Oracle® Essbase

API Reference

リリース 11.1.2.3

ORACLE

ENTERPRISE PERFORMANCE
MANAGEMENT SYSTEM

Essbase API Reference, 11.1.2.3

著者: EPM 情報開発チーム

目次

ドキュメントのアクセシビリティについて

Oracle のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc を参照してください。

Access to Oracle Support

Oracle サポート・サービスでは、My Oracle Support を通して電子支援サービスを提供しています。詳細情報は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info か、聴覚に障害のあるお客様は http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

第 I 部
予備情報

予備情報の内容：

1 『Oracle Essbase APIリファレンス』の概要

この章の内容

Essbase API 概要

Oracle Essbase は、財務、会計、およびマーケティングなどの部門における、企業を横断したエンド・ユーザー分析者の複雑な計算要件を満たす、ビジネス・パフォーマンス管理ソリューションを提供します。Essbase はローカル・エリア・ネットワーク(LAN)上のクライアント・サーバー・コンピューティング環境で動作し、複数のユーザーが、集中化されたデータを取得および分析できます。

Essbase クライアント・ツールは、次のような様々なインタフェースにより、集中化されたデータへのアクセスを提供します:

- Oracle Hyperion Smart View for Office などのグリッド・インタフェース。
- アプリケーションとデータの管理機能。
- Essbase アプリケーション・プログラミング・インタフェース(API)を使用して開発できるカスタム・プログラム。

Essbase API は、次のような一連の強力で高度な機能を提供します:

- クライアントとサーバー間の透過的なアクセス
- データの操作、集計およびレポート
- サーバーへのログイン手順のカプセル化
- リモート・ファイルの管理
- アプリケーションとデータベースの管理
- ユーザーおよびグループの管理
- 透過的な組込みセキュリティ
- カスタマイズされたメモリーとメッセージ処理
- 複数のプラットフォームのサポート

- 関数ライブラリ。このライブラリを使用すると、C または Visual Basic のプログラムからデータベース・アウトラインを直接作成、操作および保守できます
- 新しい機能のリストは、『Oracle Essbase 新機能』を参照してください。

API とはカスタム・クライアント・プログラムと Essbase 間のインタフェースであり、クライアントとサーバー間のデータの転送を管理します。プログラムによって API 内の関数が呼び出されて、接続する Essbase サーバーからデータが戻されます。

また、クライアントと同じ API 関数を使用して、サーバー・マシン上でカスタム・プログラムを実行することもできます。カスタム API プログラムを書くときは、Essbase サーバー・コンピュータがネットワークでどこに位置するかについて心配する必要はありません。サーバーの位置確認およびデータの転送は API が処理します。

API 用のプログラムを作成する前に、このドキュメントで API の概念と表記規則について理解してください。

API の機能を使用するには、プログラムのソース・コードにヘッダー・ファイルを記述し、一連のライブラリをプログラムにリンクします。

このドキュメントの記載項目

この文書は、Essbase サーバーへアクセスするカスタム・フロントエンド・プログラムを開発するプログラマ向けに作成されています。

『Oracle Essbase API リファレンス』は、Essbase アプリケーション・サーバーにアクセスするカスタム・フロントエンド・プログラムの開発に使用できる関数およびライブラリへの包括的なリファレンスです。この文書では次のことについて記載しています:

- API のインストールと使用方法に関する概要
- C 開発環境、Java 開発環境および Visual Basic 開発環境のプログラマを対象とした具体的な参考資料

必要な基礎知識

このドキュメントを使用するには次が必要です:

- サーバーおよびクライアントで使用するオペレーティング・システムの操作方法。
- Essbase の理解。
- Windows または UNIX におけるプログラミングの知識。
- C 言語、Visual Basic、または Java に関する高度な知識。

このドキュメントでの表記規則

表 1 には、Oracle Essbase API リファレンスでコードと例を理解しやすくするために使用する表記規則を記載しています。

表 1　構文とテキストの表記規則

表記規則	目的	例
等幅フォント	関数、構造体、ファイル、ディレクトリおよび環境変数の名前を表します	ESS_STS_T、ESSAPIW.LIB
斜体	構文内で、ユーザーの指定する値と置換される文字列を表します	EsbOtlCloseOutline (hOutline)
" "	テキスト・パラメータ、またはスペースを含むパラメータは、二重引用符で囲みます	" appName " SETDEFAULTCALC "CALC ALL";
()	関数パラメータは、丸カッコで囲みます。また、丸カッコは演算の実行順序を表します	EsbOtlDeleteMember (hOutline, hMember); (a + b) * c
//	これはコメント記号です。//からその行の行末まではコメントであり、プロセス時には処理されません	//結果を取得
;	これはコマンドの末尾を示す終端文字です	EXIT;

関数参照項目の使用

表 2 には、API 関数項目が提供する情報が提示されています。

表 2　API 関数項目

関数項目	説明
説明	関数の簡単な説明。
構文	関数の構文。関数名と必要なキーワード: **太字**。パラメータ: 斜体。
パラメータ	関数パラメータの定義を表します。

関数項目	説明
戻り値	関数から戻される値を表します。
注意	関数の使用に際しての注意。
アクセス	関数を使用するために必要な、セキュリティその他のアクセス・レベルを表します。
例	関数の使用方法。
関連項目	関連する関数を表します。

2 プログラムの作成

この章の内容

サポートされているコンパイラ

表 3 では、現在の Essbase API のリリースがサポートするコンパイラを示します。

表 3　サポートされているコンパイラ

プラットフォーム	コンパイラ
Windows 2003 Server / 2008 Server (32/64 ビット)	Visual Studio 2010 (Service Pack 1 を導入済)
HP-UX 11.X (64 ビットのみ)	HP-UX C コンパイラ(最新のパッチを適用したバージョン 5 以降)
AIX (5.3 以降、32/64 ビット)	AIX コンパイラ(11.1 以降)
Solaris (10 以降、32/64 ビット)	Sun Studio (12.2 以降)
Red Hat Linux または Oracle Enterprise Linux (4.0 以降、32/64 ビット)	GCC コンパイラ(4.4.4 以降)

注：　Essbase API は VB.NET をサポートしません

Windows make ファイルの例

32 ビットまたは 64 ビット Windows の make ファイルの例を次に示します。64 ビット・プラットフォームのサポートも参照してください。

```
#
   common.mak

# Common Windows settings
```

```
UTF8    = 1

#--------------------------------------------------------------------
# Essbase's include and library path
#--------------------------------------------------------------------
ESSINCDIR = /I$(APIPATH)/api/include
ESSLIBDIR = /LIBPATH:$(APIPATH)/api/lib

#--------------------------------------------------------------------
# MSDEV compiler options
#--------------------------------------------------------------------
CP = cp
MKDIR = mkdir
RM = rm
MAKE = nmake
CC = cl
CPPC = cl
LINK = link
SVRLINK = link

!IF "$(SXR_64BIT)" == "1"
STDLIBS    = kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib kernel32.lib
user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib
oleaut32.lib uuid.lib odbc32.lib odbccp32.lib bufferoverflowu.lib

CFLAGS = /nologo /c /w /D"_CRT_SECURE_NO_DEPRECATE" -DBIT64 -DWIN64
CPPFLAGS = /nologo /c /w /D"_CRT_SECURE_NO_DEPRECATE" -DBIT64 -DWIN64

!IF "$(PROCESSOR_ARCHITEW6432)" == "IA64"
LFLAGS = /nologo /DEBUG /MACHINE:IA64
LPPFLAGS = /nologo /DEBUG /MACHINE:IA64
LIBFLAGS = /nologo /MACHINE:IA64
!ELSE
LFLAGS     = /nologo /DEBUG /MACHINE:AMD64
LPPFLAGS = /nologo /DEBUG /MACHINE:AMD64
LIBFLAGS = /nologo /MACHINE:AMD64
!ENDIF

!ELSE
STDLIBS    = kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib
shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib kernel32.lib
user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib
oleaut32.lib uuid.lib odbc32.lib odbccp32.lib

CFLAGS     = /nologo /MLd /c /w -D_USE_32BIT_TIME_T
CPPFLAGS = /nologo /MLd /c /w
LFLAGS     = /nologo /DEBUG /MACHINE:I386
LPPFLAGS = /nologo /DEBUG /MACHINE:I386
LIBFLAGS = /nologo /MACHINE:I386
!ENDIF

!IF "$(UTF8)" == "0"
ESSLIBS    = essapin.lib essgapin.lib essotln.lib
!ELSE
ESSLIBS = essapinu.lib essgapinu.lib essotlnu.lib
```

```
!ENDIF


#
   Makefile.dat


include common.mak

APITESTSOURCE = \
  CuTest.c \
  EssUtil.c \
  apgd9096056.c \
  capimain.c \

#--------------------------------------------------------
# Make rule
#--------------------------------------------------------

INCDIR1   = /IC:/api_view/src
INCDIR2   = /IK:/essexer/base/src

APITESTMAIN    = capimain
APITESTOBJS    = $(APITESTSOURCE:.c=.obj)

$(APITESTMAIN).exe:    $(APITESTOBJS)
    $(LINK) $(LFLAGS) /out:$(APITESTMAIN).exe $(APITESTOBJS) $(STDLIBS) $(ESSLIBDIR)
$(ESSLIBS)

$(APITESTOBJS):    $(APITESTSOURCE)
    $(CC) $(CFLAGS) $(APITESTSOURCE) $(ESSINCDIR) $(INCDIR1) $(INCDIR2)
```

サポートされているプラットフォーム

Essbase API の現在のリリースでサポートされるプラットフォームのリストは、Oracle Hyperion Enterprise Performance Management System 動作保証マトリックス (http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html) を参照してください

64 ビット・プラットフォームのサポート

- Essbase C API または Visual Basic API を使用して開発されたクライアント・プログラムは、32 ビットまたは 64 ビットの Essbase サーバーに接続している 32 ビットのプラットフォーム上で実行できます。
- 32 ビットの Essbase Visual Basic API を使用して開発され、プリコンパイルされたクライアント・プログラムは、64 ビット Essbase サーバーに接続する 64 ビット Windows プラットフォームで動作します。ただし、32 ビットの実行時環境が正しく設定されている必要があります。

- Essbase C API を使用して開発されたクライアント・プログラムは、64 ビットの Essbase サーバーに接続している 64 ビットのプラットフォーム上で実行できます。
- プリコンパイル済の 32 ビット・クライアント・プログラムを 64 ビット・コンピュータ上で実行する場合は、ESSBASEPATH に 32 ビット・ランタイム・クライアントのインストール・ディレクトリを設定してあり、PATH に ESSBASEPATH ディレクトリの bin サブディレクトリが含まれている状態で、コマンド・プロンプトなどのシェル・ウィンドウから実行します。
- Windows 上で 64 ビットのオブジェクトをビルドするには、次のコンパイラ・フラグおよびリンカー・フラグを使用します:
  - コンパイラ:

```
-DBIT64 -DWIN64
```

