当てて解放する必要があります。

関連項目

PvStart()
PvOpenDatabase()
PvDropIndex()
PvDropIndexByName()

PvCloseDictionary()
PvStop()

PvAddLicense()

接続によって示されるコンピューターの指定したライセンスを適用（認証）します。

ヘッダー ファイル：dtilicense.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav80.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvAddLicense(
   BTI_LONG        hConnection,
   BTI_CHAR_PTR    license);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*license*	適用（認証）するライセンス。

戻り値

P_OK	操作は正常に終了しました。
P_E_FAIL	操作は正常に終了しませんでした。
P_E_LIC_ALREADY_INSTALLED	ライセンスは既に適用されています。
P_E_LIC_INVALID	指定されたライセンスが無効です。
ライセンス管理または認証に関するステータス コード	License Administrator のステータス コードおよび許可のステータス コードは、『*Status Codes and Messages*』を参照してください。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

例

```
BTI_CHAR_PTR add_lic = "ERXVD3U4ZS9KR94QPDHV5BN2";
status = PvAddLicense(P_LOCAL_DB_CONNECTION, add_lic);
```

関連項目

PvValidateLicenses()
PvDeleteLicense()
PvGetProductsInfo()
PvStop()
PvStart()

PvAddTable()

既存の辞書とデータ ファイルの、テーブル プロパティで指定された位置に、新規のテーブルを作成します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvAddTable(
    WORD              dictHandle,
  TABLEINFO*          tableProps,
  COLUMNMAP*            columnList,
    WORD              columnCount,
  INDEXMAP*           indexList,
    WORD              indexCount);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableProps*	テーブル情報を含む構造体。
In	*columnList*	テーブルに定義された列の配列。
In	*columnCount*	columnList 内の列数。
In	*indexList*	インデックス定義の配列。
In	*indexCount*	indexList 配列内のインデックス数。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。
PCM_errTableNotFound	指定されたテーブルが見つかりません。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。

PCM_errInvalidColumnName	指定された列名が無効です。
PCM_errInvalidDataType	指定されたデータ型が無効です。
PCM_errDuplicateColumnName	この列名は既にテーブルに存在します。
PCM_errInvalidDataSize	データ サイズが無効です。
PCM_errInvalidIndexName	インデックス名が無効です。
PCM_errColumnNotFound	セグメントに指定された列が見つかりません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

この関数にはテーブル情報、列、およびインデックスを指定する必要があります。テーブルの作成にインデックスは必須ではないため、*IndexCount* パラメーターと *indexList* パラメーターは任意です。

同じ名前のテーブルが指定された辞書に既に存在している場合、この関数は失敗します。

テーブル プロパティは正しく設定する必要があり、少なくとも 1 列の配列を渡さなければなりません。

テーブルの説明に使用する COLUMNMAP、INDEXMAP 配列と TABLEINFO 構造体を割り当てて解放する必要があります。COLUMNMAP フラグも参照してください。

PvGetTable() 関数で、行内でのフィールドのオフセットにアクセスできます。ddfstrct.h の COLUMNMAP 構造体は変更されて、この追加情報を持つようになりました。この新しいフィールドは、PvAddTable() および PvFreeTable() 関数を呼び出すときは無視されます。ddfstrct.h および ddf.h を参照してください。

関連項目

PvStart()
PvOpenDatabase()
PvGetTableNames()
PvFreeTableNames()
PvDropTable()

PvCloseDictionary()
PvStop()

PvAddUserToGroup()

既存ユーザーをデータベースの既存グループに追加します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvAddUserToGroup(
   BTI_WORD            dbHandle,
   const BTI_CHAR*     user,
   const BTI_CHAR*     group);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名
In	*group*	データベース グループ名

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたアカウントまたはユーザー名は存在しません。
PCM_errUserAlreadyPartOfGroup	ユーザーは既にグループの一員です。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

この関数は、指定したグループまたはユーザーがデータベースにあらかじめ存在していない場合や、ユーザーが別のグループのメンバーである場合は失敗します。

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。
- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- ユーザーおよびグループは指定したデータベースに既に存在している。
- ユーザーは別のグループのメンバーではない。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAlterUserName()
PvCreateGroup()
PvCreateUser()
PvDropGroup()
PvDropUser()
PvRemoveUserFromGroup()
PvOpenDatabase()
PvCloseDatabase()

PvAlterUserName()

指定されたデータベースの既存のユーザーの名前を変更します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvAlterUserName(
   BTI_WORD             dbHandle,
   const BTI_CHAR*      user,
   const BTI_CHAR*      newName);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名。
In	*newName*	データベース ユーザーの新しい名前。ヌルを設定すると関数は失敗します。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	アカウントまたはユーザー名が存在しないか、新しい名前が無効です。
PCM_errUserAlreadyExists	新しいユーザー名は既に存在します。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

この関数は、*newName* にヌルが設定されていたり *newName* が既にデータベースに存在すると失敗します。

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、辞書を正常に開いておく。
- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- ユーザー名は指定したデータベースに既に存在している。
- 新しいユーザー名が指定したデータベースに存在していない。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAlterUserPassword()
PvAddUserToGroup()
PvRemoveUserFromGroup()
PvCreateUser()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvAlterUserPassword()

既存のユーザーのパスワードを変更します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvAlterUserPassword(
   BTI_WORD            dbHandle,
   const BTI_CHAR*     user,
   const BTI_CHAR*     newPassword);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名。
In	*newPassword*	新しいユーザー パスワード。ヌルを設定するとパスワードがクリアされます。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたアカウントまたはユーザー名は存在しません。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。
- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- ユーザー名は指定したデータベースに既に存在している。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAlterUserName()
PvAddUserToGroup()
PvRemoveUserFromGroup()
PvCreateUser()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvCheckDbInfo()

データベースの整合性をチェックします。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvCheckDbInfo(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   dbName,
   BTI_ULONG      checkFlags);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	既存の名前付きデータベース名。特定サーバーの名前付きデータベースの全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つの名前付きデータベースを取得するには、PvGetDbName() 関数を使用します。
In	*checkFlags*	予約済み関数はすべてのデータベース フラグをチェックします。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	サーバーを識別する接続ハンドルが無効です。
P_E_NULL_PTR	関数はヌル ポインターによって呼び出されました。
P_E_ACCESS_RIGHT	関数を呼び出すのに十分なアクセス権がありません。
P_E_NOT_EXIST	*dbName* で指定された名前付きデータベースは存在しません。
P_E_FAIL	一般エラーが発生しました。

備考

データベースに一貫性がある場合、この関数の戻り値は P_OK になります。データベースに一貫性がない場合、または関数呼び出しが失敗した場合、戻り値は上に挙げたエラー コードのいずれかになります。

例

```
BTI_WORD     res;          // 関数呼び出しから返される値
BTI_CHAR_PTR dbName;       // データベース名
BTI_ULONG    checkFlags;   // データベース フラグ
BTI_LONG     hConnection; // 接続ハンドル
BTI_LONG     reserved;
  // PvStart() および PvStop() 用に予約されている値

  // 変数を初期化する
dbName    = "demodata";
  // データベースの名前は「demodata」
checkFlags = 0xFFFFFFFF;   // すべてのフラグをチェックする
hConnection = P_LOCAL_DB_CONNECTION;
  // 接続ハンドルをローカル接続に設定する

  // P_LOCAL_DB_CONNECTION は config.h に定義されている
reserved = 0;

 // DTI 呼び出しを実行する前に、DTI セッションを開始する
res = PvStart (reserved);

if (res == P_OK)
 {
    // DTI セッションは正常に開始されました
    // これで、さまざまな DTI 呼び出しを実行できます

         res = PvCheckDbInfo (hConnection,
                              dbName,
                              checkFlags);

         if (res == P_OK)
           {
       // データベースは整合しています
}
           else
            {
       // ここに、PvCheckDbInfo() から返されたエラー コード
       // を処理するコードを記述します
}
      // DTI セッションを閉じる
            Res = PvStop (&reserved);
}
```

関連項目

PvStart()
PvConnectServer()

PvGetDbNamesData()
PvGetDbName()
PvFreeDbNamesData()
PvDisconnect()
PvStop()

PvCloseDatabase()

開いているデータベース ハンドルを閉じます。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvCloseDatabase(
   BTI_WORD       dbHandle);
```

引数

In	*dbHandle*	PvOpenDatabase() によって開かれたデータベースのハンドル

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errDictionaryNotOpen	指定されたハンドルでデータベースは開いていません。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvOpenDatabase() によって返された有効なデータベース ハンドルがある。

関連項目

PvStart()
PvConnectServer()
PvOpenDatabase()
PvDisconnect()
PvStop()

PvCloseDictionary()

開いている辞書を閉じます。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvCloseDictionary(
   WORD              dictHandle);
```

引数

In	*dictHandle*	開いているか新規に作成した辞書のハンドル。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errDictionaryNotOpen	指定された辞書は開いていませんでした。

備考

この関数は、開いている辞書ファイルのハンドルを必要とします。このハンドルは、PvCreateDictionary() 関数を使って取得できます。

同時に複数の辞書を開くことができるため、辞書を開いたり新規に作成するたびにこの関数を呼び出す必要があります。

例

```
PRESULT status = 0;
status = PvCloseDictionary(myDictionaryHandle);
```

関連項目

PvStart()
PvOpenDatabase()
PvCreateDictionary()
PvStop()

PvConnectServer()

Zen データベース エンジンがインストールされているターゲット サーバーに接続を試みます。接続が正常に確立すると、以降の参照用の接続ハンドルが返されます。

ヘッダー ファイル：connect.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvConnectServer(
   BTI_CHAR_PTR        serverName,
   BTI_CHAR_PTR        userName,
   BTI_CHAR_PTR        password,
   BTI_LONG_PTR        phConnection);
```

引数

In	*serverName*	接続先のサーバー名または IP アドレス。 『*Getting Started with Zen*』のドライブ ベースの形式も参照してください。
In	*userName*	*serverName* に接続するユーザー名。このパラメーターを省略する場合の説明については、下記の「備考」を参照してください。
In	*password*	ユーザー パスワード。このパラメーターを省略する場合の説明については、下記の「備考」を参照してください。
In/Out	*phConnection*	接続が成功した場合に、接続ハンドルを受け取る Long 型整数のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	名前付きサーバーとの接続に失敗しました。
P_E_SERVER_NOT_FOUND	指定されたサーバーは見つかりません。

P_E_ENGINE_NOT_LOADED	指定されたエンジンは実行されていません。
P_E_REQUESTER_NOT_LOADED	クライアント リクエスターがロードされていません。
P_E_SERVER_TABLE_FULL	内部サーバー名テーブルがいっぱいです。
P_E_CLIENT_CONNECTIONS_LIMIT_REACHED	クライアント接続の制限に達したため、接続できませんでした。サーバーの設定をチェックしてください。
P_E_PERMISSION_ERROR	操作でアクセス許可エラーが発生しました。
P_E_NO_MEMORY	操作でメモリ エラーが発生しました。
P_E_NO_AVAILABLE_TRANSPORT	リモート接続を一切確立できませんでした。
P_E_CONNECTION_LOST	サーバーへのリモート接続が失われました。

備考

接続先のサーバー名を知っておく必要があります。複数のサーバーに対して開かれた接続を持つことができます。

データベース エンジンが起動しているローカルで実行されているアプリケーションでは、ユーザー名とパスワードを省略できます。省略しても、任意の DTI 関数の呼び出しや、すべての設定の表示および変更は行えます。

ただし、DTI アプリケーションがターミナル サービス セッションを介して実行されている場合、あるいはリモートで実行されている場合は、サーバー マシン上で、管理者レベルの権限を持つユーザーのユーザー名とパスワードを提供してください。これにより、アプリケーションが DTI 関数へのフル アクセスを持つようになります。管理者レベル権限がない場合、アプリケーションはほとんどの DTI 関数に対してアクセス エラーを返します。関数のサブセットのみが動作します。たとえば、フル アクセスが許可されている場合に設定を変更できる関数の多くは、読み取り専用のアクセスに制限されます。

メモ： この関数を使ってサーバーへの接続を試みる前に、PvStart() を呼び出して DTI を初期化しておく必要があります。

例

```
BTI_CHAR_PTR uName = "jsmith";
BTI_CHAR_PTR pword = "123";
BTI_CHAR_PTR svrName = "myserver";
BTI_LONG_PTR phConn = 0xFFFFFFFF;
BTI_SINT status = 0;

status = PvConnectServer(svrName,
                         uName,
                         pword,
                         &phConn);
```

関連項目

PvStart()
PvGetServerName()
PvDisconnect()
PvStop()

PvCopyDatabase()

データベースを新規データベースにコピーし、必要に応じて参照整合性を調整します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvCopyDatabase(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR_PTR      newdbName,
   BTI_CHAR_PTR      newdictPath,
   BTI_CHAR_PTR      newdataPath);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	コピーするデータベースの名前。
In	*newdbName*	新しいデータベースの名前。
In	*newdictPath*	新規データベースの辞書パス。
In	*newdataPath*	データ パス。デフォルトのデータ パス（つまり、辞書パスと同じパス）を使用するには、空文字列を渡します。 複数のパスに保存された MicroKernel エンジン データ ファイルから構成される新規データベースを作成する場合は、このパラメーターにセミコロン（;）で区切られた一覧を指定します。たとえば、`C:¥data¥path1;C:¥data¥path2` のように指定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_DICTIONARY_ALREADY_EXISTS	辞書は既に存在するため、作成できません。
P_E_SHARED_DDF_EXIST	辞書パスは別のデータベースが使用中です。
P_E_DUPLICATE_NAME	名前付きデータベースはサーバーに既に存在します。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- データベースとデータベースのファイルは閉じている必要がある。
- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

例

```
BTI_LONG connectionHandle = P_LOCAL_DB_CONNECTION;
BTI_CHAR_PTR newdataPath = "c:¥¥data¥¥gallery2";
BTI_CHAR_PTR newdictPath = "c:¥¥data¥¥gallery2";
BTI_CHAR_PTR databaseName = "Gallery";
BTI_CHAR_PTR newdatabaseName = "GalleryCopy";
BTI_SINT status = 0;
BTI_CHAR_PTR server = "MyServer";
BTI_CHAR_PTR user = "Administrator";
BTI_CHAR_PTR pwd = "Admin";
// リモートの場合のみ、サーバーへの接続が必要となる
// リモートでない場合は、ハンドルとして P_LOCAL_DB_CONNECTION を渡すことができる