  - リンカー(Intel および AMD ベースのプロセッサ):

```
/MACHINE:AMD64
```

命名ルール

API は、その API の命名ルールを関数、定数、およびデータ型に使用します。API の今後のリリースとの互換性を確保するためには、プログラムでこれらの定数とデータ型の宣言を使用してください:

- **関数名** - 関数が実行するアクションについて記述します。名前は、インタフェースを表す接頭辞で始まり、アクションとその対象について記述する 1 つ以上の語句が続きます。名前の各部分はスペースで分割せず、解釈しやすいように大文字を使用します。名前は次のフォーマットに従います:

名前のフォーマットと構成要素	例
InterfaceVerbObject	Interface
プログラミング・インタフェース	EssCreateGroup
Ess = C API	EssUpdate = C API
Esb = Visual Basic API	EsbUpdate = Visual Basic API
EssOtl = C アウトライン API	EssOtlOpenOutline = C アウトライン API
EsbOtl = Visual Basic アウトライン API	EsbOtlOpenOutline = Visual Basic アウトライン API
EssG = C グリッド API	EssGSetGridOption = C グリッド API
Verb 実行するアクション(「Report」など)	EssReportFile (動詞の指定なし)は、レポートを送信します
Object。アクションの対象(「Group」など)	EssUpdate (オブジェクトの指定なし)は、現在のオブジェクトで実行されます

- **データ構造体名**- インタフェースを表す接頭辞で始まり、構造体について記述する語句を含んで、typedef 定義またはマクロを示す接尾辞で終わります。アンダースコア文字が、名前の各部分を分割します。名前は次のフォーマットに従います:

名前のフォーマットと構成要素	例
Interface_Name_Type	Interface
プログラミング・インタフェース(ESS または ESB)	EssCreateGroup
Verb データ型。STR (文字列)など。	ESB_BOOL_T = Visual Basic に対する、Boolean 型の typdef
Type 構造体のタイプ。T(typedef 定義)または M(マクロ)のいずれか。	ESS_STR_T = C 言語の文字列型の typedef

- **C API 定数名** - C のインタフェースを表す接頭辞 ESS から始まり、定数について説明する単語を含み、接尾辞はありません。アンダースコア文字が、名前の各部分を分割します。名前は次のフォーマットに従います:

名前のフォーマットと構成要素	例
Structure データ型または構造体フィールド。(「Boolean」など)	ESS_Structure_Value
Value。定数に保管する値のタイプ	ESS_STS_NOERROR は、ESS_STS_T データ型の値を保管できます

プログラムへの API ファイルの組込み

プログラムで EssbaseAPI を使用するためには、API 定義を含むファイルを含める必要があります。このトピックでは、C または Visual Basic の API に必要なファイルについて説明します。

ヘッダー・ファイル

プログラムがメイン API を使用している場合、`essapi.h` を含める必要があります。アウトライン API を使用している場合、`essotl.h` を含める必要があります。グリッド API を使用している場合、`essgapi.h` を含める必要があります。

C プログラムのための API ファイル

C プログラムでメイン API を使用するためには、適切なソース・モジュールに API ヘッダー定義ファイル(ESSAPI.H)を含める必要があります。このファイルは、必ず C ランタイム・ライブラリ・ヘッダー・ファイルの後に組み込みます。Windows 環境でプログラミングしている場合は、Windows 組込みファイル WINDOWS.H の後に ESSAPI.H を配置します。

C コンパイラのオプション(32 ビット Windows のみ)

Microsoft Visual C++などのカプセル化された C 開発環境を使用している場合、API が正しく動作するよう、コンパイラおよびリンカーのオプションを注意深く確認

する必要があります。特に、構造体フィールドがバイト整列であり、適切なライブラリが使用されていることを確認する必要があります。必ずリンク・プロセスに適切な API ライブラリを組み込むようにします(45 ページの「API ライブラリ」を参照)。

次のプログラム・ステートメントは、バイト整列を確実に行うためのものであり、プログラムの INCLUDE セクションに配置する必要があります:

```
   #ifdef WINNT
#pragma pack (1)
#endif
#include
#include
#pragma pack ()
#endif
```

Visual Basic プログラムのための API ファイル(Windows のみ)

Visual Basic プログラムで API を使用するためには、32 ビット・プログラムには `ESB32.BAS` ファイルを組み込む必要があります。このファイルは、全 Essbase 関数用の定数の定義および宣言を含んでいます。ファイルは、出荷された状態のまま使用することも、用途のニーズに応じてカスタマイズすることも可能です。

➤ `ESB32.BAS` を使用するには:

1. **プロジェクトを開きます。**
2. **「ファイル」>「ファイルの追加」を選択し、`%ESSBASEPATH%\API\INCLUDE\ESB32.BAS` を指定します。**

UNIX でのプログラムの作成

EssbaseAPI は、Essbase がサポートするのと同じプラットフォーム、HP-UX、AIX、Solaris および Linux でサポートされています。EssbaseAPI は、Essbase がサポートするのと同じ CPU アーキテクチャ(32 ビットおよび 64 ビットに関して)をサポートします。『Oracle Hyperion Enterprise Performance Management System インストールおよび構成ガイド』を参照してください。

このトピックでは、API を使用するアプリケーション・プログラムを UNIX 上でコンパイルするために必要になる情報を説明します。

メモリー割当て

UNIX 用の EssbaseAPI は、デフォルトのメモリー関数として、標準 C ライブラリのメモリー割当て関数、**malloc()**、**realloc()**および **free()**を使用します。ESS_INIT_T 初期化構造体の `AllocFunc`、`ReallocFunc` および `FreeFunc` フィールドで NULL を渡せば、デフォルトのメモリー関数を使用します。詳細は、88 ページの「C プログラムにおけるメモリーの使用」を参照してください。

UNIX サポート

EssAutoLogin()は Essbase API の UNIX バージョンではサポートされません。

Essbase API の UNIX バージョンを使用するときは必ず、UNIX のファイル命名ルールに従ってください。

HP-UX 情報

- **HP-UX 付属のファイル** - HP-UX 用の Essbase API に付属のファイルのリストは、45 ページの「API ライブラリ」についての説明を参照してください。

  共有ライブラリを検索する場所をリンカーに伝えるには、次のように-L フラグを使用します:`$(CC) file1.o file2.o -L /essbase/lib -lessapi \ $ (LIBS) -o`

  `libess*.sl` ファイルはすべて、+s フラグと関連付けられています。これにより、リンクしたプログラムが実行されると、SHLIB_PATH 検索パスを使用して、共有ライブラリの位置を確認できます。SHLIB_PATH に関する詳細は、HP-UX プログラミング・マニュアルを確認してください。

- **HP-UX 上でのプログラムのリンク** - Essbase6.0 リリースでは、Essbase と使用されるサード・パーティ・ライブラリとの互換性を維持するため、CC を使用してプログラムをリンクする必要があります。以前のバージョンを使用している場合、ld コンパイラをリンクのために使用する必要があります。

- **HP-UX の make ファイルの例** - 次のサンプルは、HP-UX 用の make ファイルを示しています。

```
# Compiler Flags
CC=cc
CFLAGS = -I$(<Location of API>)/api/include -g

# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinu -lessotlnu -lessgapinu

main: main.o
    $(CC) -o $@ $^ $(LIBS)

main.o: main.c
    $(CC) $(CFLAGS) $< -c -o $@
```

  実際に API ファイルがインストールされたディレクトリが反映されるように、また、必要なコンパイル・オプションを追加して、この make ファイルの例を変更してください。

  リンク行で、リンク先としてライブラリを 3 つしか指定しなくても、`.sl` ファイルは実行時にすべて使用可能である必要があります。

- **HP-UX64 ビットの make ファイルの例** - 64 ビット HP-UX においては、コンパイラ・フラグ`+DD64` を使用します。リンカー・フラグは必要ありません。

```
# Compiler Flags
CC=cc
CFLAGS = +DD64 -I$(<Location of API>)/api/include -g

# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinu -lessotlnu -lessgapinu
main: main.o
    $(CC) -o $@ $? $(LIBS)

main.o: main.c
    $(CC) $(CFLAGS) -c $< -o $@
```

実際に API ファイルがインストールされたディレクトリが反映されるように、また、必要なコンパイル・オプションを追加して、この make ファイルの例を変更してください。

リンク行で、リンク先として Essbase ライブラリを 3 つしか指定しなくても、.so ファイルは実行時にすべて使用可能である必要があります。

AIX 情報

- **AIX 付属のファイル**- AIX 用の Essbase API に付属のファイルのリストは、45 ページの「API ライブラリ」についての説明を参照してください。
- **AIX の make ファイルの例**- 次のサンプルは、AIX 用の make ファイルを示しています。

```
# Compiler Flags
CC=cc_r
CFLAGS = -qcpluscmt -I$(<Location of API>)/api/include -g

# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinuS -lessotlnuS -lessgapinuS
main: main.o
    $(CC) -o $@ $^ $(LIBS)

main.o: main.c
    $(CC) $(CFLAGS) $< -c -o $@
```

実際に API ファイルがインストールされたディレクトリが反映されるように、また、必要なコンパイル・オプションを追加して、この make ファイルの例を変更してください。

64 ビット AIX については、-q64 -DAIX64 -DBIT64 コンパイラ・フラグおよび-b64 リンカー・フラグを使用してください。

Solaris 情報

- **Solaris 付属のファイル**- Solaris 用の Essbase API に付属のファイルのリストは、45 ページの「API ライブラリ」についての説明を参照してください。
- **Solaris の make ファイルの例**- 次のサンプルは、Solaris 用の make ファイルを示しています。

```
# Compiler Flags
CC=cc
CFLAGS = -I$(<Location of API>)/api/include -g

# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinu -lessotlnu -lessgapinu

main: main.o
    $(CC) -o $@ $^ $(LIBS)

main.o: main.c
    $(CC) $(CFLAGS) $< -c -o $@
```

実際に API ファイルがインストールされたディレクトリが反映されるように、また、必要なコンパイル・オプションを追加して、この make ファイルの例を変更してください。

64 ビット Solaris については、`-xarch=generic64 -DBIT64` コンパイラ・フラグおよび`-xarch=generic64` リンカー・フラグを使用してください。