status = PvCopyDatabase(
connectionHandle,
databaseName,
newdatabaseName
dictPath,
dataPath);
```

関連項目

PvStart()
PvConnectServer()

PvCreateDatabase()
PvGetDbFlags()
PvModifyDatabase()
PvDropDatabase()
PvDisconnect()
PvStop()

PvCountDSNs()

データ ソース名（DSN）の数を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvCountDSNs(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pdsnCount,
   BTI_CHAR          filtering);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pdsnCount*	DSN の数を受け取るための符号なし Long 型のアドレス。
In	*filtering*	Zen の DSN のみが必要な場合は 1 を設定します。すべての DSN が必要な場合は 0 を設定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

ユーザーにログイン要求をしないで DSN の数を取得するには、PvConnectServer() を使ってサーバー接続を確立するとき、userName と password に空文字列を渡します。

メモ： userName と password に空文字列を渡して確立した接続は、セキュリティで保護されていない接続であるため、DTI のこれ以外のほとんどの操作を実行するのに十分なアクセス権を持たない接続となります。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvGetDSN()
PvDisconnect()
PvStop()

PvCountSelectionItems()

選択タイプ（PVSETTING_SINGLE_SEL または PVSETTING_MULTI_SEL）の設定用の選択項目の数を数えます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvCountSelectionItems(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pNumItems);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。カテゴリの一覧は、PvGetCategoryList() 関数を使って取得できます。特定のカテゴリの設定の一覧は、PvGetSettingList() から取得できます。
Out	*pNumItems*	選択項目の数を受け取る符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は選択タイプの設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvCreateDatabase()

dbnames.cfg ファイルへエントリを追加することによって、データベースを作成します。このエントリは、後で DSN の作成に使用されます。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvCreateDatabase(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR_PTR      dictPath,
   BTI_CHAR_PTR      dataPath,
   BTI_ULONG         dbFlags);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは **PvConnectServer()** 関数によって取得されます。
In	*dbName*	データベース名。
In	*dictPath*	辞書パス。
In	*dataPath*	データ パス。デフォルトのデータ パス（つまり、辞書パスと同じパス）を使用するには、空文字列を渡します。 複数のパスに保存された MicroKernel エンジン データ ファイルから構成されるデータベースを作成する場合は、このパラメーターにセミコロン（;）で区切られた一覧を指定します。たとえば、`C:¥data¥path1;C:¥data¥path2` のように指定します。

In	*dbFlags*	データベース フラグ。P_DBFLAG_ 定数を組み合わせて指定できます。 • P_DBFLAG_RI（参照整合性およびトリガーを含む、整合性制約を設定します。） • P_DBFLAG_BOUND。DDF ファイルを作成してデータベース名を辞書ファイルにスタンプし、そのデータベースのみが DDF を使用できるようにします。データベースがバインドされていない場合は、複数のデータベースで同一の辞書ファイル セットを使用できます。バウンド データベースを作成するときに既存の DDF ファイルにバインドしたい場合は、P_DBFLAG_CREATE_DDF と P_DBFLAG_BOUND の両方を指定します。 • P_DBFLAG_CREATE_DDF（DDF ファイルを作成します。*dictPath* に指定されたディレクトリは存在している必要があります。） • P_DBFLAG_DBSEC_AUTHENTICATION（データベース セキュリティ認証の混合セキュリティを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_DBSEC_AUTHORIZATION（データベース セキュリティ認証のデータベース セキュリティ ポリシーを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_LONGMETADATA（メタデータ バージョン 2 を使用します。メタデータのバージョンを参照してください。）

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_DICTIONARY_ALREADY_EXISTS	辞書は既に存在するため、作成できません。
P_E_SHARED_DDF_EXIST	辞書パスは別のデータベースが使用中です。
P_E_DUPLICATE_NAME	名前付きデータベースはサーバーに既に存在します。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

Btrieve セキュリティ ポリシー

次の表は、新規データベースでセキュリティ モデルを指定する方法、および既存データベースのセキュリティ モデルを解釈する方法を示しています。セキュリティにほかのフラグの組み合わせを使用すると、ステータス コード 7024 が返される結果となります。

フラグの組み合わせ	相当するセキュリティ モデル
フラグなし	クラシック
P_DBFLAG_DBSEC_AUTHENTICATION	混合
P_DBFLAG_DBSEC_AUTHENTICATION + P_DBFLAG_DBSEC_AUTHORIZATION	データベース

メタデータのバージョン

P_DBFLAG_LONGMETADATA を指定した場合、dbnames.cfg のデータベース プロパティはメタデータ バージョン 2 に設定されます。P_DBFLAG_LONGMETADATA と P_DBFLAG_CREATE_DDF を指定した場合は、作成される DDF もメタデータ バージョン 2 です。

DDF 作成の結果は、辞書の場所に既に存在する DDF のバージョンによって異なります。

辞書の場所にあるもの	DDF 作成の結果
DDF なし	辞書の場所に新規 DDF が追加されます。
別のメタデータ バージョンの DDF	既存の DDF のグループに新規 DDF が追加されます。
同じメタデータ バージョンの DDF	新規 DDF が既存の DDF を上書きします。古い DDF に含まれる情報は失われます。

たとえば、辞書の場所にメタデータ バージョン 1 DDF があってメタデータ バージョン 2 DDF を作成するとします。辞書の場所には、メタデータ バージョン 1 DDF とメタデータ バージョン 2 DDF の組み合わせが含まれることになります。あるデータベースは 1 組の DDF またはほかの 1 組の DDF を使用できますが、両方を同時に使用することはできません。

例

以下の例では、メタデータ バージョン 2 を使用するデータベースと DDF を作成します。

```
BTI_LONG connectionHandle = P_LOCAL_DB_CONNECTION;
BTI_CHAR_PTR dataPath = "c:¥¥data¥¥gallery";
BTI_CHAR_PTR dictPath = "c:¥¥data¥¥gallery";
BTI_CHAR_PTR databaseName = "Gallery";
BTI_SINT status = 0;
BTI_CHAR_PTR server = "MyServer";
BTI_CHAR_PTR user = "Administrator";
BTI_CHAR_PTR pwd = "Admin";
// リモートの場合のみ、サーバーへの接続が必要となる
// リモートでない場合は、ハンドルとして P_LOCAL_DB_CONNECTION を渡すことができる

status = PvCreateDatabase(
connectionHandle,
databaseName,
dictPath,
dataPath,
```

```
P_DBFLAG_CREATE_DDF,
P_DBFLAG_LONGMETADATA);
```

関連項目

PvStart()
PvConnectServer()
PvGetDbFlags()
PvModifyDatabase()
PvDropDatabase()
PvDisconnect()
PvStop()

PvCreateDatabase2()

dbnames.cfg ファイルへエントリを追加することによって、データベースを作成します。この関数は、データベース コード ページが指定されることを除けば、PvCreateDatabase() とまったく同じです。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvCreateDatabase2(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       dbName,
   BTI_CHAR_PTR       dictPath,
   BTI_CHAR_PTR       dataPath,
   BTI_ULONG          dbFlags,
   BTI_LONG           dbCodePage);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは **PvConnectServer()** 関数によって取得されます。
In	*dbName*	データベース名。
In	*dictPath*	辞書パス。
In	*dataPath*	データ パス。デフォルトのデータ パス（つまり、辞書パスと同じパス）を使用するには、空文字列を渡します。 複数のパスに保存された MicroKernel エンジン データ ファイルから構成されるデータベースを作成する場合は、このパラメーターにセミコロン（;）で区切られた一覧を指定します。たとえば、C:¥data¥path1;C:¥data¥path2 のように指定します。

In	*dbFlags*	データベース フラグ。P_DBFLAG_ 定数を組み合わせて指定できます。 • P_DBFLAG_RI（参照整合性およびトリガーを含む、整合性制約を設定します。） • P_DBFLAG_BOUND（DDF ファイルを作成してデータベース名を辞書にスタンプし、そのデータベースのみが DDF を使用できるようにします。データベースがバインドされていない場合は、複数のデータベースで同一の辞書ファイル セットを使用できます)。バウンド データベースを作成するときに既存の DDF ファイルにバインドしたい場合は、P_DBFLAG_CREATE_DDF と P_DBFLAG_BOUND の両方を指定します。 • P_DBFLAG_CREATE_DDF（DDF ファイルを作成します。*dictPath* に指定されたディレクトリは存在している必要があります。） • P_DBFLAG_DBSEC_AUTHENTICATION（データベース セキュリティ認証の混合セキュリティを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_DBSEC_AUTHORIZATION（データベース セキュリティ認証のデータベース セキュリティ ポリシーを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_LONGMETADATA（メタデータ バージョン 2 を使用します。メタデータのバージョンを参照してください。）

In	*dbCodePage*	Windows プラットフォームのデータベースの場合、番号によってデータベース データとメタデータ文字列のコード ページを示します。 Linux ディストリビューションのデータベースの場合、以下の値のいずれかによってデータベース データとメタデータ文字列のコード ページを示します。 • P_DBCODEPAGE_UTF8 • P_DBCODEPAGE_EUCJP • P_DBCODEPAGE_ISO8859_1 Windows および Linux のデータベースでは、ゼロの値も使用できます。 ゼロは旧来の動作を示します。つまり、コード ページは指定されません。サーバー マシンのオペレーティング システムにおけるエンコードをデフォルトで使用します。『*Zen User's Guide*』のコード ページ データベース プロパティも参照してください。 **メモ：**データベース エンジンは、アプリケーションがデータベースに追加するデータおよびメタデータのエンコードを検証**しません**。エンジンは、すべてのデータが、『*Advanced Operations Guide*』のデータベース コード ページとクライアント エンコードで説明されているようにサーバーまたはクライアントのエンコードを使用して入力されるものと想定しています。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_DICTIONARY_ALREADY_EXISTS	辞書は既に存在するため、作成できません。
P_E_SHARED_DDF_EXIST	辞書パスは別のデータベースが使用中です。
P_E_DUPLICATE_NAME	名前付きデータベースはサーバーに既に存在します。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

Btrieve セキュリティ ポリシーとメタデータのバージョン

それぞれについて、Btrieve セキュリティ ポリシーおよびメタデータのバージョンを参照してください。

関連項目

PvConnectServer()
PvCreateDSN2()
PvDisconnect()
PvDropDatabase()
PvGetDbCodePage()
PvGetDbFlags()
PvGetDSNEx2()
PvModifyDatabase2()
PvStart()
PvStop()

PvCreateDictionary()

新しい辞書ファイル セットを作成します。辞書の完全修飾パスを指定すると、辞書ハンドルが返されます。このハンドルは、以降のカタログ関数の呼び出しで使用されます。

メモ： この関数は Zen 9 以降のバージョンでは使用が推奨されません。アプリケーションでこの関数を置き換えるには、PvCreateDatabase() と PvOpenDatabase() を参照してください。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvCreateDictionary(
   LPCSTR          path,
   WORD*           dictHandle,
   LPCSTR          user,
   LPCSTR          password);
```

引数

In	*path*	辞書ファイルへの完全修飾パス。
Out	*dictHandle*	以降の呼び出しで使用するハンドル。
In	*user*	新しい辞書で使用されるユーザー名。この引数はヌルに設定できます。
In	*password*	辞書ファイルを新規作成するために、ユーザー名と組み合わせて使用します。ヌルでもかまいません。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errPathNotFound	指定されたパスが無効です。

PCM_errSessionSecurityError	ユーザー名かパスワードのどちらかが無効です。
PCM_errDictionaryAlreadyExists	DDF ファイルのセットは、指定された場所に既に存在します。

備考

リソースを解放するには、PvCloseDictionary() を使用します。

関連項目

PvStart()
PvOpenDatabase()
PvGetDbDictionaryPath()
PvCloseDictionary()
PvStop()

PvCreateDSN()

新しいエンジンのデータ ソース名（DSN）を作成します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN の作成には ODBC API を使用してください（Linux の場合は **dsnadd** ユーティリティ）。

構文

```
BTI_API PvCreateDSN(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   pdsnName,
   BTI_CHAR_PTR   pdsnDesc,
   BTI_CHAR_PTR   pdsnDBQ,
   BTI_LONG       openMode);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pdsnName*	新しい DSN 名。
In	*pdsnDesc*	新しい DSN の説明。
In	*pdsnDBQ*	この DSN が接続するデータベース名。これは既存の名前でなければなりません。データベース名の作成方法は、PvCreateDatabase() を参照してください。
In	*OpenMode*	DSN のオープン モード。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_NAME	指定された DSN 名が無効です。
P_E_DSN_ALREADY_EXIST	指定された DSN 名は既に存在します。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_INVALID_OPEN_MODE	指定されたオープン モードが無効です。
P_E_FAIL	データ パスの検索に失敗しました。

備考

この関数はエンジン DSN のみ作成します。クライアント DSN を作成するには、ODBC API を使用する必要があります。

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- *pdsnDBQ* パラメーターで参照されるデータベース名は既に存在している。データベース名の作成方法は、PvCreateDatabase() を参照してください。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvModifyDSN()
PvGetDSN()
PvGetDSNEx()

PvDeleteDSN()
PvCountDSNs()
PvStop()

PvCreateDSN2()

新しいエンジンのデータ ソース名（DSN）を作成し、データのエンコード オプションを指定します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN の作成には ODBC API を使用してください（Linux の場合は **dsnadd** ユーティリティ）。

構文

```
BTI_API PvCreateDSN2(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   pdsnName,
   BTI_CHAR_PTR   pdsnDesc,
   BTI_CHAR_PTR   pdsnDBQ,
   BTI_LONG       openMode,
   BTI_LONG       translate);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pdsnName*	新しい DSN 名。
In	*pdsnDesc*	新しい DSN の説明。
In	*dsnDBQ*	この DSN が接続するデータベース名。これは既存の名前でなければなりません。データベース名の作成方法は、PvCreateDatabase() を参照してください。
In	*OpenMode*	DSN のオープン モード。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE 『*ODBC Guide*』の DSN オープン モードも参照してください。

In	*translate*	データのエンコード オプション。次のいずれか 1 つになります。 • DSNFLAG_DEFAULT • DSNFLAG_OEMANSI • DSNFLAG_AUTO 『*ODBC Guide*』の**エンコード変換**も参照してください。 DSNFLAG_DEFAULT は ODBC アドミニストレーターのエンコード オプション［なし］に該当します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_NAME	指定された DSN 名が無効です。
P_E_DSN_ALREADY_EXIST	指定された DSN 名は既に存在します。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_INVALID_OPEN_MODE	指定されたオープン モードが無効です。
P_E_INVALID_TRANSLATE_OPTION	指定されたエンコード変換オプションが無効です。
P_E_FAIL	データ パスの検索に失敗しました。

備考

この関数はエンジン DSN のみ作成するもので、PSQL v10 クライアント以降が必要です。クライアント DSN を作成するには、ODBC API を使用する必要があります。Linux の場合は、**dsnadd** ユーティリティを使用してクライアント DSN を作成することもできます。

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

- *pdsnDBQ* パラメーターで参照されるデータベース名は既に存在している。データベース名の作成方法は、PvCreateDatabase() を参照してください。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvModifyDSN()
PvGetDSN()
PvGetDSNEx()
PvDeleteDSN()
PvCountDSNs()
PvStop()

PvCreateGroup()

既存のデータベースに新しいユーザー グループを作成します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvCreateGroup(
   BTI_WORD            dbHandle,
   const BTI_CHAR*     group);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*group*	データベース グループ名。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたグループ名が無効です。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。
PCM_errGroupAlreadyExists	グループは既に存在します。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。

- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- 同じ名前のグループが指定したデータベースに存在していない。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAddUserToGroup()
PvRemoveUserFromGroup()
PvCreateUser()
PvAlterUserName()
PvAlterUserPassword()
PvDropGroup()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvCreateUser()

既存のデータベースに新しいユーザーを作成します。オプションとして、パスワードを設定することと新しいユーザーを既存のグループに割り当てることができます。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvCreateUser(
   BTI_WORD            dbHandle,
   const BTI_CHAR*     user,
   const BTI_CHAR*     password,
   const BTI_CHAR*     group);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名。
In	*password*	ユーザー パスワード。ヌルを設定するとパスワードは設定されません。
In	*group*	ユーザーのデータベース グループ名。ヌルを設定するとユーザーはグループに割り当てられません。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたアカウントまたはユーザー名が無効です。
PCM_errUserAlreadyExists	ユーザーは既に存在します。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。
- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- 同じ名前のユーザーが指定したデータベースに存在していない。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAlterUserName()
PvAlterUserPassword()
PvAddUserToGroup()
PvRemoveUserFromGroup()
PvCreateGroup()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvDeleteDSN()

データ ソース名を削除します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvDeleteDSN(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   pdsnName);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pdsnName*	削除する DSN。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_DSN_DOES_NOT_EXIST	指定された DSN 名は存在しません。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvModifyDSN()
PvGetDSN()
PvGetDSNEx()
PvCreateDSN()
PvCountDSNs()
PvStop()

PvDeleteLicense()

接続によって示されるコンピューターの指定したライセンスを削除（認証解除）します。

ヘッダー ファイル：dtilicense.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav80.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvDeleteLicense(
   BTI_LONG        hConnection,
   BTI_CHAR_PTR    licenses);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*licenses*	削除するライセンス。

戻り値

P_OK	操作は正常に終了しました。
P_E_FAIL	操作は正常に終了しませんでした。
P_E_LIC_NOT_FOUND	指定されたライセンスは現在認証されていません。
P_E_LIC_INVALID	指定されたライセンスが無効です。
ライセンス管理または認証に関するステータス コード	License Administrator のステータス コードおよび許可のステータス コードは、『*Status Codes and Messages*』を参照してください。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

例

```
BTI_CHAR_PTR delete_lic = "ERXVD3U4ZS9KR94QPDHV5BN2";
status = PvDeleteLicense(P_LOCAL_DB_CONNECTION, delete_lic);
```

関連項目

PvAddLicense()
PvValidateLicenses()
PvGetProductsInfo()
PvStop()
PvStart()

PvDisconnect()

以前に PvConnectServer 関数によって確立された接続の切断を試みます。

ヘッダー ファイル：connect.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvDisconnect(
   BTI_LONG          hConnection);
```

引数

In	*hConnection*	切断される接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

例

```
BTI_SINT status = 0;
status = PvDisconnect(m_hConn);
```

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeCommStat()
PvGetMkdeUsage()
PvGetOpenFilesData()

PvFreeOpenFilesData()
PvDisconnectMkdeClient()
PvDisconnectSQLConnection()
PvStop()

PvDisconnectMkdeClient()

クライアント ID を指定することによって、アクティブな MicroKernel エンジン クライアントの切断を試みます。有効なクライアント ID を取得するには、PvGetMkdeClientData および PvGetMkdeClientId 関数を使用します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvDisconnectMkdeClient(
    BTI_LONG            hConnection,
    PVCLIENTID*         pClientId);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pClientId*	MicroKernel エンジン クライアントを識別する PVCLIENTID 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_CLIENT	クライアント ID が無効です。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

例

```
unsigned long count = 0;

// この例では、アクティブな MKDE 接続をすべて切断します
BTI_SINT status = 0
PVCLIENTID clientId;
 status = PvGetMkdeClientsData(connection, &count);

while (count > 0)
{
status = PvGetMkdeClientId(connection, 0, &client Id);
status = PvDisconnectMkdeClient(connection, &clientId);
status = PvGetMkdeClientsData(connection, &count);
}
  PvFreeMkdeClientsData(connection);
```

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeClientId()
PvGetMkdeClientInfo()
PvGetMkdeClientHandlesData()
PvDisconnect()
PvStop()

PvDisconnectSQLConnection()

SQL の接続 ID を渡すことによって、アクティブな SQL 接続の切断を試みます。有効な接続 ID を取得するには、PvGetSQLConnectionsData および PvSQLConnectionInfo 関数を使用します。

メモ： SQL 接続ごとに MicroKernel エンジン接続も確立しています。これらの接続を切断するには PvDisconnectMKDEClient を使用します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvDisconnectSQLConnection(
   BTI_LONG           hConnection,
   PVSQLCONNID*       pSQLConnId);
```

引数

In	*hConnection*	切断される SQL 接続を含むサーバー接続ハンドル。サーバー接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pSQLConnId*	SQL 接続を識別するための PVSQLCONNID 構造体のアドレス。SQL 接続は PvGetSQLConnectionsData() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_CLIENT	クライアント ID が無効です。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

例

```
BTI_SINT status = 0;
PVSQLCONNINFO connectionInfo;
PVSQLCONNID   connId;
status = PvGetSQLConnectionsData (connection, &count);
while (count > 0)
 {
     status = PvGetSQLConnectionInfo(connection, 0,
                  &connectionInfo);
      connId.u132ProcessId =
           connectionInfo.u132ProcessId;
      connId.u132ThreadId =
           connectionInfo.u132ThreadId;
      status = PvDisconnectSQLConnection(connection,
                  &connId);
      status = PvGetSQLConnectionsData (connection,
                  &count);
      }
     PvFreeSQLConnectionsData(connection, &count);
```

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetSQLConnectionsData()
PvGetSQLConnectionInfo()
PvDisconnect()
PvStop()

PvDropDatabase()

dnames.cfg から指定されたエントリを削除します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvDropDatabase(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR          option);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
In	*option*	オプションを示すビットマスクです。DDF ファイルを削除したい場合は、ファイル名に加えて下位ビットに 1（0001h）を設定します。そうしない場合、データベース名は削除されますが DDF ファイルは残ります。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvCreateDatabase()
PvModifyDatabase()
PvDisconnect()
PvStop()

PvDropGroup()

データベースから既存のグループを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvDropGroup(
   BTI_WORD            dbHandle,
   const BTI_CHAR*     group);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*group*	データベース グループ名。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたグループ名は存在しません。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。
PCM_errGroupNotEmpty	ユーザーをこのグループに関連付けます。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。

- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- グループは指定したデータベースに既に存在している。
- グループにはメンバーが含まれていない。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvCreateGroup()
PvAddUserToGroup()
PvRemoveUserFromGroup()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvDropIndex()

インデックス番号を指定することによって、辞書ファイルとデータ ファイルからインデックスを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvDropIndex(
   WORD             dictHandle,
   LPCSTR           tableName,
   WORD             indexNumber,
   BOOL             renumber);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	削除する、インデックスを持つテーブルの名前。
In	*indexNumber*	削除するインデックスの番号。
In	*renumber*	残りのインデックスの番号を付け替えるかどうかを示します。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。
PCM_errTableNotFound	指定されたテーブルが見つかりません。
PCM_errInvalidIndex	指定されたインデックスが見つかりません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableName によって指定されたテーブルが、*dictHandle* で指定された辞書の中になければなりません。

関連項目

PvStart()
PvOpenDatabase()
PvDropIndexByName()
PvAddIndex()
PvCloseDictionary()
PvStop()

PvDropIndexByName()

名前を指定することによって、辞書ファイルとデータ ファイルからインデックスを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvDropIndexByName(
   WORD            dictHandle,
   LPCSTR          tableName,
   LPCSTR          indexName);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	削除する、インデックスを持つテーブルの名前。
In	*indexName*	削除するインデックスの名前。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。
PCM_errTableNotFound	*tableName* で指定されたテーブルは辞書内に見つかりません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableName によって指定されたテーブルが、*dictHandle* で指定された辞書の中になければなりません。

関連項目

PvStart()
PvOpenDatabase()
PvAddIndex()
PvDropIndex()
PvCloseDictionary()
PvStop()

PvDropTable()

辞書ハンドルによって指定された開いている辞書から、指定されたテーブルを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvDropTable(
   WORD             dictHandle,
   LPCSTR           tableName,
   WORD             keepFile);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	削除するテーブルの名前。
In	*keepFile*	データ ファイルを削除するかどうかを示します。0 を設定すると、テーブルと関連付けられたデータ ファイルが削除されます。0 以外を設定すると、データ ファイルは削除されません。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。
PCM_errTableNotFound	指定されたテーブル名が見つかりません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableName によって指定されたテーブルが、*dictHandle* で指定された辞書の中になければなりません。

関連項目

PvStart()
PvOpenDatabase()
PvGetTableNames()
PvGetTable()
PvAddTable()
PvCloseDictionary()
PvStop()

PvDropUser()

データベースから既存のユーザーを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvDropUser(
   BTI_WORD           dbHandle,
   const BTI_CHAR*    user);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたアカウントまたはユーザー名は存在しません。
PCM_errNotAllowedToDropAdministrator	Master ユーザーを削除しようとしました。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。

- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- 同じ名前のユーザーが指定したデータベースに存在している。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvAddUserToGroup()
PvAlterUserName()
PvAlterUserPassword()
PvCreateUser()
PvRemoveUserFromGroup()
PvOpenDatabase()
PvCloseDatabase()

PvFreeDbNamesData()

接続されたサーバーのデータベース名に割り当てられたリソースを解放します。この関数を呼び出すには、それより前に PvGetDbNamesData が呼び出されている必要があります。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvFreeDbNamesData(
   BTI_LONG            hConnection);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	データベース名に関連するデータがありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetDbNamesData() 呼び出しによって、データベース名のデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetDbNamesData()
PvGetDbName()
PvDisconnect()
PvStop()

PvFreeMkdeClientsData()

アクティブな MicroKernel エンジン クライアントに関連するキャッシュ情報を解放します。この関数を呼び出すには、それより前に PvGetMkdeClientsData が呼び出されている必要があります。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvFreeMkdeClientsData(
   BTI_LONG           hConnection);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

- PvGetMkdeClientsData() 呼び出しによって、アクティブなクライアントのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeClientInfo()
PvDisconnect()
PvStop()

PvFreeOpenFilesData()

開いているファイルに関連するキャッシュ情報を解放します。この関数を呼び出すには、それより前に PvGetOpenFilesData が呼び出されている必要があります。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvFreeOpenFilesData(
   BTI_LONG           hConnection);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetOpenFilesData() 呼び出しによって、開いているファイルのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFilesData()
PvGetOpenFileName()
PvDisconnect()
PvStop()

PvFreeSQLConnectionsData()

SQL 接続に関連するキャッシュ情報を解放します。この関数を呼び出すには、それより前に PvGetSQLConnectionsData が呼び出されている必要があります。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvFreeSQLConnectionsData(
   BTI_LONG            hConnection);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetSQLConnectionsData() 呼び出しによって、開いているファイルのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetSQLConnectionsData()
PvGetSQLConnectionInfo()
PvDisconnect()
PvStop()

PvFreeTable()

PvGetTable() 関数呼び出しによって割り当てられたメモリを解放します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvFreeTable(
  TABLEINFO*      tableProps,
  COLUMNMAP*      columnList,
  INDEXMAP*       indexList);
```

引数

In/Out	*tableProps*	テーブル情報を含む構造体へのポインター。
In/Out	*columnList*	テーブルに定義された列の配列へのポインター。
In/Out	*indexList*	テーブルに定義されたセグメントの配列へのポインター。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	一般エラーが発生しました。

備考

この関数は、PvGetTable() で作成された構造体を解放します。

例

```
PRESULT status = 0;
status = PvFreeTable(mytableProps, MyColumnList, MyindexList);
```

関連項目

PvStart()
PvOpenDatabase()
PvGetTableNames()
PvGetTable()
PvFreeTableNames()
PvCloseDictionary()
PvStop()

PvFreeTableNames()

PvGetTableNames() 関数呼び出しによって割り当てられたメモリを解放します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvFreeTableNames(
   TABLEMAP*       tableList);
```

引数

In/Out	*tableList*	テーブル名を格納する TABLEMAP 構造体の配列。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。

備考

この関数で解放されるメモリは、PvGetTableNames() 呼び出しで、指定された辞書のテーブル名をすべて取得するために割り当てられたメモリです。

例

```
PRESULT status = 0;
status = PvFreeTableNames(&mytableList);
```

関連項目

PvStart()
PvOpenDatabase()
PvGetTableNames()

PvGetTable()
PvFreeTable()
PvCloseDictionary()
PvStop()

PvGetAllPossibleSelections()

選択タイプ（PVSETTING_SINGLE_SEL または PVSETTING_MULTI_SEL）の設定で使用可能な選択肢をすべて取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetAllPossibleSelections(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pNumItems,
   BTI_ULONG_PTR     pSelectionList);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pNumItems*	選択項目の総数を受け取る符号なし Long 型のアドレス。選択項目の数は、PvCountSelectionItems() を呼び出して取得することもできます。
Out	*pSelectionList*	使用可能な選択肢すべてを含む配列。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は選択タイプの設定ではありません。
P_E_BUFFER_TOO_SMALL	配列のサイズが小さすぎます。この場合、必要なサイズが *pNumItems* に返されます。

P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvCountSelectionItems()
PvDisconnect()
PvStop()

PvGetBooleanStrings()

ブール型の設定に関連する表示文字列を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetBooleanStrings(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_LONG_PTR      trueStringSize,
   BTI_CHAR_PTR      trueString,
   BTI_LONG_PTR      falseStringSize,
   BTI_CHAR_PTR      falseString);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*trueStringSize*	*trueString* の長さを格納する Long 型整数。
Out	*trueString*	True に対する表示文字列（サイズ >= 16 バイト）。
Out	*falseStringSize*	*falseString* の長さを格納する Long 型整数。
Out	*falseString*	False に対する表示文字列（サイズ >= 16 バイト）。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は Long 型の設定ではありません。

P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetBooleanValue()

ブール型の設定の値を取得します。デフォルト値または現在値のいずれかを取得できます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetBooleanValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_SINT_PTR      pValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pValue*	設定値を受け取るブール型変数のアドレス。
In	*whichData*	どちらの値を要求するかを示すフラグ。 PVDATA_DEFAULT はデフォルト値を返します。 PVDATA_CURRENT は現在値を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定はブール型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetBooleanStrings()
PvSetBooleanValue()
PvDisconnect()
PvStop()

PvGetCategoryInfo()

エンジン設定のカテゴリについての情報を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetCategoryInfo(
   BTI_LONG           hConnection,
   BTI_ULONG          categoryID,
   PVCATEGORYINFO*    pCatInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*categoryID*	カテゴリの一意な識別子。識別子の一覧は、PvGetCategoryList() 関数を使って取得できます。
Out	*pCatInfo*	カテゴリ情報を受け取る PVCATEGORYINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

PVCATEGORYINFO 構造体に返される設定の数は、そのカテゴリについての、クライアントとサーバー両方の設定の総数を表します。適用可能な設定の数を取得するには、PvGetSettingList() を呼び出します。リモート接続の場合、サーバー側の設定は適用できません。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvDisconnect()
PvStop()

PvGetCategoryList()

現在の接続によって特定されるエンジンの、カテゴリ ID の一覧を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetCategoryList(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pnumCategories,
   BTI_ULONG_PTR     pCategoriesList);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In/Out	*pnumCategories*	*pCategoriesList* で返すことができるカテゴリの数を格納する、符号なし Long 型のアドレス。また、PvGetCategoryListCount() を呼び出すと、この値を取得することができます。
Out	*pCategoriesList*	カテゴリ ID を格納する配列。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。
P_E_BUFFER_TOO_SMALL	配列のサイズが小さすぎます。必要なサイズが *pnumCategories* に返されます。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryInfo()
PvDisconnect()
PvStop()

PvGetCategoryListCount()

現在の接続によって特定されるエンジンの、カテゴリの数を取得します。この数は、後で PvGetCategoryList() に渡す配列を割り当てるのに使用できます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetCategoryListCount(
   BTI_LONG           hConnection,
   BTI_ULONG_PTR      pListCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pListCount*	カテゴリの数を格納する符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryInfo()
PvDisconnect()
PvStop()

PvGetDbCodePage()

名前付きデータベースに関連付けられたコード ページを取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbCodePage(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   dbName,
   BTI_LONG_PTR   pDbCodePage);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
Out	*pDbCodePage*	データベースのコード ページ。値がゼロの場合は、サーバー上のデフォルトのコード ページが使用されることを示します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvConnectServer()
PvCreateDatabase2()
PvCreateDSN2()
PvModifyDatabase2()
PvModifyDSN2()

PvGetDSNEx2()
PvStart()

PvGetDbDataPath()

名前付きデータベースのデータ パス（データ ファイルが保存されている場所）を取得します。この情報は dbnames.cfg に格納されています。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbDataPath(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       dbName,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       dataPath);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
In/Out	*pBufSize*	バッファーのサイズを格納する符号なし Long 型のアドレス。返されるパスの実際のサイズを受け取ります。
Out	*dataPath*	成功した場合はデータ パスが格納され、失敗した場合は空文字列が格納されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが *pBufSize* に返されます。
P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetDbNamesData()
PvGetDbName()
PvGetDbDictionaryPath()
PvGetDbServerName()
PvDisconnect()
PvStop()

PvGetDbDictionaryPath()

名前付きデータベースの辞書パス（DDF ファイルが保存されている場所）を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbDictionaryPath(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       dbName,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       dictPath);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
In/Out	*pBufSize*	バッファーのサイズを格納する符号なし Long 型のアドレス。返されるパスの実際のサイズを受け取ります。
Out	*dictPath*	成功した場合は辞書パスが格納され、失敗した場合は空文字列が格納されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが *pBufSize* に返されます。

P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetDbNamesData()
PvGetDbName()
PvGetDbDataPath()
PvGetDbServerName()
PvDisconnect()
PvStop()

PvGetDbFlags()

名前付きデータベースに関連付けられたデータベース フラグを取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbFlags(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       dbName,
   BTI_ULONG_PTR      pDbFlags);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
Out	*pDbFlags*	データベース フラグ。P_DBFLAG_ 定数を組み合わせて指定できます。 • P_DBFLAG_RI（参照整合性およびトリガーを含む整合性制約） • P_DBFLAG_BOUND（データベース名を DDF ファイルにスタンプし、そのデータベースのみが DDF を使用できるようにします。 • P_DBFLAG_DBSEC_AUTHENTICATION（混合セキュリティ ポリシー。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_DBSEC_AUTHORIZATION（データベース セキュリティ ポリシー。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_LONGMETADATA（メタデータのバージョンを参照してください。）

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

Btrieve セキュリティ ポリシー

次の表は、既存データベースのセキュリティ モデルを解釈する方法を示しています。

フラグの組み合わせ	相当するセキュリティ モデル
フラグなし	クラシック
P_DBFLAG_DBSEC_AUTHENTICATION	混合
P_DBFLAG_DBSEC_AUTHENTICATION + P_DBFLAG_DBSEC_AUTHORIZATION	データベース

関連項目

PvStart()
PvConnectServer()
PvCreateDatabase()
PvModifyDatabase()
PvGetDbNamesData()
PvGetDbName()
PvDisconnect()
PvStop()