Red Hat Linux 情報

- **Red Hat Linux 付属のファイル**- Red Hat Linux 用の Essbase API に付属のファイルのリストは、45 ページの「API ライブラリ」についての説明を参照してください。
- **Red Hat Linux の make ファイルの例**- 次に示すリストは、GCC コンパイラを使用して Red Hat Linux の API プログラムをコンパイルおよびリンクするための make ファイルの例です:

```
# Compiler Flags
CC=gcc
CFLAGS = -I$(<Location of API>)/api/include -g

# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinu -lessotlnu -lessgapinu

main: main.o
    $(CC) -o $@ $^ $(LIBS)

main.o: main.c     $(CC) $(CFLAGS) $< -c -o $@
```

Linux 64 ビットの make ファイルの例 - 64 ビット Linux においては、コンパイラ・フラグ`-DBIT64` を使用します。

```
# Compiler Flags
CC=gcc
CFLAGS = -I$(<Location of API>)/api/include -g -DBIT64
```

```
# Library files;
LIBS = -L$(<Location of API>)/api/lib -lessapinu -lessotlnu -lessgapinu

main: main.o
 $(CC) -o $@ $^ $(LIBS)

main.o: main.c
 $(CC) $(CFLAGS) $< -c -o $@
```

実際に API ファイルがインストールされたディレクトリが反映されるように、また、必要なコンパイル・オプションを追加して、この make ファイルの例を変更してください。

3 Essbaseと使用中の製品との統合

この章の内容

Essbase ディレクトリ構造

Essbase クライアント・プログラムのインストール先のコンピュータは、表 4 に関する説明にある事前定義済のディレクトリ構造を使用します。ルート・ディレクトリの正確な名前は、ユーザーによるインストール時に選択される名前によって異なりますが、ルート・ディレクトリの下の構造は常に同じです。

表 4　Essbase のインストール用に事前に定義されたディレクトリ構造

ディレクトリ	説明
\root	ルート・ディレクトリ: すべての Essbase ファイル
\root\bin	バイナリ・ディレクトリ: 実行可能ファイル、共有ライブラリ、その他のプログラム・ファイル
\root\client	クライアント・ディレクトリ: クライアント・アプリケーションおよびデータベース・ファイル
\root\client \appname	アプリケーション appName (各アプリケーションに 1 つ含まれる)に関連するファイル
\root\client \appname\dbname	アプリケーション appName のデータベース dbName (アプリケーションのデータベースごとに 1 つ含まれる)に関連するファイル

root ディレクトリには、インストール時にユーザーが選択する任意の名前を付けることができます。

注：　ルート・ディレクトリ名には、スペースを使用できません。

表 5　Essbase API およびランタイム・クライアントのディレクトリ構造(Windows)

コンポーネント	ディレクトリ
ランタイム・クライアント、32 ビット	%EPM_ORACLE_HOME%\common\EssbaseRTC\releaseNumber たとえば、%EPM_ORACLE_HOME%\common\EssbaseRTC\11.1.2.0
ランタイム・クライアント、64 ビット	%EPM_ORACLE_HOME%\common\EssbaseRTC-64\releaseNumber たとえば、%EPM_ORACLE_HOME%\common\EssbaseRTC-64\11.1.2.0
API、32 ビット	%EPM_ORACLE_HOME%\products\Essbase\EssbaseClient\api
API、32 ビット(64 ビットにインストール)	%EPM_ORACLE_HOME%\products\Essbase\EssbaseClient-32\api
API、64 ビット	%EPM_ORACLE_HOME%\products\Essbase\EssbaseClient\api
Oracle ランタイム・ファイル、32 ビット	%EPM_ORACLE_HOME%\bin-32
Oracle ランタイム・ファイル、64 ビット	%EPM_ORACLE_HOME%\bin

ランタイム環境のカスタマイズ

Essbase API によって、いくつかの API 機能へのアクセスをカスタマイズして、プログラムに統合できます。メモリー管理とメッセージ処理のカスタマイズ以外に、次のトピックで説明しているアイテムもカスタマイズできます:

- 41 ページの「Essbase CLIENT ディレクトリへのパスのカスタマイズ」
- 41 ページの「メッセージ・データベースへのパスのカスタマイズ」
- 43 ページの「Essbase ログインのヘルプ・ファイルへのパスのカスタマイズ」
- 43 ページの「自動ログインのための独自のオンライン・ヘルプの作成」

EsxInit()を呼び出す際に Essbase API 初期化構造体の該当するフィールドにエントリを渡して、これらのパスを変更できます。パスを変更できるので、任意の場所にこれらのディレクトリやファイルをインストールして、必要に応じて名前も変更できます。

プログラムに関連付けられたファイルを特定のディレクトリに配置する必要が生じることがあります。その場合は、該当するパスを ESX_INIT_T に明示的に設定する必要があります。

パスを明示的に設定するその他の方法は、ユーザーの ESSBASEPATH および ARBORMSGPATH 環境変数によって異なります。EsxInit()の呼出しの際に、既存の Essbase ファイル(ESSBASEPATH)または ARBORMSGPATH のルート・ディレクトリに基づき、API で初期化構造体のパスを定義できます。

注： 初期化構造体のすべての設定は、API ライブラリの呼出し元プログラムのインスタンスにのみ適用されます。ユーザー・プログラム内のカスタム設定は、API ライブラリを使用している他のプログラムには影響しません。

Essbase CLIENT ディレクトリへのパスのカスタマイズ

API は CLIENT を使用してローカル・アプリケーションやデータベースに関するファイル(データベース・アウトラインやレポート・スクリプトなど)を保管します。CLIENT ディレクトリのディレクトリ構造は、Essbase サーバーの\App ディレクトリ構造に準じています。各アプリケーションには独自のサブディレクトリがあり、アプリケーションのサブディレクトリごと、そのアプリケーションのデータベースごとに個別のサブディレクトリがあります。アプリケーションとデータベースのリストは、特定のサーバーのものと一致する必要はありません。

アプリケーションとデータベースのサブディレクトリ構造は変更できませんが、アプリケーション・ディレクトリを格納するクライアント・ディレクトリはカスタマイズできます。

初期化構造体のローカル・パス・フィールドの設定

クライアント・ディレクトリ・パスを設定する主な方法は、CLIENT ディレクトリのフル・パス名を示す文字列をポイントするように API 初期化構造体の LocalPath フィールドに明示的に設定することです。この設定により、API はすべてのクライアント・アプリケーションおよびデータベース関連ファイルについてこのディレクトリを検索します。たとえば、CLIENT ディレクトリを D:\PRODUCT\CLIENT に設定するには、初期化構造体に次の変更を行います:

```
ESS_INIT_T InitStruct;Initstruct.LocalPath = "D:\PRODUCT";
```

Visual Basic では、Dim pInitをeSB_INIT_TpInit.LocalPath="D:\PRODUCT"に変更します

クライアント・ディレクトリ・パスのもう 1 つの設定方法として、LocalPath を NULL に設定することもできます。するとデフォルト設定として、Essbase が ESSBASEPATH 環境変数を使用して CLIENT ディレクトリへのパスを判断します。

メッセージ・データベースへのパスのカスタマイズ

Essbase は、デフォルトでは ESSBASE.MDB という名前の、メッセージ・データベース・ファイルを使用します。API を使用すれば、メッセージ・データベース・ファイルを希望のファイル名で、希望のディレクトリ・パスに保管できます。ESSBASE.MDB ファイルを使用する必要がありますが、その名前を変更できます。ESX_INIT_T の MessageFile フィールドを使用すると、明示的にメッセージ・データベースの場所および名前を設定できます。

初期化構造体の MessageFile フィールドの設定

メッセージ・データベース・ファイルの名前およびディレクトリ・パスを変更できます。それには、メッセージ・データベースのフル・パスおよびファイル名を示す文字列をポイントするよう、初期化構造体の MessageFile フィールドを設定します。これにより、Essbase メッセージ・システムは、Essbase システム・メッセージのテキストを参照する必要のある場合は常に、指定されたパスおよびファイル名を探します。たとえば、メッセージ・データベース・ファイル

PRODUCT.MDB を呼び出し、C:\PRODUCT\MESSAGE ディレクトリにそれをインストールする場合、次のように初期化構造体を変更します: ESS_INIT_T InitStruct;Initstruct.MessageFile = " :\PRODUCT\MESSAGE \PRODUCT.MDB";

Visual Basic の場合は、次のように変更します

```
   Dim pInit as ESB_INIT_T
pInit.MessageFile="C:\PRODUCT\MESSAGE\PRODUCT.MDB"
```

名前および場所を明示的に設定しない場合、MessageFile フィールドを NULL に設定できます。デフォルトでは、API は、ユーザーのマシン上の ARBORMSGPATH 環境変数で完全修飾ファイル名を探します。この変数が設定されていない場合、API は、ESSBASEPATH 環境変数に\BIN を追加し、そのディレクトリ名を使用して、ESSBASE.MDB を探します。

ARBORMSGPATH 変数の設定

ARBORMSGPATH 環境変数を使用する場合、Windows プラットフォームでプログラミングをしているときは、AUTOEXEC.BAT ファイルに ARBORMSGPATH ステートメントを入れます。UNIX では、使用しているシェルに対応する環境スクリプトで、この変数を設定します。詳細は、インストール・ノートのトピックを参照してください。パスおよびファイル名を C:\PRODUCT\MESSAGE\PRODUCT.MDB へ設定するには、次のステートメントを使用します: ARBORMSGPATH = C:\PRODUCT \MESSAGE\PRODUCT.MDB

ARBORMSGPATH または ESSBASEPATH 環境変数を使用するには、ESX_INIT_T の MessageFile フィールドを NULL に設定します。

Essbase がメッセージ・データベースを見つける方法

Essbase は、メッセージ・データベースを見つけるために、次の優先度検索を実行します:

1. Essbase では、初期化構造体の MessageFile フィールドに指定した、ディレクトリ・パスとファイル名が使用されます。
2. MessageFile フィールドが NULL に設定されている場合は、Essbase では、ARBORMSGPATH 環境変数に指定されたファイル名とディレクトリ・パスが使用されます。
3. ARBORMSGPATH 変数が定義されていない場合は、Essbase では、ESSBASEPATH 環境変数に指定されたディレクトリ・パスの BIN サブディレクトリにある ESSBASE.MDB というファイル名が使用されます。
4. ESSBASEPATH 変数が定義されていない場合、Essbase はエラー・メッセージを表示します。