PvGetDbName()

シーケンス番号を使って、接続されたサーバーのデータベース名を取得します。データベース名の番号は、PvGetDbNamesData() 関数を呼び出すことによって取得できます。シーケンス番号は 1 基準です。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbName(
   BTI_LONG          hConnection,
   BTI_ULONG         sequence,
   BTI_ULONG_PTR     pBufSize,
   BTI_CHAR_PTR      dbName);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*sequence*	データベース名のシーケンス番号（1 基準）。PvGetDbNamesData() によって定義される値を上限とした、有効な範囲内の番号でなければなりません。
In/Out	*pBufSize*	データベース名を取得するために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。コピーされた文字の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
Out	*dbName*	返された文字列値。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	データベース名に関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_BUFFER_TOO_SMALL	文字列に対し、割り当てられたバッファーが小さすぎます。
P_E_INVALID_SEQUENCE	シーケンス番号が無効です。
P_E_FAIL	その他の理由により失敗しました。

例

```
BTI_ULONG i;
BTI_ULONG count = 0;
BTI_CHAR dbName[BDB_SIZE_DBNAME+1];
BTI_SINT status = PvGetDbNamesData(connection, &count);
for (i=1; i<= count; i++)
{
   BTI_ULONG dbNameSize = sizeof(dbName);
   status = PvGetDbName(connection, i, &dbNameSize, dbName);
}
status = PvFreeDbNamesData(connection);
```

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetDbNamesData() 呼び出しによって、データベース名のデータが取得されている。
- 呼び出し元には、有効なデータベース名のシーケンス番号がある。

関連項目

PvStart()
PvConnectServer()
PvGetDbNamesData()
PvFreeDbNamesData()
PvDisconnect()
PvStop()

PvGetDbNamesData()

接続されたサーバーのデータベース名の数を取得します。名前を列挙するには、PvGetDbName() 関数を使用します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbNamesData(
   BTI_LONG           hConnection,
   BTI_ULONG_PTR      pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCount*	サーバー上のデータベース名の数を受け取るための符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

この関数は、データベース名の情報を取得するほかの関数よりも先に呼び出される必要があります。呼び出し元は PvFreeDbNamesData() を呼び出して、データベース名に割り当てられたリソースを解放する必要があります。

関連項目

PvStart()
PvConnectServer()
PvGetDbName()
PvFreeDbNamesData()
PvDisconnect()
PvStop()

PvGetDbServerName()

名前付きデータベースが存在するサーバーの名前を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetDbServerName(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_ULONG_PTR     pBufSize,
   BTI_CHAR_PTR      serverName,
   BTI_SINT_PTR      pIsLocal);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。特定サーバーのデータベース名の全リストは、PvGetDbNamesData() 関数を使って取得できます。結果リストから 1 つのデータベース名を取得するには、PvGetDbName() 関数を使用します。
In/Out	*pBufSize*	バッファーのサイズを格納する符号なし Long 型のアドレス。サーバー名の実際のサイズが返されます。
Out	*serverName*	成功した場合はサーバー名が格納され、失敗した場合は空文字列が格納されます。
Out	*pIsLocal*	リモート サーバーの場合は 0 を返し、ローカル サーバーの場合は 0 以外を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが *pBufSize* に返されます。
P_E_NOT_EXIST	名前付きデータベースは存在しません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetDbNamesData()
PvGetDbName()
PvDisconnect()
PvStop()

PvGetDSN()

データ ソース名（DSN）に関する情報を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvGetDSN(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dsnName,
   BTI_ULONG_PTR     pdsnDescSize,
   BTI_CHAR_PTR      dsnDesc,
   BTI_ULONG_PTR     pdsnDBQSize,
   BTI_CHAR_PTR      dsnDBQ);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dsnName*	データ ソース名。DSN の一覧は、PvListDSNs() 関数を使って取得できます。
In/Out	*pdsnDescSize*	DSN の説明のバッファーのサイズを格納する、符号なし Long 型のアドレス。DSN の説明の実際のサイズを受け取ります。
Out	*dsnDesc*	成功した場合は、DSN の説明が格納されます。
In/Out	*pdsnDBQSize*	データベース名用のバッファーのサイズを格納する、符号なし Long 型のアドレス。データベース名の実際のサイズを受け取ります。
Out	*dsnDBQ*	成功した場合は、データベースの名前が格納されます。

戻り値

P_OK	操作は成功しました。

P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが pdsnDescSize または pdsnDBQSize に返されます。
P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

ユーザーにログイン要求をしないで DSN に関する情報を取得するには、PvConnectServer() を使ってサーバー接続を確立するとき、userName と password に空文字列を渡します。

メモ： userName と password に空文字列を渡して確立した接続は、セキュリティで保護されていない接続であるため、DTI のこれ以外のほとんどの操作を実行するのに十分なアクセス権を持たない接続となります。

関連項目

PvStart()
PvConnectServer()
PvGetDSNEx()
PvListDSNs()
PvCountDSNs()
PvCreateDSN()
PvModifyDSN()
PvDeleteDSN()
PvDisconnect()
PvStop()

PvGetDSNEx()

データ ソース名（DSN）に関する情報を取得します。この関数は、DSN のオープン モードも取得されることを除けば、PvGetDSN() とまったく同じです。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvGetDSNEx(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dsnName,
   BTI_ULONG_PTR     pdsnDescSize,
   BTI_CHAR_PTR      dsnDesc,
   BTI_ULONG_PTR     pdsnDBQSize,
   BTI_CHAR_PTR      dsnDBQ,
   BTI_LONG_PTR      pOpenMode);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dsnName*	データ ソース名。DSN の一覧は、PvListDSNs() 関数を使って取得できます。
In/Out	*pdsnDescSize*	DSN の説明のバッファーのサイズを格納する、符号なし Long 型のアドレス。DSN の説明の実際のサイズを受け取ります。
Out	*dsnDesc*	成功した場合は、DSN の説明が格納されます。
In/Out	*pdsnDBQSize*	データベース名用のバッファーのサイズを格納する、符号なし Long 型のアドレス。データベース名の実際のサイズを受け取ります。
Out	*dsnDBQ*	成功した場合は、データベースの名前が格納されます。

Out	*pOpenMode*	DSN のオープン モードが格納されます。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE 『*ODBC Guide*』の DSN オープン モードも参照してください。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが pdsnDescSize または pdsnDBQSize に返されます。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_DSN_DOES_NOT_EXIST	指定された DSN は存在しません。
P_E_INVALID_OPEN_MODE	オープン モードが無効です。
P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

ユーザーにログイン要求をしないで DSN に関する情報を取得するには、PvConnectServer() を使ってサーバー接続を確立するとき、userName と password に空文字列を渡します。

メモ： userName と password に空文字列を渡して確立した接続は、セキュリティで保護されていない接続であるため、DTI のこれ以外のほとんどの操作を実行するのに十分なアクセス権を持たない接続となります。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvCountDSNs()
PvGetDSN()
PvCreateDSN()
PvModifyDSN()
PvDeleteDSN()
PvDisconnect()
PvStop()

PvGetDSNEx2()

データ ソース名（DSN）に関する情報を取得します。この関数は、データのエンコード オプションが取得されることを除けば、PvGetDSNEx() とまったく同じです。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvGetDSNEx2(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dsnName,
   BTI_ULONG_PTR     pdsnDescSize,
   BTI_CHAR_PTR      dsnDesc,
   BTI_ULONG_PTR     pdsnDBQSize,
   BTI_CHAR_PTR      dsnDBQ,
   BTI_LONG_PTR      pOpenMode,
   BTI_LONG_PTR      translate);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dsnName*	データ ソース名。DSN の一覧は、PvListDSNs() 関数を使って取得できます。
In/Out	*pdsnDescSize*	DSN の説明のバッファーのサイズを格納する、符号なし Long 型のアドレス。DSN の説明の実際のサイズを受け取ります。
Out	*dsnDesc*	成功した場合は、DSN の説明が格納されます。
In/Out	*pdsnDBQSize*	データベース名用のバッファーのサイズを格納する、符号なし Long 型のアドレス。データベース名の実際のサイズを受け取ります。
Out	*dsnDBQ*	成功した場合は、データベースの名前が格納されます。

Out	*pOpenMode*	DSN のオープン モード。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE 『*ODBC Guide*』の DSN オープン モードも参照してください。
Out	*translate*	データのエンコード オプション。次のいずれか 1 つになります。 • DSNFLAG_DEFAULT • DSNFLAG_OEMANSI • DSNFLAG_AUTO 『*ODBC Guide*』の DSN オープン モードも参照してください。DSNFLAG_DEFAULT は ODBC アドミニストレーターのエンコード オプション［なし］に該当します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが pdsnDescSize または pdsnDBQSize に返されます。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_DSN_DOES_NOT_EXIST	指定された DSN は存在しません。
P_E_INVALID_OPEN_MODE	オープン モードが無効です。
P_E_INVALID_TRANSLATE_OPTION	指定されたエンコード変換オプションが無効です。
P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

ユーザーにログイン要求をしないで DSN に関する情報を取得するには、PvConnectServer() を使ってサーバー接続を確立するとき、userName と password に空文字列を渡します。

メモ： userName と password に空文字列を渡して確立した接続は、セキュリティで保護されていない接続であるため、DTI のこれ以外のほとんどの操作を実行するのに十分なアクセス権を持たない接続となります。

関連項目

PvConnectServer()
PvCountDSNs()
PvCreateDSN2()
PvDeleteDSN()
PvDisconnect()
PvGetDSNEx()
PvListDSNs()
PvModifyDSN2()
PvStart()
PvStop()

PvGetEngineInformation()

指定された *hConnection* のデータベース エンジンに関する情報を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetEngineInformation(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       pserverClient,
   BTI_ULONG_PTR      pdbuApiVer,
   BTI_ULONG_PTR      pmajor,
   BTI_ULONG_PTR      pminor,
   BTI_ULONG_PTR      pserverClientType);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pserverClient*	BTI_CHAR_PTR のアドレス。 True - MKDE_SERVR_ENGINE_CID False - MKDE_CLNT_ENGINE_CID
Out	*pdbuApiVer*	構造体のバージョン。ヌルでもかまいません。
Out	*pmajor*	メジャー バージョン - ヌルでもかまいません。
Out	*pminor*	マイナー バージョン - ヌルでもかまいません。
Out	*pserverClientType*	MKDE_SRVR_ENGINE_CID のみ 次のいずれか 1 つが返されます。 UNKNOWN_ENGINE_CLIENT（0） NT_SERVER（1） WIN32_CLIENT（3） UNIX_SERVER（4） CLIENT_CACHE（5） VXWIN_SERVER（6） VXLINUX_SERVER（7） REPORT_ENGINE（9）

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvDisconnect()
PvStop()

PvGetError()

直前のエラーを説明する、エラー説明文字列を返します。この関数の対象となるのは、カタログ関数で発生したエラーのみです。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvGetError(
   LPSTR           errorDesc,
   WORD*           size);
```

引数

In/Out	*errorDesc*	エラー説明を格納する文字列。
In/Out	*size*	*errorDesc* のサイズ。エラー説明を格納するのに十分なサイズでない場合は、エラーが返され、必要なサイズが *size* に格納されます。

戻り値

PCM_Success	操作は成功しました。
PCM_errStringTooShort	*size* パラメーターは、エラー説明を格納するのに十分なサイズではありませんでした。必要なサイズが *size* に返されます。

備考

errorDesc 文字列の割り当ては呼び出し元が行います。

エラー説明の最大サイズは、ヘッダー ファイル ddf.h に記述されている定数 `ERROR_LEN` で指定されます。

関連項目

PvStart()
PvStop()

PvGetFileHandlesData()

開いているファイルに関連するファイル ハンドルの情報をすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetFileHandlesData(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      fileName,
   BTI_ULONG_PTR     pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*fileName*	照会するファイルの絶対パス名。
Out	*pCount*	開いているファイルのハンドル数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FILE_NOT_OPEN	指定されたファイルは現在開いていません。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

情報は、ファイル ハンドルに関連する以降の呼び出しのために、DTI によってキャッシュされます。この関数は、開いているファイルに対し、ファイル ハンドル情報を取得するほかの関数よりも先に呼び出される必要があります。PvFreeOpenFilesData() を呼び出すと、ファイル ハンドルのキャッシュ情報が解放されます。

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetOpenFilesData() 呼び出しによって、開いているファイルのデータが取得されている。
- 呼び出し元には既に、開いているファイルの有効なファイル名がある。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFilesData()
PvFreeOpenFilesData()
PvDisconnect()
PvStop()

PvGetFileHandleInfo()

開いているファイルに関連付けられたファイル ハンドルの情報を照会します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetFileHandleInfo(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       fileName,
   BTI_ULONG          sequence,
 PVFILEHDLINFO*       pFileHdlInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*fileName*	照会するファイルの絶対パス名。
In	*sequence*	ファイル ハンドルのシーケンス番号（0 基準）。PvGetFileHandlesData() によって取得されるファイル ハンドル数を上限とした、有効な範囲内の番号でなければなりません。
Out	*pFileHdlInfo*	ファイル ハンドルの情報を受け取る PVFILEHDLINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_SEQUENCE	シーケンス番号が無効です。
P_E_FILE_NOT_OPEN	指定されたファイルは現在開いていません。

P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetOpenFilesData() 呼び出しによって、開いているファイルのデータが取得されている。
- PvGetFileHandlesData() 呼び出しによって、開いているファイル ハンドルのデータが取得されている。
- 呼び出し元には既に、開いているファイルの有効なファイル名がある。
- 呼び出し元には既に、有効なファイル ハンドルのシーケンスがある。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFilesData()
PvGetFileHandlesData()
PvGetOpenFileName()
PvFreeOpenFilesData()
PvDisconnect()
PvStop()

PvGetFileInfo()

開いているファイルの情報を照会します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetFileInfo(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      fileName,
   PVFILEINFO*       pFileInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*fileName*	照会するファイルの絶対パス名。
Out	*pFileInfo*	ファイルの情報を受け取る PVFILEINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FILE_NOT_OPEN	指定されたファイルは現在開いていません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetOpenFilesData() 呼び出しによって、開いているファイルのデータが取得されている。
- 呼び出し元には既に、開いているファイルの有効なファイル名がある。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFilesData()
PvDisconnect()
PvStop()

PvGetLongValue()

whichData によって指定されるデータ ソースから、Long 型整数の設定の値を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetLongValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_LONG_PTR      pValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pValue*	設定値を受け取る Long 型整数変数のアドレス。
In	*whichData*	どちらの値を要求するかを示すフラグ。 PVDATA_DEFAULT はデフォルト値を返します。 PVDATA_CURRENT は現在値を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は Long 型整数の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

設定で許容される最小値および最大値を取得するには、PvGetValueLimit() 関数を使用します。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetValueLimit()
PvDisconnect()
PvStop()

PvGetMkdeClientId()

アクティブな MicroKernel エンジン クライアントのクライアント ID を取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeClientId(
    BTI_LONG           hConnection,
    BTI_ULONG          sequence,
  PVCLIENTID*          pClientId);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*sequence*	MicroKernel エンジン クライアントのシーケンス番号（0 基準）。PvGetMkdeClientsData() によって返される値を上限とした、有効な範囲内の番号でなければなりません。
Out	*pClientId*	返されたクライアント ID 情報を格納する PVCLIENTID 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_SEQUENCE	シーケンス番号が無効です。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetMkdeClientsData() 呼び出しによって、アクティブなクライアントのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeClientInfo()
PvFreeMkdeClientsData()
PvDisconnect()
PvStop()

PvGetMkdeClientInfo()

アクティブな MicroKernel エンジン クライアントの情報を照会します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeClientInfo(
   BTI_LONG            hConnection,
  PVCLIENTID*          pClientId,
  PVMKDECLIENTINFO*    pClientInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pClientId*	MicroKernel エンジン クライアントを識別する PVCLIENTID 構造体のアドレス。
Out	*PClientInfo*	MicroKernel エンジン クライアントの情報を受け取る PVMKDECLIENTINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_CLIENT	クライアント ID が無効です。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetMkdeClientsData() 呼び出しによって、アクティブなクライアントのデータが取得されている。
- 呼び出し元には既に、アクティブな MicroKernel エンジン クライアントの有効なクライアント ID がある。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeClientId()
PvFreeMkdeClientsData()
PvDisconnect()
PvStop()

PvGetMkdeClientHandlesData()

アクティブなクライアントに関連する MicroKernel エンジン クライアント ハンドルの数を取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeClientHandlesData(
   BTI_LONG           hConnection,
  PVCLIENTID*         pClientId,
   BTI_ULONG_PTR      pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pClientId*	MicroKernel エンジン クライアントを識別する PVCLIENTID 構造体のアドレス。
Out	*pCount*	MicroKernel エンジン クライアントのハンドル数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	MicroKernel エンジン クライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

この関数を呼び出すと、MicroKernel エンジン クライアント ハンドルに関するすべての情報が、それ以降のクライアント ハンドル関連の関数呼び出しのために、DTI によってキャッシュされます。クライアントに関するその他の情報については、PvGetMkdeClientsData() を参照してください。

この関数は、クライアント ハンドルの情報を返すほかの関数よりも先に呼び出される必要があります。

PvFreeMkdeClientsData() を呼び出すと、MicroKernel エンジン クライアント ハンドル用にキャッシュされた情報がクライアントに関する情報と一緒に解放されます。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

- PvGetMkdeClientsData() 呼び出しによって、アクティブなクライアントのデータが取得されている。
- 呼び出し元には既に、アクティブな MicroKernel エンジン クライアントの有効なクライアント ID がある。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvFreeMkdeClientsData()
PvDisconnect()
PvStop()

PvGetMkdeClientHandleInfo()

アクティブなクライアントに関連付けられた MicroKernel エンジン クライアント ハンドルの情報を照会します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeClientHandleInfo(
    BTI_LONG               hConnection,
  PVCLIENTID*              pClientId,
    BTI_ULONG              sequence,
  PVMKDECLIENTHDLINFO*     pClientHdlInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pClientId*	MicroKernel エンジン クライアントを識別する PVCLIENTID 構造体のアドレス。
In	*sequence*	クライアント ハンドルのシーケンス番号（0 基準）。PvGetMkdeClientHandlesData() によって取得されるハンドル数を上限とした、有効な範囲内の番号でなければなりません。
Out	*pClientHdlInfo*	クライアント ハンドルの情報を受け取る PVMKDECLIENTHDLINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_INVALID_CLIENT	クライアント ID が無効です。
P_E_INVALID SEQUENCE	シーケンス番号が無効です。

P_E_FAIL	名前付きサーバーからの切断に失敗しました。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetMkdeClientsData() 呼び出しによって、アクティブな MicroKernel エンジン クライアントのデータが取得されている。
- PvGetMkdeClientHandlesData() 呼び出しによって、MicroKernel エンジン クライアント ハンドルのデータが取得されている。
- 呼び出し元には既に、アクティブな MicroKernel エンジン クライアントの有効なクライアント ID がある。
- 呼び出し元には既に、アクティブな MicroKernel エンジン クライアントの有効なハンドル シーケンスがある。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeClientsData()
PvGetMkdeClientHandlesData()
PvFreeMkdeClientsData()
PvDisconnect()
PvStop()

PvGetMkdeClientsData()

アクティブな MicroKernel エンジン クライアントに関連する情報をすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeClientsData(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCount*	アクティブな MicroKernel エンジン クライアントの数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

この関数を呼び出すと、MicroKernel エンジン クライアントに関するすべての情報が、それ以降のクライアント関連の関数呼び出しのために、DTI によってキャッシュされます。1 つの例外はクライアント ハンドルに関する情報で、この情報は類似する関数 PvGetMkdeClientHandlesData() を使ってキャッシュされます。

この関数は、クライアントの情報を返すほかの関数よりも先に呼び出される必要があります。キャッシュ情報が必要でなくなったら、呼び出し元は PvFreeMkdeClientsData() を呼び出してキャッシュ情報を解放する。

この関数を呼び出して、キャッシュ情報をリフレッシュすることもできます。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvFreeMkdeClientsData()
PvGetMkdeClientHandlesData()
PvDisconnect()
PvStop()

PvGetMkdeCommStat()

MicroKernel エンジンの通信統計データをすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeCommStat(
    BTI_LONG            hConnection,
  PVCOMMSTAT*           pCommStat);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCommStat*	MicroKernel エンジン通信統計情報を受け取る PVCOMMSTAT 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_COMPONENT_NOT_LOADED	コンポーネントがロードされていません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

- PvGetSQLConnectionsData() 呼び出しによって、開いているファイルのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetSQLConnectionsData()
PvGetMkdeUsage()
PvFreeSQLConnectionsData()
PvDisconnect()
PvStop()

PvGetMkdeCommStatEx()

MicroKernel エンジンの通信統計データをすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeCommStatEx(
    BTI_LONG              hConnection,
  PVCOMMSTATEX*           pCommStatEx);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCommStatEx*	MicroKernel エンジン通信統計情報を受け取る PVCOMMSTATEX 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_COMPONENT_NOT_LOADED	コンポーネントがロードされていません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

この関数は PvGetMkdeCommStat と同じデータを返しますが、2 つの追加要素を持つ新しい構造体 PVCOMMSTATEX を使用します。追加される要素（totalTimeouts と totalRecoveries）は、自動再接続機能に関するものです。自動再接続の詳細については、『*Advanced Operations Guide*』を参照してください。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetSQLConnectionsData() 呼び出しによって、開いているファイルのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetSQLConnectionsData()
PvGetMkdeUsage()
PvFreeSQLConnectionsData()
PvDisconnect()
PvStop()

PvGetMkdeUsage()

MicroKernel エンジンからリソース使用状況の情報を取得します。この情報には、ライセンス数、ファイル数、ハンドル数、トランザクション数、クライアント数、スレッド数、およびロック数の現在値、ピーク値、最大値が含まれます。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeUsage(
    BTI_LONG            hConnection,
  PVMKDEUSAGE*          pMkdeUsage);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pMkdeUsage*	MicroKernel エンジン リソース使用状況の情報を受け取る PVMKDEUSAGE 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeCommStat()
PvGetMkdeUsageEx()
PvDisconnect()
PvStop()

PvGetMkdeUsageEx()

MicroKernel エンジン データベース エンジンからリソース使用状況の情報を取得します。この情報には、ユーザー数、セッション数、使用中データ、ファイル数、ハンドル数、トランザクション数、クライアント数、スレッド数、およびロック数の現在値、ピーク値、最大値のほか、データベース エンジンが実行されている継続時間（秒）（[エンジン稼働時間] といいます）が含まれます。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeUsageEx(
   BTI_LONG             hConnection,
  PVMKDEUSAGEEX*        pMkdeUsage);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pMkdeUsage*	MicroKernel エンジン リソース使用状況の情報を受け取る PVMKDEUSAGEEX 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

この PvGetMkdeUsageEx() 関数は PvGetMkdeUsage() と類似しています。構造体のみが異なります。同じ要素を提供する場合、PVMKDEUSAGE では 2 バイトの要素を提供しますが、PVMKDEUSAGEEX では 4 バイトの要素を提供します。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeCommStat()
PvGetMkdeUsage()
PvDisconnect()
PvStop()

PvGetMkdeVersion()

MicroKernel エンジン バージョン情報を取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetMkdeVersion(
   BTI_LONG          hConnection,
  PVVERSION*         pMkdeVersion);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pMkdeVersion*	MicroKernel エンジン バージョン情報を受け取る PVVERSION 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_COMPONENT_NOT_LOADED	コンポーネントがロードされていません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeCommStat()
PvGetMkdeUsageEx()
PvDisconnect()
PvStop()

PvGetOpenFilesData()

開いているファイルに関連する情報をすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetOpenFilesData(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCount*	開いているファイルの数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

情報は、開いているファイルに関連する以降の呼び出しのために、DTI によってキャッシュされます。この関数は、開いているファイルの情報を取得するほかの関数よりも先に呼び出される必要があります。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

次の事後条件を満たす必要があります。

- キャッシュ情報が必要でなくなったら、呼び出し元は PvFreeOpenFilesData() を呼び出してキャッシュ情報を解放する。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFileName()
PvFreeOpenFilesData()
PvDisconnect()
PvStop()

PvGetOpenFileName()

開いているファイルの絶対パス名を取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetOpenFileName(
   BTI_LONG           hConnection,
   BTI_ULONG          sequence,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       fileName);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*sequence*	ファイルのシーケンス番号（0 基準）。PvGetOpenFilesData() によって返される値を上限とした、有効な範囲内の番号でなければなりません。
In/Out	*pBufSize*	ファイル名を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。コピーされた文字の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
In/Out	*fileName*	返された文字列値。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_DATA_UNAVAILABLE	アクティブなクライアントに関連するデータがありません。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_BUFFER_TOO_SMALL	文字列に対し、割り当てられたバッファーが小さすぎます。返された文字列は切り詰められます。この場合、必要なサイズが pBufSize に返されます。
P_E_INVALID_SEQUENCE	シーケンス番号が無効です。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetOpenFilesData() 呼び出しによって、開いているファイルのデータが取得されている。

関連項目

PvStart()
PvConnectServer()
PvGetOpenFilesData()
PvFreeOpenFilesData()
PvDisconnect()
PvStop()

PvGetProductsInfo()

License Manager で検出されたすべての Zen 製品についての情報を持つ xml 文字列を取得します。

ヘッダー ファイル：dtilicense.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvGetProductsInfo(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      productInfo,
   BTI_ULONG_PTR     pBufSize);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*productInfo*	返された製品情報を持つ XML 文字列。
In/Out	*pBufSize*	文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。選択文字列の実際の長さを受け取ります。

戻り値

DBU_SUCCESS	操作は成功しました。
P_E_FAIL	その他の理由により失敗しました。
ライセンス管理または認証に関するステータス コード	License Administrator のステータス コードおよび許可のステータス コードは、『*Status Codes and Messages*』を参照してください。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

PvGetProductsInfo() によって返される製品情報

以下に、`PvGetProductsInfo()` によって返される XML 文字列のドキュメント型定義（DTD）とその用語について説明します。

```
<!DOCTYPE products [
<!ELEMENT products (product*)>
<!ELEMENT product (name,id,licenses)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT licenses (license*)>
<!ELEMENT license
(type,productCode*,productKey*,state*,feature*,edition*,maxUserCount*,maxSessionCount*,maxDataInUseG
B*,platform*,sequence*,userCount*,sessionCount*,dataInUseGB*,timeStamp*,oemId*,application*,descript
ion*,isremovable*,gracePeriodEnd*)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT productCode (#PCDATA)>
<!ELEMENT productKey (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT feature (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT maxUserCount (#PCDATA)>
<!ELEMENT maxSessionCount (#PCDATA)>
<!ELEMENT maxDataInUseGB (#PCDATA)>
<!ELEMENT platform (#PCDATA)>
<!ELEMENT sequence (#PCDATA)>
<!ELEMENT userCount (#PCDATA)>
<!ELEMENT sessionCount (#PCDATA)>
<!ELEMENT dataInUseGB (#PCDATA)>
<!ELEMENT timeStamp (#PCDATA)>
<!ELEMENT oemId (#PCDATA)>
<!ELEMENT application (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT isremovable (#PCDATA)>
<!ELEMENT gracePeriodEnd (#PCDATA)>
]>
```

products	PvGetProductsInfo() によって返される全製品のコンテナー。
product	単独製品に関する情報のコンテナー。
name	製品の名前。
id	Zen 製品コード。dtilicense ヘッダー ファイルで、返される製品コードの一覧を参照してください。
licenses	製品に適用される全ライセンスのコンテナー。

license	単独ライセンスに関する情報のコンテナー。
type	ライセンスの種類。 1：期限なし 2：発行時に適用される期限付きライセンス 4：インストール時に適用される期限付きライセンス 7：追加ユーザー
productCode	Zen 製品コード。dtilicense ヘッダー ファイルで、返される製品コードの一覧を参照してください。
productKey	製品認証に使用されるキー。製品認証が使用されなかった場合は空になります。
state	現在のライセンスの状態。 0：アクティブ 1：期限切れ 2：無効 3：非アクティブ 4：検証失敗
feature	予約済み
edition	予約済み
maxUserCount	許可される最大同時使用ユーザー数。Zen Enterprise Server および Workgroup の場合、0（ゼロ）はユーザー数が無制限であることを示します。Zen Cloud Server の場合は適用されず、常に "0" が返ります。
maxSessionCount	許可される最大同時セッション数。Zen Cloud Server の場合、0（ゼロ）はセッションが無制限であることを示します。Zen Enterprise Server および Workgroup の場合は適用されず、常に "0" が返ります。
maxDatainUseGB	同時に使用できる最大データ量（ギガバイト単位）。Zen Cloud Server の場合、0（ゼロ）はデータ量が無制限であることを示します。Zen Enterprise Server および Workgroup の場合は適用されず、常に "0" が返ります。

platform	対応しているプラットフォーム。 0：ANY 1：WIN 2：WIN32 3：WIN64 4：LINUX 5：LINUX32 6：LINUX64
sequence	ライセンス シーケンス番号。
userCount	ライセンスによって許可されるユーザー数。Zen Enterprise Server および Workgroup の場合、-1 はユーザー数が無制限であることを示します。Zen Cloud Server の場合は適用されず、常に "0" が返ります。
sessionCount	ライセンスによって許可されるセッション数。Zen Cloud Server の場合、-1 はユーザー数が無制限であることを示します。Zen Enterprise Server および Workgroup の場合は適用されず、常に "0" が返ります。
dataInUseGB	ライセンスによって許可される使用データ量（ギガバイト単位）。Zen Cloud Server の場合、-1 は使用データ サイズが無制限であることを示します。Zen Enterprise Server および Workgroup の場合は適用されず、常に "0" が返ります。
timeStamp	一時キーの場合、失効日は 2000 年 1 月 1 日からの日数で表します。
oemId	ベンダー ID。
application	ベンダーのアプリケーション ID。
description	予約済み
isremovable	ライセンス キーは削除可能です。 0：削除できません。 1：削除できます。
gracePeriodEnd	ラインセンスの検証失敗のためエンジンが無効になるまでの日数。検証失敗期間がこの製品に適用されない場合は空。検証失敗期間はこの製品に対し適用可能だが有効になっていない場合は -1。

例

```
<?xml version="1.0" encoding='UCS-4' ?>
<!DOCTYPE products [
<!ELEMENT products (product*)>
```

```
<!ELEMENT product (name,id,licenses)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT licenses (license*)>
<!ELEMENT license
(type,productCode*,productKey*,state*,feature*,edition*,maxUserCount*,maxSessionCount*,maxDataInUseG
B*,platform*,sequence*,userCount*,sessionCount*,dataInUseGB*,timeStamp*,oemId*,application*,descript
ion*,isremovable*,gracePeriodEnd*)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT productCode (#PCDATA)>
<!ELEMENT productKey (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT feature (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT maxUserCount (#PCDATA)>
<!ELEMENT maxSessionCount (#PCDATA)>
<!ELEMENT maxDataInUseGB (#PCDATA)>
<!ELEMENT platform (#PCDATA)>
<!ELEMENT sequence (#PCDATA)>
<!ELEMENT userCount (#PCDATA)>
<!ELEMENT sessionCount (#PCDATA)>
<!ELEMENT dataInUseGB (#PCDATA)>
<!ELEMENT timeStamp (#PCDATA)>
<!ELEMENT oemId (#PCDATA)>
<!ELEMENT application (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT isremovable (#PCDATA)>
<!ELEMENT gracePeriodEnd (#PCDATA)>
]>
<products>
  <product>
    <name>DataExchange 5 Server: Real-Time Backup</name>
    <id>78</id>
    <licenses>
      <license>
        <type>1</type>
        <productCode>78</productCode>
        <productKey> ABCDE-55555-FGHIJ-55555-KLMNO-55555</productKey>
        <state>0</state>
        <feature>0</feature>
        <edition>0</edition>
        <maxUserCount>0</maxUserCount>
        <maxSessionCount>0</maxSessionCount>
        <maxDataInUseGB>0</maxDataInUseGB>
        <platform>2</platform>
        <sequence>0</sequence>
        <userCount>1</userCount>
        <sessionCount>0</sessionCount>
        <dataInUseGB>0</dataInUseGB>
        <timeStamp>0</timeStamp>
        <oemId>0</oemId>
        <application>0</application>
        <description></description>
        <isremovable>1</isremovable>
        <gracePeriodEnd>-1</gracePeriodEnd>
      </license>
    </licenses>
  </product>
  <product>
    <name>PSQL 12 Server</name>
    <id>425</id>
    <licenses>
      <license>
        <type>2</type>
        <productCode>425</productCode>
```