Essbase ログインのヘルプ・ファイルへのパスのカスタマイズ

Windows 環境では、EsxAutoLogin()を呼び出すと、「ヘルプ」ボタンを含むダイアログ・ボックスが表示されます。また、独自の「ヘルプ」ボタンを含む他のダイアログ・ボックスへのアクセスも提供されます。「ヘルプ」ボタンをクリックすると、Essbase システムのログインに関するヘルプ・トピック(または ESX_INIT_T で指定されたファイル)が表示されます。

独自のヘルプ・ファイルを作成しない場合は、製品インストールでユーザーにデフォルトのヘルプを指定できます。

初期化構造体の HelpFile フィールドの設定

初期化構造体の HelpFile フィールドを API ヘルプ・ファイルのフル・パスとファイル名に設定して、API ヘルプ・ファイルを指定できます。ユーザーがヘルプ画面を呼び出すたびに、API はこのヘルプ・ファイルを検索します。

たとえば、API ヘルプ画面が C:\PRODUCT\HELP ディレクトリの PRODUCT.HLP ファイルに含まれている場合、次のパスに初期化構造体を設定します:

```
ESS_INIT_T InitStruct;  InitStruct.HelpFile = "C:\PRODUCT\HELP\PRODUCT.HLP";
```

Visual Basic では、パスを次のように設定します:

```
Dim pInit as ESB_INIT_T
pInit.HelpFile="C:\PRODUCT\HELP\PRODUCT.HLP"
```

自動ログインのための独自のオンライン・ヘルプの作成

Windows 環境では、EsxAutoLogin()を呼び出すと、「ヘルプ」ボタンを含むダイアログ・ボックスが表示されます。また、独自の「ヘルプ」ボタンを含む他のダイアログ・ボックスへのアクセスも提供されます。「ヘルプ」ボタンをクリックすると、Essbase システムのログインに関するヘルプ・トピック(または ESX_INIT_T で指定されたファイル)が表示されます。

独自のヘルプ・ファイルで EsxAutoLogin()を使用する場合は、次のようにヘルプ・プロジェクト・ファイルに ESSHELP.H を含める必要があります:

```
[MAP]
#include <ESSHELP.H>
```

ESSHELP.H では、API で表示されるダイアログ・ボックスのヘルプ ID を定義しています。ESSHELP.H を組み込む場合、ヘッダー・ファイルの文字列に対応するコンテキスト文字列でヘルプ・ソース・ファイルのトピックを作成する必要があります。たとえば、「ログイン」ダイアログ・ボックスに対しては、コンテキスト文

字列 IDH_SYSTEM_LOGIN_DB でトピックを作成する必要があります。組み込む必要があるコンテキスト文字列のリストは、ESSHELP.H を参照してください。

プログラムにコンテキスト依存のヘルプ領域が他にもある場合、MAP セクションに次のような行を記述してヘッダー・ファイルを追加します:

```
  [MAP]
#include <ESSHELP.H>
#include <MYHELP.H>
```

出荷する必要がある API ファイル

プログラムを Essbase で実行するためには、プログラムを実行する各クライアント・マシンが、必要な Essbase ランタイム・クライアント・ファイルにアクセスする必要があります。ランタイム・クライアントがインストールされている場合は、ファイルがすでに ESSBASEPATH\bin ディレクトリに存在しています。それ以外の場合は、製品独自のインストール・プロセスの一環として、ランタイム・クライアント・ファイルをインストールする必要があります。

注： ESSBASEPATH が、EPM_ORACLE_HOME\common\EssbaseRTC\11.1.2.0(32 ビットの場合)、または EPM_ORACLE_HOME\common\EssbaseRTC-64\11.1.2.0(64 ビットの場合)に設定されていることを確認してください。

一部のプラットフォームでは、EPM_ORACLE_HOME 配下にある Essbase ランタイム・クライアント・ディレクトリに含まれている以上の追加の Oracle ランタイム・ライブラリを配布する必要があります。次のプラットフォームでは、追加のライブラリにアクセスする必要があります:

- 32 ビット Windows
- 64 ビット Windows
- 32 ビット Linux
- 64 ビット Linux
- HP-UX Itanium 64
- Solaris x86 64
- Solaris SPARC64
- AIX 64

追加の Oracle ランタイム・ライブラリは、%EPM_ORACLE_HOME%\bin (Windows の場合)および$EPM_ORACLE_HOME/lib (UNIX の場合)にあります。インストールによっては、これらのランタイム・ライブラリが%ORACLE_HOME%\bin(Windows の場合)または$ORACLE_HOME/lib(UNIX の場合)に置かれている場合もあります。

UNIX プラットフォームでは、ライブラリの配布時にシンボリック・リンクを必ず保存してください。

ファイルの概要

Essbase API ライブラリは、クライアント・マシンまたはアクセス可能なネットワーク・ファイル・サーバー上に配置できます。

オペレーティング・システムが実行時にライブラリを検出できるように、ライブラリを実行可能ファイルと同じディレクトリか、またはユーザーの PATH 変数 (Windows の場合)、LIBPATH (AIX の場合)、SHLIB_PATH (HP-UX の場合)、LD_LIBRARY_PATH (Solaris および Linux の場合)に含まれている 1 つまたは複数のディレクトリに保存しておく必要があります。詳細は、39 ページの「Essbase ディレクトリ構造」を参照してください。

使用しているプログラムが .mdb ファイルを検出できるように、ESSBASEPATH 変数を設定する必要があります。オプションで、ARBORPATH をクライアント側に設定します。

プラットフォームごとのファイル・リスト

アプリケーション・プログラムのユーザーは、特定のファイルのダウンロードを防止するために、Essbase クライアントをインストールできます。Essbase クライアントのインストールについては、『Oracle Hyperion Enterprise Performance Management System インストールおよび構成ガイド』を参照してください。

リンクしているライブラリ・ファイルのリストについては、45 ページの「API ライブラリ」を参照してください。ファイルの完全なリストは、変更される場合があるため提供されていません。

API ライブラリ

それぞれのサポートするプラットフォームのメイン、アウトライン、グリッド API のリンクに必要なファイルを以下に示します。

- **Windows API ライブラリ (32 ビットおよび 64 ビット)**
  - ESSAPINU.LIB
  - ESSOTLNU.LIB
  - ESSGAPINU.LIB
- **AIX API ライブラリ (32 ビットおよび 64 ビット)**
  - libessapinuS.a
  - libessgapinuS.a
  - libessotlnuS.a
- **HP-UX API ライブラリ (32 ビット)**
  - libessapinu.sl
  - libessotlnu.sl
  - libessgapinu.sl
- **HP-UX API ライブラリ (64 ビット)**
  - libessapinu.so

    - libessgapinu.so
    - libessotlnu.so
- **Solaris API ライブラリ (32 ビットおよび 64 ビット)**
    - libessapinu.so
    - libessgapinu.so
    - libessotlnu.so
- **Red Hat Linux の API ライブラリ**
    - libessapinu.so
    - libessgapinu.so
    - libessotlnu.so

アプリケーション・プログラムのインストール

Essbase API プログラムのインストールを作成するとき、アプリケーションのインストールの一部として、API サポート・ファイルを含めたい場合があります。かわりに、ターゲット・マシンに Essbase ランタイム・クライアントをインストールし、環境更新オプションをすべて受け入れることができます。そのプロセスは、API によって必要とされるファイルをすべてインストールし、PATH 変数を設定します。

製品のインストールの一部として Essbase API 環境設定を含むことに決めた場合、Essbase API によって必要とされるファイルをインストールするために、インストール・プロセスを構築する必要があります。必要とされる詳細な手順は、プログラムおよび対象のオペレーティング・システムにより異なります。次の手順は、典型的なインストール・プロセスを示します:

1. ユーザーに対しルート・インストール・ドライブおよびディレクトリ名を指定するように要求します。ルートとは、たとえば C:\Hyperion\products\Essbase など、インストール・ドライブおよびディレクトリの名前を示します。
2. \root および\root\bin ディレクトリを作成します。
3. 製品の実行可能ファイルを\root\bin ディレクトリにコピーします。
4. 他の製品ファイルを\root ディレクトリまたはサブディレクトリにコピーします。
5. ユーザーに対しネットワーク・プロトコルを選択するように要求します。
6. 適切な Essbase ネットワーク・ドライバ・ライブラリを\root\bin ディレクトリにコピーするか名前を変更します。
7. 残りのライブラリ・ファイルを\root\bin ディレクトリにコピーします。
8. メッセージ・データベースを\root\bin ディレクトリにコピーします。
9. ご使用のオペレーティング・システム環境で、ESSBASEPATH 環境変数を定義し、それを\root\と同等にします。ESX_INIT_T 構造体において明示的にクラ

イアント・ディレクトリ・パスを設定しなかった場合のみ、この手順は必要です。

10. ご使用のオペレーティング・システム環境で、ARBORMSGPATH 環境変数を定義し、それを\root\bin\filename と同等にします。これは、メッセージ・データベースのカスタム・ディレクトリ・パスおよびファイル名を指定します。ESX_INIT_T 構造体において明示的にメッセージ・データベース・パスを設定しなかった場合、または ESSBASEPATH 環境変数の使用によってメッセージ・データベースを見つけることができない場合にのみ、この手順は必要です。
11. プログラムの PATH に<EPM_ORACLE_HOME>/bin/を含めます。これは、プログラムが Essbase に接続できるようにするために必要です。UNIX の場合は、プログラムの LD_LIBRARY_PATH に<EPM_ORACLE_HOME>/lib/を含めます。

注： これらの手順は Windows クライアント・マシンに該当します。他のオペレーティング・システムにインストールする場合は、多少異なる手順を必要とします。

異なるプラットフォームに API プログラムをインストール

異なるオペレーティング・システムのプラットフォームにプログラムをインストールする場合、それぞれのオペレーティング・システムによって PATH、ESSBASEPATH および ARBORMSGPATH などの環境変数の設定手順が多少異なります。

Windows では、環境変数が、Windows の「システムのプロパティ」の環境セクションで設定されます。**「スタート」**>**「設定」**>**「コントロール パネル」**>**「システム」**>**「環境」**タブからシステム変数にアクセスします。Windows のパス変数に%ESSBASEPATH%\Bin パス宣言を追加するのは、以前の Windows マシン上で AUTOEXEC.BAT ファイル内の PATH ステートメントを編集するのと同等です。

UNIX システムでは、環境変数は、各ユーザーのログイン・スクリプトを使用して、通常は設定されます。UNIX でこれらの変数を設定するための標準的な手順では、適切な変数を設定するスクリプトをインストールとともに提供します。これは、システム管理者がユーザーのログイン・スクリプトに含めることができます。環境変数の設定の詳細は、{1}『Oracle Hyperion Enterprise Performance Management System インストールおよび構成ガイド』{2}を参照してください。環境変数の設定の詳細は、『Oracle Hyperion Enterprise Performance Management System インストールおよび構成ガイド』を参照してください。