```
      <productKey></productKey>
      <state>0</state>
      <feature>0</feature>
      <edition>0</edition>
      <maxUserCount>0</maxUserCount>
      <maxSessionCount>0</maxSessionCount>
      <maxDataInUseGB>0</maxDataInUseGB>
      <platform>2</platform>
      <sequence>0</sequence>
      <userCount>10</userCount>
      <sessionCount>0</sessionCount>
      <dataInUseGB>0</dataInUseGB>
      <timeStamp>4489</timeStamp>
      <oemId>8</oemId>
      <application>604</application>
      <description></description>
      <isremovable>0</isremovable>
      <gracePeriodEnd></gracePeriodEnd>
    </license>
    <license>
      <type>4</type>
      <productCode>425</productCode>
      <productKey></productKey>
      <state>0</state>
      <feature>0</feature>
      <edition>0</edition>
      <maxUserCount>0</maxUserCount>
      <maxSessionCount>0</maxSessionCount>
      <maxDataInUseGB>0</maxDataInUseGB>
      <platform>1</platform>
      <sequence>11200</sequence>
      <userCount>20</userCount>
      <sessionCount>0</sessionCount>
      <dataInUseGB>0</dataInUseGB>
      <timeStamp>4429</timeStamp>
      <oemId>0</oemId>
      <application>1</application>
     <description></description>
      <isremovable>0</isremovable>
      <gracePeriodEnd></gracePeriodEnd>
    </license>
    <license>
      <type>1</type>
      <productCode>425</productCode>
      <productKey>ABCDE-55555-FGHIJ-55555-KLMNO-55555</productKey>
      <state>0</state>
      <feature>0</feature>
      <edition>0</edition>
      <maxUserCount>0</maxUserCount>
      <maxSessionCount>0</maxSessionCount>
      <maxDataInUseGB>0</maxDataInUseGB>
      <platform>2</platform>
      <sequence>0</sequence>
      <userCount>10</userCount>
      <sessionCount>0</sessionCount>
      <dataInUseGB>0</dataInUseGB>
      <timeStamp>0</timeStamp>
      <oemId>333</oemId>
     <application>334</application>
      <description></description>
      <isremovable>1</isremovable>
      <gracePeriodEnd>-1</gracePeriodEnd>
    </license>
  </licenses>
</product>
```

```
</products>
```

関連項目

PvValidateLicenses()
PvConnectServer()
PvStart()
PvStop()

PvGetSelectionString()

選択タイプの設定の、特定の選択肢に対する表示文字列を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSelectionString(
   BTI_LONG           hConnection,
   BTI_ULONG          settingID,
   BTI_ULONG          selection,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       dispString);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In	*selection*	選択肢のインデックス。PvGetAllPossibleSelections() から返された PSelectionList。
In/Out	*pBufSize*	文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。選択文字列の実際の長さを受け取ります。
Out	*dispString*	返された表示文字列。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は選択タイプの設定ではありません。

P_E_BUFFER_TOO_SMALL	配列のサイズが小さすぎます。この場合、必要なサイズが *pBufSize* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetSelectionStringSize()
PvDisconnect()
PvStop()

PvGetSelectionStringSize()

PvGetSelectionString() 呼び出しが成功するために必要なバッファーのサイズを取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSelectionStringSize(
   BTI_LONG           hConnection,
   BTI_ULONG          settingID,
   BTI_ULONG_PTR      pBufSize);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	PvGetSelectionString() 呼び出しで文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。選択文字列の実際の長さを受け取ります。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は選択タイプの設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetSelectionValue()

whichData によって指定されるデータ ソースから、選択タイプの設定の値を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSelectionValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pNumItems,
   BTI_LONG_PTR      pValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pNumItems*	入力時は配列サイズを指定し、戻り時は個々の選択項目の数を受け取る、符号なし Long 型のアドレス。
Out	*pValue*	個々の選択インデックスの配列。
In	*whichData*	どちらの値を要求するかを示すフラグ。 PVDATA_DEFAULT はデフォルト値を返します。 PVDATA_CURRENT は現在値を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_INVALID_DATA_TYPE	要求された設定は文字列型の設定ではありません。
P_E_BUFFER_TOO_SMALL	配列のサイズが小さすぎます。この場合、必要なサイズが *pNumItems* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetServerName()

接続ハンドルによって示される接続中のサーバー名を取得します。

ヘッダー ファイル：connect.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetServerName(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pBufSize,
   BTI_CHAR_PTR      serverName);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In/Out	*pBufSize*	サーバー名を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。
In/Out	*serverName*	成功した場合はサーバー名が返され、失敗した場合は空文字列が返されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが pBufSize に返されます。
P_E_FAIL	名前付きサーバーとの接続に失敗しました。

備考

実行が最初に呼び出された際には、必要な初期化を行う必要があります。

同時に複数の接続が可能です。

関連項目

PvStart()
PvConnectServer()
PvDisconnect()
PvStop()

PvGetSettingHelp()

設定に関連するヘルプ文字列を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingHelp(
   BTI_ULONG          settingID,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       pHelpString);
```

引数

In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	設定値を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。設定値の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
Out	*pHelpString*	返された文字列値。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	割り当てられたバッファーが小さすぎるので、表示文字列は切り詰められます。この場合、必要なバッファー サイズが *pBufSize* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetSettingInfo()
PvDisconnect()
PvStop()

PvGetSettingHelpSize()

設定に関連するヘルプ文字列を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingHelpSize(
   BTI_ULONG          settingID,
   BTI_ULONG_PTR      pBufSize);
```

引数

In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	設定値を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。設定値の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetSettingInfo()
PvDisconnect()
PvStop()

PvGetSettingInfo()

設定の設定情報を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingInfo(
    BTI_LONG            hConnection,
    BTI_ULONG           settingID,
  PVSETTINGINFO*        pSettingInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pSettingInfo*	設定情報を受け取る PVSETTINGINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetSettingHelp()
PvDisconnect()
PvStop()

PvGetSettingList()

指定されたカテゴリに属する設定の一覧を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingList(
   BTI_LONG           hConnection,
   BTI_ULONG          categoryID,
   BTI_ULONG_PTR      pNumSettings,
   BTI_ULONG_PTR      pSettingList);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*categoryID*	カテゴリの一意な識別子。
Out	*pNumSettings*	入力時は配列のサイズを格納し、戻り時は返された一覧の項目数を受け取る、符号なし Long 型のアドレス。
Out	*pSettingList*	返される設定 ID の一覧へのポインター。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_BUFFER_TOO_SMALL	配列のサイズが小さすぎます。この場合、必要なサイズが *pNumSettings* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

接続がリモート接続の場合、カテゴリのサーバー側の設定のみが返されます。接続がローカル接続の場合、このカテゴリのクライアント側とサーバー側両方の設定が返されます。

設定項目に対する設定が現時点で可能であるかどうかを判断するには、PvIsSettingAvailable() を使用します。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvIsSettingAvailable()
PvGetSettingHelp()
PvGetSettingInfo()
PvGetSettingMap()
PvGetSettingUnits()
PvDisconnect()
PvStop()

PvGetSettingListCount()

指定されたカテゴリに属する設定の数を取得します。この数は、後で PvGetSettingList() に渡す配列を割り当てるのに使用できます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingListCount(
   BTI_LONG           hConnection,
   BTI_ULONG          categoryID,
   BTI_ULONG_PTR      pNumSettings);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*categoryID*	カテゴリの一意な識別子。
Out	*pNumSettings*	入力時は配列のサイズを格納し、戻り時は返された一覧の項目数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

接続がリモート接続の場合、カテゴリのサーバー側の設定のみが返されます。接続がローカル接続の場合、このカテゴリのクライアント側とサーバー側両方の設定が返されます。

設定項目に対する設定が現時点で可能であるかどうかを判断するには、PvIsSettingAvailable() を使用します。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvIsSettingAvailable()
PvGetSettingHelp()
PvGetSettingInfo()
PvGetSettingMap()
PvGetSettingUnits()
PvDisconnect()
PvStop()

PvGetSettingMap()

設定のオプション ID とコンポーネント ID を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingMap(
   BTI_ULONG          settingID,
   BTI_WORD_PTR       pComponentID,
   BTI_WORD_PTR       pOptionID);
```

引数

In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pComponentID*	コンポーネント用の符号なし Short 型のアドレス。
Out	*pOptionID*	オプション用の符号なし Short 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

オプションおよびコンポーネントは、設定を DBUGetInfo または DBUSetInfo 呼び出しにマップします。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetSettingUnits()

デフォルト単位と推奨ファクターを取得します。この関数は、Long 型整数の設定にのみ有効です。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingUnits(
   BTI_LONG           hConnection,
   BTI_ULONG          settingID,
   BTI_ULONG_PTR      pBufSize,
   BTI_CHAR_PTR       pValue,
   BTI_ULONG_PTR      pFactor,
   BTI_ULONG_PTR      pFBufSize,
   BTI_CHAR_PTR       pFValue);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	デフォルト単位の文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。デフォルト単位の文字列の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
Out	*pValue*	返されたデフォルト値の文字列。
Out	*pFactor*	ファクター用の符号なし Long 型のアドレス。
In/Out	*pFBufSize*	「ファクター」単位の文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。デフォルト単位の文字列の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
Out	*pFValue*	返されたファクター値の文字列。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は Long 型整数の設定ではありません。
P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが *pBufSize* に返されます。
P_E_FAIL	名前付きサーバーとの接続に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetSettingUnitsSize()

PvGetSettingUnits() 呼び出しで情報を取得するために必要なバッファーのサイズをバイト数で返します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSettingUnitsSize(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pBufSize,
   BTI_ULONG_PTR     pFBufSize);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	デフォルト単位の文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。デフォルト単位の文字列の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。
In/Out	*pFBufSize*	「ファクター」単位の文字列を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。デフォルト単位の文字列の実際のサイズを受け取ります。サイズにはヌル終端文字を含める必要があります。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は Long 型整数の設定ではありません。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvGetSQLConnectionsData()

SQL 接続マネージャーへの接続数と、その接続に関連する情報をすべて取得します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSQLConnectionsData(
   BTI_LONG           hConnection,
   BTI_ULONG_PTR      pCount);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
Out	*pCount*	SQL 接続の数を受け取る、符号なし Long 型のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_FAIL	その他の理由により失敗しました。

備考

情報は、SQL 接続に関連する以降の呼び出しのために、DTI によってキャッシュされます。この関数は、SQL 接続の情報を取得するほかの関数よりも先に呼び出される必要があります。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

次の事後条件を満たす必要があります。

- キャッシュ情報が必要でなくなったら、呼び出し元は PvFreeSQLConnectionsData() を呼び出してキャッシュ情報を解放する。

関連項目

PvStart()
PvConnectServer()
PvGetMkdeCommStat()
PvGetSQLConnectionInfo()
PvFreeSQLConnectionsData()
PvDisconnect()
PvStop()

PvGetSQLConnectionInfo()

SQL 接続の情報を照会します。

ヘッダー ファイル：monitor.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetSQLConnectionInfo(
   BTI_LONG           hConnection,
   BTI_ULONG          sequence,
  PVSQLCONNINFO*      pSQLConnInfo);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*sequence*	SQL 接続のシーケンス番号（0 基準）。PvGetSQLConnectionsData() によって取得される SQL 接続の数を上限とした、有効な範囲内の番号でなければなりません。
Out	*pSQLConnInfo*	SQL 接続に関する情報を受け取る PVSQLCONNINFO 構造体のアドレス。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	*hConnection* パラメーターは、有効な接続ハンドルではありません。
P_E_DATA_UNAVAILABLE	SQL 接続に関連するデータがありません。
P_E_NULL_PTR	*pSQLConnInfo* ポインターはヌルです。
P_E_INVALID_SEQUENCE	シーケンス番号が無効です。
P_E_FAIL	名前付きサーバーからの切断に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvGetSQLConnectionsData() 呼び出しによって、SQL 接続のデータが取得されている。
- 呼び出し元に既に、有効な SQL 接続シーケンスがある。

関連項目

PvStart()
PvConnectServer()
PvGetSQLConnectionsData()
PvFreeSQLConnectionsData()
PvDisconnect()
PvStop()

PvGetStringType()

PVSETTING_STRING 設定に関する追加情報を取得します。これは、文字列型の設定にのみ適用されます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetStringType(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pTypeString);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pTypeString*	返される PVSETTING_STRING のサブタイプ。

戻り値

P_OK	操作は成功しました。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は文字列型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

以下に、返される可能性のある PVSETTING_STRING のサブタイプを示します。

- PVSTRING – ディレクトリでもファイルでもない文字列
- PVFILESTRING – ファイルのパスを示す文字列
- PVDIRECTORYSTRING – ディレクトリを示す文字列

サブタイプは config.h に定義されています。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetStringValue()
PvSetStringValue()
PvDisconnect()
PvStop()

PvGetStringValue()

whichData によって指定されるデータ ソースから、文字列型の設定の値（ヌル終端文字列）を取得します。設定によっては、セミコロン（;）で区切られた文字列の一覧が返されます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetStringValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG_PTR     pBufSize,
   BTI_CHAR_PTR      value,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	設定値を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。設定値の実際のサイズを受け取ります。
Out	*value*	設定値を受け取る Long 型整数変数のアドレス。
In	*whichData*	どちらの値を要求するかを示すフラグ。 PVDATA_DEFAULT はデフォルト値を返します。 PVDATA_CURRENT は現在値を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。

P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	要求された設定は文字列型の設定ではありません。
P_E_BUFFER_TOO_SMALL	文字列に対し、割り当てられたバッファーが小さすぎます。返された文字列は切り詰められます。この場合、必要なサイズが *pBufSize* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetStringType()
PvSetStringValue()
PvGetStringValueSize()
PvDisconnect()
PvStop()

PvGetStringValueSize()

whichData によって指定されるデータ ソースから、文字列型の設定の値（ヌル終端文字列）を取得します。設定によっては、セミコロン（;）で区切られた文字列の一覧が返されます。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetStringValueSize(
   BTI_LONG           hConnection,
   BTI_ULONG          settingID,
   BTI_ULONG_PTR      pBufSize,
   BTI_SINT           whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In/Out	*pBufSize*	設定値を受け取るために割り当てられたバッファーのサイズを格納する、符号なし Long 型のアドレス。設定値の実際のサイズを受け取ります。
In	*whichData*	どちらの値を要求するかを示すフラグ。 PVDATA_DEFAULT はデフォルト値を返します。 PVDATA_CURRENT は現在値を返します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_INVALID_DATA_TYPE	要求された設定は文字列型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetStringType()
PvSetStringValue()
PvDisconnect()
PvStop()

PvGetTable()

指定されたテーブルのテーブル属性を返します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvGetTable(
    WORD            dictHandle,
    LPSTR           tableName,
  TABLEINFO**       tableProps,
  COLUMNMAP**         columnList,
    WORD*           columnCount,
  INDEXMAP**        indexList,
    WORD*           indexCount);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	取得するテーブルの名前。
Out	*tableProps*	テーブル情報を含む構造体。
Out	*columnList*	テーブルに定義された列の配列。
Out	*columnCount*	columnList 内の列数。
Out	*indexList*	テーブルに定義されたセグメントの配列。
Out	*indexCount*	*indexList* 配列内のインデックスの数。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	一般エラーが発生しました。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidDictionaryHandle	指定された辞書ハンドルは存在しません。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableProps、indexList、および columnList 配列は、PvFreeTable を使って解放する必要があります。

関連項目

PvStart()
PvOpenDatabase()
PvGetTableNames()
PvFreeTable()
PvFreeTableNames()
PvCloseDictionary()
PvStop()

PvGetTableNames()