API クライアントの TCP/IP ネットワークの最適化

Essbase の C-API ベースのクライアントはすべて、ネットワーク層を使って Essbase サーバーと情報をやりとりします。クライアントの C-API ベースのアプリケーションから Essbase への要求では、要求の最初に TCP/IP ソケットを開き、要求の終わりにそれを閉じることが必要です。ソケットとは、オペレーティング・システムによって管理されるリソースであり、一定数のそのようなリソースがあります。

その数はオペレーティング・システムに特有です。ソケットは、閉じられると、TIME_WAIT 状態と呼ばれる休止状態に入ります。その期間は、オペレーティング・システムに特有であり構成可能です。その期間の終わりに、ソケットは、再使用のためにオペレーティング・システムがリープできます。ソケットを開く呼出しが正常終了するかどうかは、どれほど速くオペレーティング・システムが再使用のために閉じたソケットをリープできるかどうかの関数です。

API クライアントが、非常に多くの接続を非常に速く確立し終了するよう設計されている場合には、利用可能なポートの定数(約 64,000)が枯渇したり枯渇しそうになるため、問題が発生する場合があります。使用されたポートはまだ TIME_WAIT 状態にあるため、利用可能なポートがオペレーティング・システムによって取得されるまで時間がかかり、結果として、接続が拒否されたり、プログラムの動作が緩慢になります。高度な並行処理が期待されている配置で、これらの兆候が発生している場合、オペレーティング・システムの TIME_WAIT 遅延を短縮することをお薦めします。たとえば、遅延を 4 分から 30 秒に短縮すると、大幅にパフォーマンスが向上する場合があります。

この状況は、コマンド・プロンプト上でコマンド netstat を実行することにより検出できます。その出力は、TIME_WAIT 状態にあるソケットの数を示します。その数が大きいほど、以後の API 要求が失敗する可能性が高くなります。

この状況を回避するためには、オペレーティング・システムの TIME_WAIT 値を小さくすることを検討してください。

Windows では、TIME_WAIT 値は Windows レジストリにあります。

UNIX では、ndd (Solaris と HPUX)、no (AIX)および echo (Linux)などのシステム・ツールを使用して、カーネル・パラメータを操作します。

Solaris および HP-UX で TIME_WAIT 値を表示、調整するには、

```
   ndd -get /dev/tcp tcp_time_wait_interval
ndd -set /dev/tcp tcp_time_wait_interval 30000
```

AIX では、次のコマンドが、すべてのパラメータに対する値を表示します:

```
   /usr/sbin/no -a
```

AIX で次のコマンドを発行し、TCP_TIMEWAIT 状態を 30 秒に設定します(ただし、すでに 30 未満である場合は、調整しないでください):

```
   /usr/sbin/no -o tcp_timewait =2
/usr/sbin/no -o tcp_ephemeral_low = 32768
/usr/sbin/no -o  tcp_ephemeral_high = 65535
```

Linux では、次のコマンドを発行して、timeout_timewait パラメータを 30 秒に設定します:

```
   echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
```

4 簡単なAPIプログラムの作成

この章の内容

はじめに

このトピックでは、API の新規ユーザーにはなじみがないトピックへのヒントなど、簡単な Essbase API アプリケーションの開発に関する詳細が、数多く記述されています。

Essbase API 関数には、C API の場合は「Ess」、Visual Basic API の場合は「Esb」が先頭に付きます。ここでは、両方の言語で使用できる API 関数の操作を説明する場合、先頭に「Esx」が付く関数を使用します。たとえば、**EsxLogin()**は**EssLogin()**、**EsbLogin()**、またはこの両方を指します。同様に、接頭辞「ESX」はデータ型または定数の接頭辞として「ESS」または「ESB」を示します。たとえば、ESX_NULL などです。

このチュートリアルでは、API ドキュメントに付属する関数のサンプル・プログラムを参照します。サンプル・プログラムを探すには、`/Docs/Api/Samples` を探してください。C プログラムは`/Samples/Cexecs` にあります。Visual Basic プログラムは`/Samples/VBexecs` にあります。コンパイル可能なソースとコンパイル済実行ファイルの両方があります。付録 A「API サンプル・プログラム」を参照してください。

デザイン環境設定の問題

Essbase の API プログラムを構築する前に、設計環境でいくつかの構成オプションを設定しておく必要があります。ここでは、Microsoft Visual C++ version 6 および Visual Basic version 6 を中心に説明します。特定の開発環境では、別の方法で構成設定を行う場合もありますが、ここでは API プログラムの構築に役立ついくつかのヒントを紹介します:

- すべての API プログラム構造体にバイト配列を使用します。バイト配列は、ほとんどの C コンパイラではデフォルト設定ではないので注意が必要です!
- そのかわりに、他の整列を設定したコードに自分のコードをリンクする必要がある場合(たとえば、他の外部 API を使用している場合)は、次の`#pragma`

ディレクティブを使用します(ただし、Microsoft C/C++コンパイラでのコンパイル時のみです):

```
     #pragma pack(push,localid,1)
#include <essapi.h>
#pragma pack(pop,localid)
```

- 必ずメモリー容量が大きいモデルでコンパイルしてください(X86 プラットフォームの場合)。
- API を使用するすべてのプログラム・ファイルにヘッダー・ファイル `essapi.h` を含め、アウトライン API を使用するすべてのファイルにヘッダー・ファイル `essotl.h` を含めます。
- リンク・プロセスに適切なリンク・ライブラリを含めます。次のライブラリを追加してください: Windows 用には `ESSAPIN.LIB` を追加します。アウトライン API を使用しているプログラムでは、アウトライン API ライブラリ(Windows 用には `ESSOTLN.LIB`)を追加します。Essbase の API ライブラリに関する詳細は、45 ページの「API ライブラリ」についての説明を参照してください。

基本的要件

すべての API プログラムは、ログインなどのコア操作を実行する必要があります。これらのセクションでは、アプリケーションのシェルを作成するプロセスの詳細を説明し、Essbase API を初めて使用するプログラマを対象としています:

ネストされたプログラム・モデル

API を使用してプログラミングする際、コードにネストしたプログラミング・モデルを取り入れる必要があります。ネストしたプログラミング・モデルでは、コードには初期関数と対応する最終的な関数への呼出しがあります。呼出しはサンドイッチ状に配置され、なんらかのアクションを実行するコードが中間にはさまれます。次の例を検討してください:

```
begin action 1
 begin action 2
    begin action 3
       perform action 3
```

```
        end action 3
        begin action 4
            perform action 4
        end action 4
    end action 2
end action 1
```

この配置の実装は、開始した各アクションおよび操作を終了できるようにするものです。実際の API アクションを使用してより具体的な例を示します:

```
   Initialize the API
    Login to a server
        Connect to a database
            Open a database outline
                Browse the outline
            Close the outline
            Open a report
                Modify & save the report
            Close the report
        Disconnect from a database
    Logout from the server
Terminate the API
```

前述の例では Essbase API にアクセスする任意のコードの基本構造を示しています。

関数の戻りコードの使用

最初に知っておく必要があることの 1 つは、API 関数から戻されるステータス・コードの処理方法です。一般に戻りコードがゼロの場合は正常終了を示し、ゼロ以外の戻りコードはエラーを示します。後者の場合、プログラムは進行中の操作を中断して、デフォルトの状態に戻り、クリーンアップに必要な API 関数のみを呼び出します。プログラムが API に呼出しを行うたびに、戻りコードを確認して適切に処理する必要があります。

API はステータスの戻りコード(ESS_STS_T)の型宣言および定数宣言(ESS_STS_NOERR)を提供します。定数宣言を使用して、API 関数からのステータス戻りコードを実装に依存しない方法でテストできます。

```
   /* C Example of checking return value from an API function */
ESS_STS_T    sts;
if ((sts = EssSomeFunction (.....)) == ESS_STS_NOERR)
{
    do something else;
}
else
{
    process error;
}
' VB Example of checking return value from an API function */
Dim   sts as ESB_STS_T
if ((sts = EsbSomeFunction (.....)) == ESB_STS_NOERR)
```

```
    do something else
else
    process error
endif
```

ネストしたプログラミング・モデルは Essbase 関数が失敗してエラーの戻り値を戻した場合のリソース解放に適しています。次の例を検討してください:

```
   allocate resource 1
begin action 1
    allocate resource 2
    begin action 2
        action 2
    end action 2
    free resource 2
end action 1
free resource 1
```

API 関数の呼出し

各 API 関数には、接頭辞 Ess (C の場合)または Esb (Visual Basic の場合)が付き、その後に verb-object 命名ルールが続きます。たとえば **EssGetDatabaseInfo()**となります。製品の特定の領域に関係するいくつかの関数には、その関係を示すために追加の接頭辞が付いています。たとえば、アウトライン API 関数にはすべて EssOtl または EsbOtl の接頭辞が付いています。

API 関数はすべて、一連の引数をとります。引数は関数ごとに異なり、論理的な順序に従います。通常、ほとんどの関数で最初の引数はハンドルであり、インスタンス・ハンドル、コンテキスト・ハンドル、アウトライン・ハンドルまたはメンバー・ハンドルのいずれかです。「ハンドル」という用語は、(ファイル・ハンドルのように)システムにおいて異なるオブジェクトを追跡するため、API が使用する識別子を意味します。異なるハンドルが、特定の関数によって戻されます。次に、ハンドルはプログラムに保管され、必要に応じて他の API 関数に渡される必要があります。

ハンドルは C と Visual Basic で異なります。様々なタイプの API ハンドルとその使用方法の詳細は、第 6 章「C のメイン API の使用」および第 18 章「Visual Basic のメイン API の使用」を参照してください。

関数へ渡す引数がある場合、通常、引数は順序で次にきます。最後に、関数が値を戻す場合、それらの戻り値を保管する変数は、引数リストの終わりで渡されます。

次の例では、最初の引数はコンテキスト・ハンドル(hCtx)です。次の 2 つの引数(アプリケーションとデータベースの名前、Sample と Basic)が渡され、戻ってくる引数(データベース情報構造体、ESX_DBINFO_T)が最後に渡されます:

```
   /* C Example of passing arguments to an API function */
ESS_STS_T    sts;
ESS_HCTX_T    hCtx;
ESS_PDBINFO_T  pDbInfo;
```

```
sts = EssGetDatabaseInfo (hCtx, "Sample", "Basic", &pDbInfo);
if (sts == ESS_STS_NOERR)
{
    do something;
}

' VB Example of passing arguments to an API function
Dim   sts as ESB_STS_T
Dim   hCtx as ESB_HCTX_T
Dim   DbInfo as ESB_DBINFO_T
sts = EsbGetDatabaseInfo (hCtx, "Sample", "Basic", DbInfo)
if (sts = ESB_STS_NOERR)
    do something
endif
```

C の例では、戻された引数(pDbInfo)が、二重間接(ポインタへのポインタ)として関数に渡されることに注意してください。(&演算子を使用して)宣言された構造体ポインタ変数のアドレスを渡します。次に、この変数は、API 関数が内部に割り当てたデータベース情報構造体のアドレスを割り当てられます。

Visual Basic の例では、呼出し元が最初に構造体(DbInfo)を割り当て、次に構造体を API 関数に(参照で明示的に)渡しています。

API の初期化

すべてのアプリケーション・プログラムは、API を **EsxInit**()で初期化してから、その他の Essbase 関数を使用する必要があります。プログラムは初期化を 1 回のみ実行し、これはプログラムのスタートアップ・シーケンス時に実行することを推奨します。

```
   /* C Example of initializing the API */
ESS_STS_T    sts;
ESS_INIT_T    InitStruct;
ESS_HINST_T   hInst;
/* first clear the init structure (use API defaults) */
memset (&InitStruct, 0, sizeof (ESS_INIT_T));
sts = EssInit (&InitStruct, &hInst);

' VB Example of initializing the API
Dim   sts as ESB_STS_T
Dim   InitStruct as ESB_INIT_T
Dim   hInst as ESB_HINST_T
sts = EsbInit (InitStruct, hInst)
```

大半のアプリケーション・プログラムには、API のデフォルト設定が適しています。設定の変更が必要な場合、使用しているプログラムの API 初期化構造体(148 ページの「ESS_INIT_T」および1282 ページの「ESB_INIT_T」)にある個別のフィールド設定の詳細は、EssInit または EsbInit あるいはその両方を参照してください。

EsxInit()から戻されるインスタンス・ハンドル(hInst)は、プログラム内に保存して後続の API 呼出しで使用できるようにします。このインスタンス・ハンドルはプログラムおよび関連するリソースを API に対して一意に識別します。

付録 A「API サンプル・プログラム」を参照してください。

サーバーへのログオン

API が初期化された後、プログラムは、サーバーでアクションを実行するためには、Essbase サーバーにログインする必要があります。一般には、具体的なアクション(通常は、データベース接続操作)がユーザーによって要求された場合のみ、ログインを実行する必要があります。サーバーへのログインは、そのサーバー上の特定のアプリケーションまたはデータベースへの接続を必ずしも意味しないことに注意してください。管理操作には、特定のデータベースへの接続を必要としないものや、サーバーへの接続さえ必要としないものがあります。

ログインは **EsxLogin**()を使用して実行できます。Microsoft Windows のみについては、カプセル化されたログイン・ダイアログ関数(**EsxAutoLogin**())が利用可能です。この関数が表示するダイアログ・ボックスは、管理サービス・コンソールまたは Smart View が使用するものに似ています。オプションで、そのダイアログ・ボックスを使用して、接続するアプリケーションおよびデータベースを選択できます(56 ページの「データベースへの接続」を参照)。また、ユーザーは、パスワードを変更するなど、他の操作を実行できます。

```
   /* C Example of a login using the EssLogin function */
ESS_STS_T    sts;
ESS_HINST_T   hInst;
ESS_SVRNAME_T  Server = "Larch";
ESS_USERNAME_T  Username = "Joe User";
ESS_PASSWORD_T  Password = "secret";
ESS_ACCESS_T   Access;
ESS_HCTX_T    hCtx = ESS_INVALID_HCTX;
sts = EssLogin (hInst, Server, Username, Password, &Access, &hCtx);

' VB Example of a login using the EsbLogin function
Dim   sts as ESB_STS_T
Dim   hInst as ESB_HINST_T
Dim   Server as ESB_SVRNAME_T
Dim   Username as ESB_USERNAME_T
Dim   Password as ESB_PASSWORD_T
Dim   Access as ESB_ACCESS_T
Dim   hCtx as ESB_HCTX_T
Server = "Larch"
Username = "Joe User"
Password = "secret"
hCtx = ESB_INVALID_HCTX
sts = EsbLogin (hInst, Server, Username, Password, Access, hCtx)
```

次は、ログインの類似した例です。ここでは、**EsxAutoLogin**()を使用します。この関数を使用するとき、ユーザーは、すべての関連情報(サーバー名、ユーザー名、パスワード、アプリケーション名およびデータベース名)をダイアログ・ボックスの適切なフィールドへ入力することによって提供します:

```
   /* C Example of a login using the EssAutoLogin function */
ESS_STS_T       sts;
ESS_HINST_T      hInst;
ESS_ACCESS_T      Access;
ESS_HCTX_T       hCtx = ESS_INVALID_HCTX;
sts = EssAutoLogin (hInst, ESS_NULL, ESS_NULL, ESS_NULL, ESS_NULL,
    ESS_NULL, AUTO_DEFAULT, &Access, &hCtx);

' VB Example of a login using the EsbAutoLogin function
Dim   sts as ESB_STS_T
Dim   hInst as ESB_HINST_T
Dim   Access as ESB_ACCESS_T
Dim   hCtx as ESB_HCTX_T
hCtx = ESB_INVALID_HCTX
sts = EsbAutoLogin (hInst, ESB_NULL, ESB_NULL, ESB_NULL, ESB_NULL,
    ESB_NULL, ESB_AUTO_DEFAULT, Access, hCtx)
```

EssLogin、EsbLogin、EssAutoLogin および EsbAutoLogin を参照してください。

ESX_NULL のかわりに、ユーザー入力パラメータとして文字列変数を関数に渡すと、関数から戻るときには、ユーザーがログイン・ダイアログ・ボックスに入力した値が、これらの文字列変数に入ります。

通常、プログラムは(ユーザー・セッションの最初に)1 度ログインする必要があります。ただし、未使用のサーバー・ポートの拘束が大きな問題である場合、各操作の最初にログインし、各操作の終わりにログアウトすることを検討してください(57 ページの「ログアウト」を参照)。ただし、このプロセスによって、ユーザー応答時間が著しく遅くなる場合があることに注意してください。

EsxLogin()または **EsxAutoLogin**()を使用すると、戻されたログイン・コンテキスト・ハンドル(hCtx)は、以後の API 呼出しのためにプログラム内で保存する必要があります。このログイン・コンテキスト・ハンドルは、API に対するその特定のログインを一意に識別します。

ローカル・コンテキスト・ハンドルの使用

クライアント・マシン上で API の管理操作(ファイル操作など)を実行している場合、API へのローカル・ログインを表すために、ダミーのログイン・コンテキスト・ハンドルを使用できます。ダミーのハンドルは、サーバー・コンテキスト・ハンドルと同様に使用できますが、サーバー特有およびデータベース特有の操作はほとんど実行できません。ローカル・コンテキスト・ハンドルを作成するためには、**EsxCreateLocalContext**()を使用します。次の例を検討してください:

```
   /* C Example of creating a local context handle */
ESS_STS_T    sts;
ESS_HINST_T   hInst;
ESS_HCTX_T    hLocalCtx = ESS_INVALID_HCTX;
sts = EssCreateLocalContext (hInst, ESS_NULL, ESS_NULL, &hLocalCtx);

' VB Example of creating a local context handle
Dim   sts as ESB_STS_T
Dim   hInst as ESB_HINST_T
Dim   hLocalCtx as ESB_HCTX_T
```

```
hLocalCtx = ESB_INVALID_HCTX
sts = EsbCreateLocalContext (hInst, ESB_NULL, ESB_NULL, hLocalCtx)
```

データベースへの接続

多数の Essbase API 関数(サーバー管理、セキュリティ、アウトライン保守など)は、プログラムのログイン後に実行されます。ただし、データベース関連の関数には(たとえばレポート生成や計算実行など)、プログラムが特定のアプリケーションやデータベースを接続していなければならないものも多数あります。**EsxSetActive()**を使用して該当する Essbase データベースを識別してください。**EsxAutoLogin()**を使用してログインして、該当するデータベースを識別することもできます。

ユーザーはデータベースへのアクセスに必要な権限を持っている必要があります。特定のユーザーがアクセス可能なすべてのアプリケーションとデータベースのリストは、**EsxLogin()**で戻され、**EsxListDatabases()**を使用して入手可能です。

稼働していないデータベースに接続した場合、Essbase はデータベースを自動的に起動します。データベースから切断する必要はありません。ただし、同じログイン・コンテキスト・ハンドルを使用して別なデータベースに接続すると、元のデータベースとの接続は切断されます。同時にどうしても複数のデータベースに接続する必要がある場合は、プログラムで複数回ログイン(そして各コンテキスト・ハンドルを個別に管理)する必要があります。

```
   /* C Example of connecting to a database */
ESS_STS_T    sts;
ESS_HCTX_T    hCtx;
ESS_APPNAME_T  AppName = "Sample";
ESS_DBNAME_T   DbName = "Basic";
ESS_ACCESS_T   Access;
sts = EssSetActive (hCtx, AppName, DbName, &Access);

' VB Example of connecting to a database
Dim   sts as ESB_STS_T
Dim   hCtx as ESB_HCTX_T
Dim   AppName as ESB_APPNAME_T
Dim   DbName as ESB_DBNAME_T
Dim   Access as ESB_ACCESS_T
AppName = "Sample"
DbName = "Basic"
sts = EsbSetActive (hCtx, AppName, DbName, Access)
```

選択したデータベースへのユーザーのアクセス権は、EssSetActive(および**EsxAutoLogin()**)によって戻されます。このアクセス・レベルは、セキュリティ定数定義を使用して確認できます。セキュリティ定数定義によって、メニューのグレー表示などにより、アプリケーション・プログラムによるユーザー・オプションの変更が可能になります。

ログアウト

ユーザーが 1 つ以上のデータベース操作を完了して Essbase を終了した後、プログラムはサーバーからログアウトします。ログアウトは、明示的なユーザー要求の結果として、または自動的(たとえば特定のアクション・シーケンスが完了した後)に実行されます。すべてのアクティブな接続もログアウトされてから、プログラムが終了します。