開いているデータ辞書にあるすべてのテーブル名を返します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvGetTableNames(
    WORD            dictHandle,
  TABLEMAP**        tableList,
    WORD*           tableCount);
```

引数

In	*dictHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
Out	*tableList*	テーブル名を格納する TABLEMAP 構造体の配列。
Out	*tableCount*	*tableList* に返されるテーブル名の数。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidDictionaryHandle	PvOpenDatabase() によって取得された特定の辞書ハンドルが無効です。

備考

まず、PvOpenDatabase() を使用して辞書を正常に開く必要があります。

tableList 配列は PvFreeTableNames() を使って解放する必要があります。

特定のテーブルについての詳しい情報は、PvGetTable() を使って取得できます。

関連項目

PvStart()
PvOpenDatabase()
PvGetTable()
PvFreeTable()
PvFreeTableNames()
PvCloseDictionary()
PvStop()

PvGetTableStat()

指定されたテーブルの統計情報を返します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvGetTableStat(
   BTI_WORD              dbHandle,
   const BTI_CHAR*       tableName,
   TABLESTAT*            tableStat);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	統計情報を取得したいテーブル名。
Out	*tableStat*	テーブル統計情報を含む TABLESTAT 構造体。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidDictionaryHandle	PvOpenDatabase() によって取得された特定の辞書ハンドルが無効です。
PCM_errTableNotFound	指定されたテーブルが見つかりません。

備考

まず PvOpenDatabase() を使用して、データベース ハンドルを取得する必要があります。

特定のテーブルについての詳しい情報は、PvGetTable() を使って取得できます。

データ ファイル内のレコード数が TABLESTAT 構造体が返すことのできる最大値よりも大きい場合は、代わりに、2 バイトの符号なし整数の最大許容値である 65535 が返されます。

関連項目

PvCloseDatabase()
PvFreeTable()
PvFreeTableNames()
PvGetTable()
PvGetTableStat2()
PvGetTableStat3()
PvOpenDatabase()
PvStart()
PvStop()

PvGetTableStat2()

指定されたテーブルの統計情報を返します。これには、そのデータ ファイルが圧縮データ ページを使用しているかどうかも含まれます。『*Zen Programmer's Guide*』のページ レベル圧縮を用いたファイルの作成および『*Advanced Operations Guide*』のレコードおよびページ圧縮を参照してください。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvGetTableStat2(
   BTI_WORD              dbHandle,
   const BTI_CHAR*       tableName,
   TABLESTAT2*           tableStat2);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	統計情報を取得したいテーブル名。
Out	*tableStat*	テーブル統計情報を含む TABLESTAT2 構造体。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidDictionaryHandle	PvOpenDatabase() によって取得された特定の辞書ハンドルが無効です。
PCM_Success	操作は成功しました。

備考

まず PvOpenDatabase() を使用して、データベース ハンドルを取得する必要があります。

特定のテーブルについての詳しい情報は、PvGetTable() を使って取得できます。

詳細については、TABLESTAT2 と TABLESTAT の相違点を参照してください。

データ ファイル内のレコード数が TABLESTAT2 構造体が返すことのできる最大値よりも大きい場合は、代わりに、4 バイトの符号付き整数の最大許容値である 2,147,483,647 が返されます。

関連項目

PvGetTable()
PvGetTableStat()
PvGetTableStat3()
PvStart()
PvOpenDatabase()

PvOpenDatabase()
PvGetTable()
PvFreeTable()
PvFreeTableNames()
PvCloseDictionary()
PvCloseDatabase()
PvStop()

PvGetTableStat3()

指定されたテーブルの統計情報を返します。これには、最大 2^{63}-1 レコード、つまり 9223372036854775807 まで示すことができる 64 ビットのレコード数も含まれます。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvGetTableStat3(
   BTI_WORD              dbHandle,
   const BTI_CHAR*       tableName,
   TABLESTAT3*           tableStat3);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*tableName*	統計情報を取得したいテーブル名。
Out	*tableStat*	テーブル統計情報を含む TABLESTAT3 構造体。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errInvalidDictionaryHandle	PvOpenDatabase() によって取得された特定の辞書ハンドルが無効です。
PCM_errTableNotFound	指定されたテーブルが見つかりません。

備考

まず PvOpenDatabase() を使用して、データベース ハンドルを取得する必要があります。

特定のテーブルについての詳しい情報は、PvGetTable() を使って取得できます。

詳細については、TABLESTAT3 と TABLESTAT2 の相違点を参照してください。

関連項目

PvGetTable()
PvGetTableStat()
PvGetTableStat2()
PvStart()
PvOpenDatabase()
PvOpenDatabase()
PvGetTable()
PvFreeTable()
PvFreeTableNames()
PvCloseDictionary()
PvCloseDatabase()
PvStop()

PvGetValueLimit()

Long 型の設定の上限および下限を取得します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvGetValueLimit(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_LONG_PTR      pMaxValue,
   BTI_LONG_PTR      pMinValue);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
Out	*pMaxValue*	上限値を受け取る Long 型整数のアドレス。ここにヌルが渡された場合は、返す値がありません。 負の値が返された場合は、次のように解釈します。 /* 最大有効メモリまたはディスク サイズ */P_MAX_MEM_DISK_SIZE -129 /* 使用可能なディスク スペースによって制限される最大サイズ */ P_MAX_LIMITED_BY_DISK -2 /* 使用可能なメモリによって制限される最大サイズ */ P_MAX_LIMITED_BY_MEMORY -1
Out	*pMinValue*	下限値を受け取る Long 型整数のアドレス。ここにヌルが渡された場合は、返す値がありません。

戻り値

P_OK	操作は成功しました。

P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_INVALID_DATA_TYPE	要求された設定は Long 型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetLongValue()
PvSetLongValue()
PvDisconnect()
PvStop()

PvIsDatabaseSecured()

指定されたデータベースにセキュリティが設定されているかどうかを判断します。

ヘッダー ファイル：dtisecurity.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvIsDatabaseSecured(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_LONG_PTR      dbAuthentication);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	調べるデータベースの名前。
Out	*dbAuthentication*	2：ドメイン認証を使用してデータベースにセキュリティが設定されている場合 1：Zen データベース認証を使用してデータベースにセキュリティが設定されている場合 0：データベースにセキュリティが設定されていない場合

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_FAIL	その他の理由によりデータベースを開けませんでした。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvOpenDatabase()
PvSecureDatabase()
PvSecureDatabase2()
PvUnSecureDatabase()
PvCloseDatabase()
PvDisconnect()
PvStop()

PvIsSettingAvailable()

設定に利用できる設定項目かどうかを照会します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvIsSettingAvailable(
   BTI_LONG            hConnection,
   BTI_ULONG           settingID);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。

戻り値

0	設定は利用できません。
0 以外	設定は利用できます。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

設定へアクセスするのに権利が不十分であったり、指定した設定 ID が存在しないために、設定を利用できない場合があります。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvDisconnect()
PvStop()

PvListDSNs()

Pervasive ODBC エンジン インターフェイスのシステムのデータ ソース名（DSN）の一覧を取得します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvListDSNs(
   BTI_LONG          hConnection,
   BTI_ULONG_PTR     pdsnListSize,
   BTI_CHAR_PTR      pdsnList,
   BTI_CHAR          filtering);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In/Out	*pdsnListSize*	DSN 一覧のためのバッファー サイズを格納する、符号なし Long 型のアドレス。返される DSN 一覧の実際のサイズを受け取ります。
Out	*pdsnList*	成功した場合は DSN の一覧が格納されます。
In	*filtering*	Zen の DSN のみが必要な場合は 1 を設定します。すべての DSN が必要な場合は 0 を設定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。

P_E_BUFFER_TOO_SMALL	文字列に対しバッファーが小さすぎます。この場合、必要なバッファー サイズが *pdsnListSize* に返されます。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

ユーザーにログイン要求をしないで DSN の一覧を取得するには、PvConnectServer() を使ってサーバー接続を確立するとき、userName と password に空文字列を渡します。

メモ： userName と password に空文字列を渡して確立した接続は、セキュリティで保護されていない接続であるため、DTI のこれ以外のほとんどの操作を実行するのに十分なアクセス権を持たない接続となります。

例

```
BTI_WORD            res = 0;
BTI_ULONG           dsncount = 0;
BTI_ULONG           dsnListSize = 0;
BTI_CHAR            * dsnList;

// MAX_DSN_NAME_LENGTH は catalog.h 内で
// 32 に定義されている
res = PvCountDSNs (hConnection,
                   &dsnCount,
                   1);
dsnlistSize = dsnCount * (MAX_DSN_NAME_LENGTH+1);
dsnList = new char[dsnListSize];
res = PvListDSNs (hConnection,
                  &dsnListSize,
                  dsnList,
                  1);
```

関連項目

PvStart()
PvConnectServer()
PvCountDSNs()
PvGetDSN()

PvDisconnect()
PvStop()

PvModifyDatabase()

新しいデータベース名、辞書パス、データ パス、およびデータベース フラグに指定された情報を使用して、既存のデータベースを変更します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvModifyDatabase(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbNameExisting,
   BTI_CHAR_PTR      dbNameNew,
   BTI_CHAR_PTR      dictPath,
   BTI_CHAR_PTR      dataPath,
   BTI_ULONG         dbFlags);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbNameExisting*	既存データベースの名前。
In	*dbNameNew*	新しいデータベースの名前。データベース名を変更しない場合は、このパラメーターにヌルを設定します。
In	*dictPath*	辞書パス。
In	*dataPath*	データ パス。デフォルトのデータ パス（つまり、辞書パスと同じパス）を使用するには、この値にヌルを設定します。 複数のパスに保存された MicroKernel エンジン データ ファイルを含んでいるデータベースを変更する場合は、このパラメーターにセミコロン（;）で区切られた一覧を指定します。たとえば、`C:¥data¥path1;C:¥data¥path2` のように指定します。

In	*dbFlags*	データベース フラグ。P_DBFLAG_ 定数を組み合わせて指定できます。 • P_DBFLAG_RI（参照整合性およびトリガーを含む、整合性制約を設定します。） • P_DBFLAG_BOUND（データベース名を辞書にスタンプし、そのデータベースのみが DDF を使用できるようにします。） • P_DBFLAG_DBSEC_AUTHENTICATION（データベース セキュリティ認証の混合セキュリティを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_DBSEC_AUTHORIZATION（データベース セキュリティ認証のデータベース セキュリティ ポリシーを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_LONGMETADATA（メタデータ バージョン 2 を使用します。メタデータのバージョンを参照してください。）

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_NOT_EXIST	名前付きデータベースがサーバー上にありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

Btrieve セキュリティ ポリシー

次の表は、新規データベースでセキュリティ モデルを指定する方法、および既存データベースのセキュリティ モデルを解釈する方法を示しています。セキュリティにほかのフラグの組み合わせを使用すると、ステータス コード 7024 が返される結果となります。

フラグの組み合わせ	相当するセキュリティ モデル
フラグなし	クラシック
P_DBFLAG_DBSEC_AUTHENTICATION	混合
P_DBFLAG_DBSEC_AUTHENTICATION + P_DBFLAG_DBSEC_AUTHORIZATION	データベース

関連項目

PvStart()
PvConnectServer()
PvCreateDatabase()
PvGetDbNamesData()
PvGetDbName()
PvGetDbFlags()
PvGetDbDataPath()
PvGetDbDictionaryPath()
PvGetDbServerName()
PvFreeDbNamesData()
PvDisconnect()
PvStop()

PvModifyDatabase2()

新しいデータベース名、辞書パス、データ パス、データベース フラグおよびコード ページに指定された情報を使用して、既存のデータベースを変更します。この関数は、データベース コード ページが指定されることを除けば、PvModifyDatabase() とまったく同じです。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvModifyDatabase2(
   BTI_LONG            hConnection,
   BTI_CHAR_PTR        dbNameExisting,
   BTI_CHAR_PTR        dbNameNew,
   BTI_CHAR_PTR        dictPath,
   BTI_CHAR_PTR        dataPath,
   BTI_ULONG           dbFlags,
   BTI_LONG            dbCodePage);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbNameExisting*	既存データベースの名前。
In	*dbNameNew*	新しいデータベースの名前。データベース名を変更しない場合は、このパラメーターにヌルを設定します。
In	*dictPath*	辞書パス。
In	*dataPath*	データ パス。デフォルトのデータ パス（つまり、辞書パスと同じパス）を使用するには、ヌルに設定します。 複数のパスに保存された MicroKernel エンジン データ ファイルを含んでいるデータベースを変更する場合は、このパラメーターにセミコロン（;）で区切られた一覧を指定します。たとえば、`C:¥data¥path1;C:¥data¥path2` のように指定します。

In	*dbFlags*	データベース フラグ。P_DBFLAG_ 定数を組み合わせて指定できます。 • P_DBFLAG_RI（参照整合性およびトリガーを含む、整合性制約を設定します。） • P_DBFLAG_BOUND（データベース名を辞書にスタンプし、そのデータベースのみが DDF を使用できるようにします。） • P_DBFLAG_DBSEC_AUTHENTICATION（データベース セキュリティ認証の混合セキュリティを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_DBSEC_AUTHORIZATION（データベース セキュリティ認証のデータベース セキュリティ ポリシーを使用します。Btrieve セキュリティ ポリシーを参照してください。） • P_DBFLAG_LONGMETADATA（メタデータ バージョン 2 を使用します。メタデータのバージョンを参照してください。）
In	*dbCodePage*	Windows プラットフォームのデータベースの場合、番号によってデータベース データとメタデータ文字列のコード ページを示します。 Linux ディストリビューションのデータベースの場合、以下の値のいずれかによってデータベース データとメタデータ文字列のコード ページを示します。 • P_DBCODEPAGE_UTF8 • P_DBCODEPAGE_EUCJP • P_DBCODEPAGE_ISO8859_1 Windows および Linux のデータベースでは、ゼロまたは P_DBCODEPAGE_NA の値も使用できます。 ゼロは旧来の動作を示します。つまり、コード ページは指定されません。サーバー マシンのオペレーティング システムにおけるエンコードをデフォルトで使用します。『*Zen User's Guide*』のコード ページ データベース プロパティも参照してください。 P_DBCODEPAGE_NA はコード ページをそのままの状態にしておくことを指定します（データベース コード ページは変更されません）。 **メモ：**データベース エンジンは、アプリケーションがデータベースに追加するデータおよびメタデータのエンコードを検証**しません**。エンジンは、すべてのデータが、『*Advanced Operations Guide*』のデータベース コード ページとクライアント エンコードで説明されているようにサーバーまたはクライアントのエンコードを使用して入力されるものと想定しています。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_NOT_EXIST	名前付きデータベースがサーバー上にありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

Btrieve セキュリティ ポリシー

Btrieve セキュリティ ポリシーを参照してください。

関連項目

PvConnectServer()
PvCreateDatabase2()
PvCreateDSN2()
PvDisconnect()
PvFreeDbNamesData()
PvGetDbCodePage()
PvGetDbDataPath()
PvGetDbDictionaryPath()
PvGetDbFlags()
PvGetDbName()
PvGetDbNamesData()
PvGetDbServerName()

PvGetDSNEx2()
PvModifyDSN2()
PvStart()
PvStop()

PvModifyDSN()

既存のデータ ソース名を変更します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav78.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvModifyDSN(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   pdsnName,
   BTI_CHAR_PTR   pdsnDesc,
   BTI_CHAR_PTR   pdsnDBQ,
   BTI_LONG       openMode);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pdsnName*	変更する DSN の名前。
In	*pdsnDesc*	DSN の新しい説明。
In	*pdsnDBQ*	DSN の新しいデータベース名。
In	*openMode*	DSN の新しいオープン モード。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE 『*ODBC Guide*』の DSN オープン モードも参照してください。

戻り値

P_OK	操作は成功しました。

P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_DSN_DOES_NOT_EXIST	指定された DSN 名は存在しません。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_INVALID_OPEN_MODE	指定されたオープン モードが無効です。
P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvStart()
PvConnectServer()
PvListDSNs()
PvCreateDSN()
PvGetDSN()
PvGetDSNEx()
PvDeleteDSN()
PvCountDSNs()
PvStop()

PvModifyDSN2()

既存のデータ ソース名を変更します。この関数は、データのエンコード オプションが指定されることを除けば、PvModifyDSN() とまったく同じです。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

この関数を Zen v11 以降のバージョンで使用することは推奨されません。クライアント DSN を使った作業には、ODBC API を使用してください。

構文

```
BTI_API PvModifyDSN(
   BTI_LONG       hConnection,
   BTI_CHAR_PTR   pdsnName,
   BTI_CHAR_PTR   pdsnDesc,
   BTI_CHAR_PTR   pdsnDBQ,
   BTI_LONG       openMode,
   BTI_LONG       translate);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*pdsnName*	変更する DSN の名前。
In	*pdsnDesc*	DSN の新しい説明。
In	*pdsnDBQ*	DSN の新しいデータベース名。
In	*openMode*	DSN のオープン モード。次のいずれか 1 つになります。 • NORMAL_MODE • ACCELERATED_MODE • READONLY_MODE • EXCLUSIVE_MODE 『*ODBC Guide*』の DSN オープン モードも参照してください。
In	translate	データのエンコード オプション。次のいずれか 1 つになります。 • DSNFLAG_DEFAULT • DSNFLAG_OEMANSI • DSNFLAG_AUTO 『*ODBC Guide*』のエンコード変換も参照してください。DSNFLAG_DEFAULT は ODBC アドミニストレーターのエンコード オプション［なし］に該当します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_DSN_DOES_NOT_EXIST	指定された DSN 名は存在しません。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_INVALID_OPEN_MODE	指定されたオープン モードが無効です。
P_E_INVALID_TRANSLATE_OPTION	指定されたエンコード変換オプションが無効です。

P_E_FAIL	データ パスの検索に失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

関連項目

PvConnectServer()
PvCountDSNs()
PvCreateDatabase2()
PvCreateDSN2()
PvDeleteDSN()
PvGetDSN()
PvGetDSNEx2()
PvListDSNs()
PvStart()
PvStop()

PvOpenDatabase()

名前によってデータベースを開き、データベース カタログを操作するのに使用できるハンドルを返します。

ヘッダー ファイル：catalog.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvOpenDatabase(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR_PTR      dbUser,
   BTI_CHAR_PTR      dbPassword,
   BTI_WORD_PTR      dbHandle);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*dbName*	データベース名。
In	*dbUser*	セキュリティが設定されている場合は、データベースのユーザー名。
In	*dbPassword*	セキュリティが設定されている場合は、データベースのパスワード。
Out	*dbHandle*	データベースに返されるハンドル。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。

P_E_FAIL	その他の理由によりデータベースを開けませんでした。
PCM_errSessionSecurityError	ユーザー名またはパスワードが無効です。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- データベースでセキュリティが有効になっている場合は、有効なデータベースのユーザー名とパスワードを入力する。返されたデータベース ハンドルのセキュリティは、そのデータベースに定義されているアクセス権に基づいて施行されます。このセキュリティは SQL または ODBC アクセス方法で見られる動作と一致します。

関連項目

PvStart()
PvConnectServer()
PvGetDbFlags()
PvModifyDatabase()
PvCloseDatabase()
PvDropDatabase()
PvDisconnect()
PvStop()

PvOpenDictionary()

既存の辞書を開きます。辞書の絶対パスまたはデータ ソース名を指定すると、以降の任意の関数への呼び出しで使用する辞書ハンドルが返されます。

メモ： この関数は Zen 9 以降のバージョンでは使用が推奨されません。アプリケーションでこの関数を置き換えるには、PvOpenDatabase() を参照してください。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT PvOpenDictionary(
   LPTSTR          path,
   WORD*           dictHandle,
   LPSTR           user,
   LPSTR           password);
```

引数

In	*path*	辞書ファイルへの完全修飾パス。
Out	*dictHandle*	以降の呼び出しで使用するハンドル。
In	*user*	辞書を開くために必要なユーザー名。この引数はヌルに設定できます。
In	*password*	辞書ファイルを開くためにユーザー名と共に使用します。ヌルでもかまいません。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errMemoryAllocation	メモリの割り当て中にエラーが発生しました。
PCM_errDictionaryPathNotFound	指定された辞書のパスが無効です。
PCM_errDictionaryAlreadyOpen	指定された辞書ファイルは現在開いています。

PCM_SessionSecurityError	ユーザー名かパスワードのどちらかが無効です。

備考

この関数は、DTI を使って DDF にアクセスする場合は、最初に呼び出される必要があります。

複数の辞書を同時に開くことができます。

リソースを解放するには、PvCloseDictionary() を使用します。

関連項目

PvStart()
PvCreateDictionary()
PvCreateDatabase()
PvCloseDictionary()
PvStop()

PvRemoveUserFromGroup()

既存のグループから既存のユーザーを削除します。

ヘッダー ファイル：ddf.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
PRESULT DDFAPICALLTYPE PvRemoveUserFromGroup(
   BTI_WORD           dbHandle,
   const BTI_CHAR*    user,
   const BTI_CHAR*    group);
```

引数

In	*dbHandle*	PvOpenDatabase() によって返される、開いている辞書のハンドル。
In	*user*	データベース ユーザー名。
In	*group*	データベース グループ名。

戻り値

PCM_Success	操作は成功しました。
PCM_errFailed	操作は成功しませんでした。
PCM_errInvalidAccountName	指定されたアカウントまたはユーザー名は存在しません。
PCM_errUserNotPartOfGroup	指定されたユーザーは、グループのメンバーではありません。
PCM_errDatabaseHasNoSecurity	データベースにはセキュリティが設定されていません。
PCM_errSessionSecurityError	不十分な権限でデータベースが開かれました。

備考

以下の前提条件を満たす必要があります。

- まず "Master" ユーザーとして PvOpenDatabase() を使用し、データベースを正常に開いておく。
- 関連するデータベースはデータベース レベルのセキュリティが有効である。
- 指定されたグループおよびユーザー名がデータベースに存在している。
- 指定されたユーザーは、指定されたグループのメンバーである。

次の事後条件を満たす必要があります。

- PvCloseDatabase() を使用してリソースを解放する。

関連項目

PvCreateGroup()
PvCreateUser()
PvAlterUserName()
PvAddUserToGroup()
PvDropGroup()
PvDropUser()
PvOpenDatabase()
PvCloseDatabase()

PvSecureDatabase()

既存のデータベースのセキュリティを有効にします。

ヘッダー ファイル：dtisecurity.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvSecureDatabase2(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR_PTR      dbUser,
   BTI_CHAR_PTR      dbPassword);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは **PvConnectServer()** 関数によって取得されます。
In	*dbName*	データベース名。
In	*dbUser*	データベースのユーザー名ーセキュリティを設定するには Master である必要があります。
In	*dbPassword*	Master ユーザーのデータベース パスワード。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_FAIL	その他の理由によりデータベースを開けませんでした。
PCM_errSessionSecurityError	ユーザー名またはパスワードが無効です。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- セキュリティを有効にする際、データベース ユーザー名として Master を指定し、パスワードを選択している。データベースのセキュリティは、そのデータベースに定義されているアクセス権に基づいて施行されます。このセキュリティは SQL または ODBC アクセス方法で見られる動作と一致します。

関連項目

PvStart()
PvConnectServer()
PvOpenDatabase()
PvUnSecureDatabase()
PvIsDatabaseSecured()
PvCloseDatabase()
PvDisconnect()
PvStop()

PvSecureDatabase2()

既存のデータベースのセキュリティを有効にします。ドメイン認証をサポートしている点が、PvSecureDatabase() とは異なります。

ヘッダー ファイル：dtisecurity.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvSecureDatabase2(
   BTI_LONG          hConnection,
   BTI_CHAR_PTR      dbName,
   BTI_CHAR_PTR      dbPassword,
   BTI_LONG          dbAuthentication);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは **PvConnectServer()** 関数によって取得されます。
In	*dbName*	データベース名。
In	*dbPassword*	Master ユーザーのデータベース パスワード。
In	*dbAuthentication*	有効にする認証の種類。値は、データベースの場合は 1、ドメインの場合は 2 です。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_FAIL	その他の理由によりデータベースを開けませんでした。
PCM_errSessionSecurityError	パスワードが無効です。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- セキュリティを有効にする際、Master ユーザーのパスワードを選択している。データベースのセキュリティは、そのデータベースに定義されているアクセス権に基づいて施行されます。このセキュリティは SQL または ODBC アクセス方法で見られる動作と一致します。
- Windows サーバーに接続している。Linux サーバーへの呼び出しは、一般エラー（ステータス コード 7004）を返します。Active Directory ドメイン認証は Windows のみです。

関連項目

PvStart()
PvConnectServer()
PvOpenDatabase()
PvSecureDatabase()
PvUnSecureDatabase()
PvIsDatabaseSecured()
PvCloseDatabase()
PvDisconnect()
PvStop()

PvSetBooleanValue()

whichData によって指定されるデータ ターゲットに、ブール型の新しい設定値を保存します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvSetBooleanValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_SINT          newValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In	*newValue*	設定される整数値。
In	*whichData*	どちらの値が設定されるかを示すフラグ。 • PVDATA_CURRENT は、設定の変更内容が現在のセッションに適用され、レジストリ、ini、または ncf ファイルへ保存されることを示します。Btrieve 6.15 NT リリースでは、トレース オペレーションに対してのみ有効です。 • PVDATA_PERSISTENT では、設定の変更内容は現在のセッションに適用されません。設定はレジストリ、ini、または ncf ファイルにのみ保存されます。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。

P_E_INVALID_DATA_TYPE	設定はブール型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvConnectServer() を使って管理者レベルの権限でログオンしている。

メモ： この関数は、「制限ユーザー」を使ってログインしているユーザーからは呼び出せません。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetBooleanValue()
PvGetBooleanStrings()
PvIsSettingAvailable()
PvDisconnect()
PvStop()

PvSetLongValue()

whichData によって指定されるデータ ターゲットに、Long 型整数の新しい設定値を保存します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvSetLongValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_LONG          newValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In	*newValue*	設定される整数値。 この関数を呼び出す前に、PvGetValueLimit() 関数を使って、この値が特定の設定に対して有効範囲内にあるかどうかを確認してください。
In	*whichData*	どちらの値が設定されるかを示すフラグ。 PVDATA_CURRENT は現在値を設定します。 PVDATA_PERSISTENT は不変値を設定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_INVALID_DATA_TYPE	設定は Long 型の設定ではありません。

P_E_OUT_OF_RANGE	指定された設定値は、範囲外です。
P_E_FAIL	その他の理由により失敗しました。

備考

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvConnectServer() を使って管理者レベルの権限でログオンしている。

メモ： この関数は、「制限ユーザー」を使ってログインしているユーザーからは呼び出せません。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetLongValue()
PvGetValueLimit()
PvIsSettingAvailable()
PvDisconnect()
PvStop()

PvSetSelectionValue()

whichData によって指定されるデータ ターゲットに、選択タイプの新しい設定値を保存します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvSetSelectionValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_ULONG         numItems,
   BTI_LONG_PTR      pNewValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In	*numItems*	設定される個々の選択項目の数。
In	*pNewValue*	設定される個々の選択項目の配列。
In	*whichData*	どちらの値が設定されるかを示すフラグ。 PVDATA_CURRENT は現在値を設定します。 PVDATA_PERSISTENT は不変値を設定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	設定は選択タイプの設定ではありません。

P_E_INVALID_SELECTION	無効な選択項目が少なくとも 1 つあります。
P_E_FAIL	その他の理由により失敗しました。

備考

この関数は単一選択と複数選択の両方のデータ タイプを使った作業に使用します。単一選択の項目に対して複数の項目を設定した場合は、最初の値が使用されます。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvConnectServer() を使って管理者レベルの権限でログオンしている。

メモ： この関数は、「制限ユーザー」を使ってログインしているユーザーからは呼び出せません。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetSelectionValue()
PvGetSelectionString()
PvGetAllPossibleSelections()
PvCountSelectionItems()
PvIsSettingAvailable()
PvDisconnect()
PvStop()

PvSetStringValue()

whichData によって指定されるデータ ターゲットに、文字列型の新しい設定値を保存します。

ヘッダー ファイル：config.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvSetStringValue(
   BTI_LONG          hConnection,
   BTI_ULONG         settingID,
   BTI_CHAR_PTR      newValue,
   BTI_SINT          whichData);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。
In	*settingID*	設定の一意な識別子。設定の一覧は、PvGetSettingList() から取得できます。
In	*newValue*	設定される文字列値。
In	*whichData*	どちらの値が設定されるかを示すフラグ。 PVDATA_CURRENT は現在値を設定します。 PVDATA_PERSISTENT は不変値を設定します。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_INVALID_DATA_TYPE	設定は文字列型の設定ではありません。
P_E_FAIL	その他の理由により失敗しました。

備考

設定によっては、セミコロン（;）で区切られた複数の文字列を取ることがあります。

以下の前提条件を満たす必要があります。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- PvConnectServer() を使って管理者レベルの権限でログオンしている。

メモ： この関数は、「制限ユーザー」を使ってログインしているユーザーからは呼び出せません。

関連項目

PvStart()
PvConnectServer()
PvGetCategoryList()
PvGetSettingList()
PvGetStringType()
PvGetStringValue()
PvIsSettingAvailable()
PvDisconnect()
PvStop()

PvStart()

Distributed Tuning Interface（DTI）セッションを開始します。この関数は、どの DTI 呼び出しが実行されるよりも先に呼び出す必要があります。

ヘッダー ファイル：connect.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvStart(BTI_LONG reserved);
```

引数

In	*reserved*	今後の使用に備えて予約されています。

戻り値

P_OK	操作は成功しました。
P_E_FAIL	一般エラーが発生しました。

備考

この関数は、DTI のリソースのバインドと初期化を実行します。

例

```
BTI_SINT status = 0;

status = PvStart(0);
// 複数の DTI 呼び出しを実行する
status = PvStop (0);
```

関連項目

PvStop()

PvStop()

DTI セッションを終了し、関連するリソースを解放します。

ヘッダー ファイル：connect.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav75.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_SINT PvStop(BTI_LONG_PTR preserved);
```

引数

In	*preserved*	今後の使用に備えて予約されています。

戻り値

P_OK	操作は成功しました。
P_E_FAIL	一般エラーが発生しました。

備考

この関数は、DTI のリソースを解放し、DTI セッションを閉じます。この関数は、アプリケーションを終了する前に呼び出す必要があります。

例

```
BTI_LONG status = 0;

status = PvStop (0);
```

関連項目

PvStart()

PvUnSecureDatabase()

データベース上でデータベース セキュリティを無効にします。

ヘッダー ファイル：dtisecurity.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvUnSecureDatabase(
   BTI_LONG           hConnection,
   BTI_CHAR_PTR       dbName,
   BTI_CHAR_PTR       dbUser,
   BTI_CHAR_PTR       dbPassword);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは **PvConnectServer()** 関数によって取得されます。
In	*dbName*	データベース名。
In	*dbUser*	データベースのユーザー名ーセキュリティを有効または無効にするには、Master である必要があります。
In	*dbPassword*	Master ユーザーのデータベース パスワード。

戻り値

P_OK	操作は成功しました。
P_E_INVALID_HANDLE	接続ハンドルが無効です。
P_E_NULL_PTR	ヌル ポインターによる呼び出しです。
P_E_ACCESS_RIGHT	操作を実行するための十分なアクセス権がありません。
P_E_FAIL	その他の理由によりデータベースを開けませんでした。
PCM_errSessionSecurityError	ユーザー名またはパスワードが無効です。

備考

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。
- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。
- データベースにセキュリティが設定されている。

関連項目

PvStart()
PvConnectServer()

PvOpenDatabase()
PvSecureDatabase()
PvIsDatabaseSecured()
PvCloseDatabase()
PvDisconnect()
PvStop()

PvValidateLicenses()

接続によって示されるコンピューターのすべてのキーに関する有効性のチェックを開始します。

ヘッダー ファイル：dtilicense.h（ヘッダー ファイルも参照）

関数が最初に利用可能になるライブラリ：w3dbav90.dll（Windows）、libpsqldti.so（Linux）（リンク ライブラリも参照）

構文

```
BTI_API PvValidateLicenses(BTI_LONG hConnection);
```

引数

In	*hConnection*	サーバーを識別する接続ハンドル。接続ハンドルは PvConnectServer() 関数によって取得されます。

戻り値

P_OK	検証操作は正常に終了しました。
P_E_FAIL	検証操作は正常に終了しませんでした。
ライセンス管理または認証に関するステータス コード	License Administrator のステータス コードおよび許可のステータス コードは、『*Status Codes and Messages*』を参照してください。

備考

PvValidateLicenses は、検証チェック**要求**から生じた結果のみを返します。キーの状態に関する情報は何も**返しません**。キーの状態に関する情報も含め、製品情報の XML 文字列を取得するには、別に PvGetProductsInfo() を呼び出す必要があります。

以下の前提条件を満たす必要があります。

- PvStart() 呼び出しによって DTI セッションが開始されている。

- PvConnectServer() によって接続が確立している。ただし、ローカル マシンで操作を実行する場合は、接続ハンドルとして P_LOCAL_DB_CONNECTION を使用できます。

例

```
status = PvValidateLicenses(P_LOCAL_DB_CONNECTION);
```

関連項目

PvGetProductsInfo()
PvStart()
PvStop()