プログラムが常にデータ・アクセス操作のたびにログアウトする必要があるとはかぎりません。ログアウト(Essbase サーバーのポートを解放する)するかログインしたままにする(後続のユーザー要求への応答を速くする)かは、設計上の判断です。

```
   /* C Example of logging a user out */
ESS_STS_T    sts;
ESS_HCTX_T   hCtx;
sts = EssLogout (hCtx);
hCtx = ESS_INVALID_HCTX;

' VB Example of logging a user out
Dim   sts as ESB_STS_T
Dim   hCtx as ESB_HCTX_T
sts = EsbLogout (hCtx)
hCtx = ESB_INVALID_HCTX
```

ログアウトした後、同じコンテキスト・ハンドルを使用しないでください。プログラムがクラッシュすることがあります。

ローカル・コンテキスト・ハンドルの処分が必要な場合は、次の **EsxDeleteLocalContext()**を使用してください:

```
   /* C Example of deleting a local context handle */
ESS_STS_T    sts;
ESS_HCTX_T   hLocalCtx;
sts = EssDeleteLocalContext (&hLocalCtx);

' VB Example of deleting a local context handle
Dim   sts as ESB_STS_T
Dim   hLocalCtx as ESB_HCTX_T
sts = EsbDeleteLocalContext (hLocalCtx)
```

API の終了

実行の終了時に、プログラムは Essbase API を **EsxTerm()**を呼び出して終了させ、すべての API リソースを適切に解放する必要があります。この関数は、すべてのアクティブなサーバー接続のログアウトも行います(すでにプログラムによって明示的にログアウトされていない場合)。

```
   /* C Example of terminating the API */
ESS_STS_T    sts;
ESS_HINST_T   hInst;
```

```
sts = EssTerm (hInst);
hInst = ESS_INVALID_HINST;

' VB Example of terminating the API
Dim   sts as ESB_STS_T
Dim   hInst as ESB_HCTX_T
sts = EsbTerm (hInst)
hInst = ESB_INVALID_HINST
```

API を終了した後、API 関数にさらに呼出しを行わないでください。さらに呼出しを実行すると、プログラムがクラッシュすることがあります。

プログラムのアセンブル

この考察ではこれまでのところ、すべてのプログラムに共通の API の側面について検討してきました。プログラムが行うよう設計されている操作については、まだ説明していません。プログラムはすべて、ネストされたプログラミング・モデルを理解し、API 関数間で引数を一貫した方法で渡し、関数の戻りコードを解釈し、API を初期化し、サーバーにログインし、データベースに接続し、ログアウトして、終了することを要求します。ここでは、プログラムの実際の要点について説明する必要があります。プログラムは、なんらかの操作を実行する必要があるのです。

この考察では、C のメイン API の主要機能グループを扱います。いくつかのセクションは、サンプル・プログラムを取り上げますが、サンプル・プログラムが、API のすべての領域を含んでいるとはかぎりません。サンプル・プログラムは、データをロードし、データベースのコンテンツをレポートし、更新および計算を実行して、データの新しいステータスをレポートします。コード内のコメントは、追加の操作を実行するために関数を今後追加できる場所を示します。

API が実行できる操作のタイプの概要を知るためには、211 ページの「C のメイン API 関数のカテゴリ」および1301 ページの「Visual Basic のメイン API 関数のカテゴリ」を参照してください。C のメイン API には、ほぼ 200 個の関数があり、20 の機能グループに分かれています。つまり、API が実行できる操作は多岐に及びます。C のアウトライン API (78 個の関数)およびグリッド API (59 個の関数)は、API プログラムをさらに複雑にできることを示します。サンプル・プログラムは、できるだけ簡単にする必要があります。したがって、C のメイン API から少数の関数のみを使用し、アウトライン API またはグリッド API はまったく使用していません。

サンプル・プログラムは、Essbase に付属の Sample Basic データベースを使用します。このデータベースは、空の状態で提供されており、データをロードする必要があります。データは、CALCDAT.TXT という名前のテキスト・ファイルで提供されています。サンプル・プログラムは、あらかじめ構成された計算スクリプトおよびレポート・スクリプトを使用します。これらのプログラムが使用するログイン情報(サーバー名、アプリケーション名、データベース名、ユーザー名およびパスワード)は、プログラムにハードコードされています。プログラムはログイン・ダイアログ・ボックスを表示しますが、フィールドはすべて記入されています。ユーザーは、ダイアログ・ボックスに対して「OK」をクリックするのみです。

サーバー名は"LocalHost"です。アプリケーション名は"Sample"です。データベース名は"Basic"です。ユーザー名は"admin"で、パスワードは"password"です。

プログラムをアセンブルする方法について説明するトピック:

次元の構築

次元はデータベースの構成要素です。次元は、データベースの構造(一般にはアウトラインまたはメタデータと呼ばれている)を定義します。データベースの構築は、最初に必要な次元と各次元に関連するメンバーを組み立てて行います。その後、データを追加します。アウトラインは新規作成することも、既存のデータベースの次元やメンバーを追加、または削除して作成することもできます。サンプル Basic アプリケーション/データベースが完全なアウトラインと一緒に提供されているため、サンプル・プログラムの実行用にアウトラインを構築する必要はありません。ただし、Oracle Essbase Administration Services、MaxL またはサンプル・プログラムの実行によって、データをロードする必要があります。

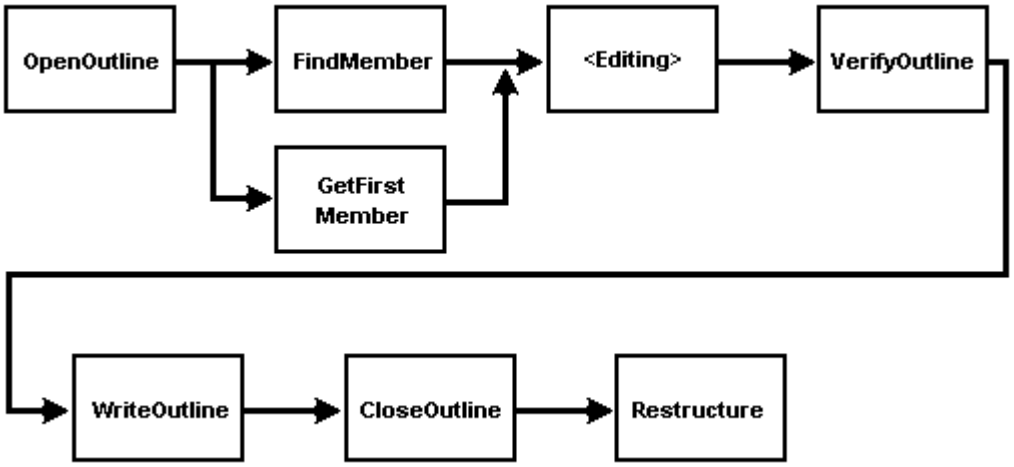
API は、データ・ファイルまたは SQL ソースから動的に次元を再構築する処理を自動化できます。処理を自動化するには、最初に管理サービス・コンソールを使用してルール・ファイルを作成する必要があります。そして、次に `EssBuildDimension` または `EsbBuildDimension` を呼び出して、ルール・ファイルを使用して次元を構築します。

これらの関数はルールとデータ・ファイル・オブジェクト定義を引数として使用し、ルール・ファイルで設定されたパラメータに従ってサーバー上のアウトラインを動的に変更します。また、データベース内のデータを再構築して、アウトライン内の新しい次元に対応させます。

API は必要な構造体が完成するまで、次元やメンバーの追加と削除を行って既存のデータベースを変更します(アウトライン API を使用)。アウトラインが完成すると、`EssImport` または `EsbImport` を使用してデータをデータベースにロードします。

アウトラインの編集

データベースのアウトラインは、アウトライン API 関数を使用して、その中での移動および変更ができます。これらの関数では、アウトライン階層内での移動、メンバーの情報およびプロパティの変更、メンバーの追加および削除などができます。



アウトライン API 関数の制御フロー

アウトラインの使用を開始するには、**EsxOtlOpenOutline()**を呼び出します。アウトラインを編集する場合、**EsxOtlOpenOutline()**へ渡す `fLock` および `fKeepTrans` 引数を両方とも TRUE に設定する必要があります。`fLock` フラグが、他の誰かが更新しないようにアウトラインをロックします(表示はできます)。`fKeepTrans` フラグは、アウトラインが後に再構成されるときのために、アウトラインの編集の間に実行されたトランザクションをすべて保存します。

最初の次元メンバーからアウトライン内での移動を始めるには、**EsxOtlGetFirstMember()**を呼び出します。あるいは、**EsxOtlFindMember()**または **EsxOtlFindAlias()**を使用して、名前でメンバーの位置を指定できます。関数は、そのメンバーに関する情報を取得、設定するため、またはアウトライン階層の隣接したメンバーのメンバー・ハンドルを取得するために使用できる、メンバー・ハンドルを戻します。

現在のメンバーに関する情報を取得するためには、**EsxOtlGetMemberInfo()**、**EsxOtlGetMemberAlias()**および **EsxOtlGetMemberFormula()**を使用します。現在のメンバーの情報を設定するためには、対応する Set 関数を使用します。

メンバーの親を取得するためには、**EsxOtlGetParent()**を呼び出します。メンバーの最初の子を取得するためには、**EsxOtlGetChild()**を呼び出します。メンバーの兄弟を取得するためには、**EsxOtlGetNextSibling()**または **EsxOtlGetPrevSibling()**を呼び出します。メンバーの次の共有オカレンスの位置を確認するためには、**EsxOtlGetNextSharedMember()**を呼び出します。

アウトラインで次元を追加または削除するには、**EsxOtlAddDimension()**または **EsxOtlDeleteDimension()**を使用します。

アウトライン階層内のメンバーを変更するには、**EsxOtlAddMember()**、**EsxOtlDeleteMember()**または **EsxOtlMoveMember()**を使用します。

変更された後のアウトラインについては、確認を行うには **EsxOtlVerifyOutline()**、保存を行うには **EsxOtlWriteOutline()**、閉じるには **EsxOtlCloseOutline()**を使用します。

サーバーのアウトラインへの変更が有効になる前に、**EsxOtlRestructure()**を呼び出して、データベースを再構成する必要があります。この関数は、アウトラインの古いバージョンに対してアウトラインに行われた編集を適用し、アウトラインおよび関連するデータの両方を再構成します。

これらの関数の詳細は、`EssOtlOpenOutline`、`EssOtlGetMemberInfo`、`EsbOtlOpenOutline`、`EsbOtlGetMemberInfo` および各関数の関連項目リストを参照してください。

データのロード

アウトライン次元の作成後、API を使用してデータをデータベースにロードできます。データのロードはルール・ファイルと一緒にデータ・ファイルまたは SQL ソースを使用するか、フリーフォームのデータ・ファイルのロード、あるいはフリーフォームのデータを 1 レコードずつロードして行います。

データ・ファイルまたは SQL ソースと一緒にルールを使用してロードするには、**EsxImport()**を使用します。有効なルールとデータ・ファイル・オブジェクト定義を引数として渡します。フリーフォームのデータ・ファイルをルール・ファイルなしでロードするには、NULL ルール・ファイル・オブジェクト定義を渡すだけでかまいません。

データ・レコードを 1 つずつロードするには、Unlock 引数を FALSE に設定して **EsxBeginUpdate()**を呼び出し、その後、ロードする各データ・レコードで **EsxSendString()**を呼び出します。この方法によって、更新するブロックのロックが必要なくなります。この方法は、バッチ・データ・ロードの場合のみ使用する必要があります。複数のユーザーが存在している場合は、この方法は使用しないでください。ロックをかけ忘れると、データの整合性が失われることがあります。

また、この方法でサーバーに送られた各レコードには、各行の末尾に終了改行文字が必要なことに注意してください。

これらの関数の詳細は、`EssImport`、`EssBeginUpdate`、`EsbImport`、および `EsbBeginUpdate` を参照してください。

45 ページの「API ライブラリ」を参照してください。

レポート作成

Essbase API でのレポート機能では、レポート・スクリプトを使用する必要があります。レポート・スクリプトは API 経由で Essbase サーバーに送信され、実行されます。結果は API 経由で呼出し元に送り返されます。結果の出力データを表示、印刷、ファイルへ送信できます。また、解析してプログラム内の配列データ構造に保管できます。

次のトピックで、レポートについて説明します:

- 62 ページの「レポート・スクリプトの作成」

レポート・スクリプトの作成

レポート・スクリプトは、Essbase サーバーから出力を生成するのに必要なデータ抽出およびデータ・フォーマット・コマンドを含むテキスト文字列です。レポート・ライター言語の詳細な説明は、『Oracle Essbase テクニカル・リファレンス』を参照してください。通常、API アプリケーションのレポート・スクリプトには、次の主要な要素を含む必要があります:

- **{TABDELIMIT}コマンド**- API に送信するレポート・スクリプトの初めに含めます。プログラム内で解析するのに役立つフォーマットで出力データを戻します。このコマンドは、不要なフォーマット(たとえば、数の 1000 ごとの区切りとして使用されるコンマ)をすべて抑制して、解析しセルに分割できるタブ区切りのトークンとして、各メンバー名またはデータ値を戻します。
- **{DECIMALS n}コマンド**- 戻される数値データの小数点以下の桁数を指定します(内部では、数はすべて、精度が 15 桁の浮動小数点数として保管されています)。たとえば、{DECIMALS 2}は、小数点以下の桁数を 2 桁に指定します。
- **{INDENTGEN n}コマンド**- レポート出力の行で親メンバーまたは子メンバーのインデントのオプションが利用できます。n が負の値の場合、子に対して n 個のスペースで親メンバーをインデントします。n が正の値の場合、親に対して n 個のスペースで子メンバーをインデントします。n の値が 0 の場合、すべてのインデントを解除します。たとえば、{INDENTGEN -2}は、レベル当たり 2 つのスペースで親メンバーをインデントします(デフォルト):

```
     100-10  47    41    50    138
100-20  44    38    49    131
100-30  21    14    20    55
 100   112   93    119   324
200-10  25    19    23    67
200-20  18    14    18    50
200-30  17     9    14    40
 200   60    42    55    157
  Product    287   217   290   794
```

- **{SUPMISSING}および{SUPZERO} コマンド**- レポート出力での不要な行を削除します。{SUPMISSING}コマンドは、#Missing 値のみを含む(つまり、実際のデータがない)すべてのデータ列の出力を抑制します。また、{SUPZERO}コマンドは、0 値しか含んでいない行の出力を抑制します。

  0 値と#Missing 値の両方を非表示にする{SUPBLANK}コマンドと、一定範囲のレポート出力パラメータを非表示にする{SUPALL}コマンドも役に立ちます。

- **{MISSINGTEXT string}コマンド**- 出力データ内の#Missing 値をプログラムによって指定された文字列に変換します。たとえば、{MISSINGTEXT "N/A"}は、#Missing 値を文字列「N/A」に変換します。
- **{OUTALTNAMES}または{OUTMBRNAMES}コマンド**- {OUTALTNAMES}によって、出力でメンバー名のかわりに別名を使用できます。メンバー名に戻すには、{OUTMBRNAMES}を使用します(デフォルト)。
- **<PAGE、<COL および<ROW コマンド**- レポートでの異なる次元の配置方法を指定します。<PAGE は、ページ・ヘッダー(レポートの最上部)にどの次元を配置するかを指定します。また、<COL および<ROW は、それぞれ次元を列および行に配置するよう指定します。たとえば、<ROW(Market, Product)は、レポートの行に Market および Product 次元メンバーをこの順に表示します。

  どの次元のどのメンバーも<PAGE、<COL および<ROW で指定できます。各次元は、これらのコマンドで 1 回しか使用できません。それ以外の場合は、最後のコマンドが優先されます。次元はすべて指定する必要があります(そうしないと、レポートのレイアウトは予測不能になります)。
- **メンバー名のリスト**(マクロ・コマンドを含む)- 最も簡単な方法によってレポートで必要とされるデータを抽出するには、関係するメンバーをリストします。たとえば、「Actual Sales Ohio Jan Feb Mar Product」は、次のレポート出力を生成します:

```
         Actual  Sales  Ohio
   Jan   Feb   Mar
Product 287   217   290
```

  またはマクロ・コマンドを使用すると、ある次元から一定範囲のメンバーを指定できます。次の例を検討してください:

  - <CHILDREN / <ICHILDREN
  - <DESCENDANTS / <IDESCENDANTS
  - <DIMBOTTOM
  - <ALLINSAMEDIMENSION
  - <ONSAMELEVELAS
  - <PARENT
  - <ANCESTORS

  注： 前述のマクロ・コマンドはすべて略記できます(たとえば、<DESC、<ICHILD、<PAR)。

  前述のマクロ・コマンドのうち最もよく使用するコマンドは、単一レベルのドリルダウンを実行する場合は<CHILD (または<ICHILD)コマンド、複数レベルのドリルダウンを実行する場合は<DESC (または<IDESC)コマンド、ある次元の最下位レベルのメンバーまでのドリル・ダウンを実行する場合は<DIMBOTTOM コマンドです。

たとえば、「Actual Sales Ohio <ICHILD Qtr1 <DESC Product」と指定すると、次のようなレポート出力が得られます:

```
          Actual  Sales  Ohio
    Jan   Feb   Mar   Qtr1
100-10  47    41    50    138
100-20  44    38    49    131
100-30  21    14    20    55
 100   112   93    119   324
200-10  25    19    23    67
200-20  18    14    18    50
200-30  17     9    14    40
 200   60    42    55    157
300-10  30    19    32    81
300-20  24    16    25    65
300-30  12    7     11    30
 300   66    42    68    176
400-10  30    27    32    89
400-20  14    10    12    36
400-30  5     3     4     12
 400   49    40    48    137
100-20  44    38    49    131
200-20  18    14    18    50
300-30  12    7     11    30
 Diet  74    59    78    211
```

メンバー名が数字(たとえば「100」)であったり、メンバー名にスペースが埋め込まれている(たとえば「New York」)場合があるので、API にレポート・スクリプトを送信するときは、二重引用符でメンバー名を囲むようにしてください。リリース 4.0 以上では、{QUOTEMBRNAMES}コマンドの使用により、このフォーマットでメンバー名を出力するよう強制できます。

- **感嘆符(!)**- レポート・スクリプトの最後の要素は、必ず感嘆符(!)である必要があります。各スクリプトには(少なくとも)1 つの感嘆符がなければ、データが生成されません。レポート・スクリプトが正しく実行されているようでもデータが出力されない場合は、レポート・スクリプトに感嘆符が付記されているかどうか確認してください。

これらの要素の多くは、ユーザー構成が可能な典型的なパラメータです。これらのパラメータは、ユーザーが事前に一括またはレポート単位(あるいはその両方)で設定します。

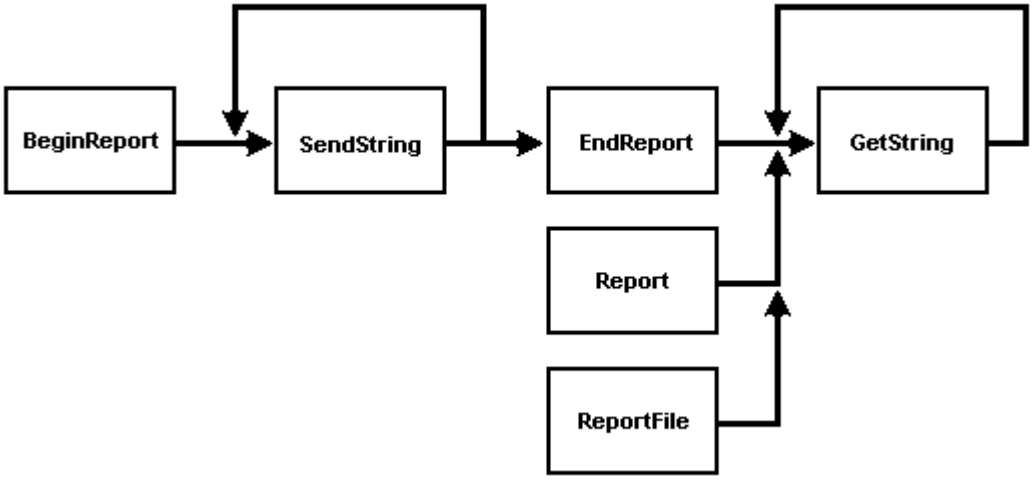
45 ページの「API ライブラリ」を参照してください。

レポート・スクリプトの実行

レポート・スクリプトを実行するには、次の 3 つの方法があります:

- レポート・スクリプトを文字列として **EsxReport()**に渡す方法
- **EsxBeginReport()**、**EsxSendString()**および **EsxEndReport()**を使用して一連の文字列を渡す方法
- **EsxReportFile()**を使用してレポート・スクリプト・ファイルを指定する方法

このいずれのメソッドでも、レポート指定が処理のためにサーバーに送信されます。するとサーバーからの出力がクライアントに戻され、同じコンテキスト・ハンドルを使用して他の API関数を呼び出す前に、そのレポートからすべての出力を読み取る必要があります。



レポート関数の制御フロー

レポートを実行するために、**EsxReport()**を呼び出してレポート・スクリプトを単一の文字列として渡せます。ロックして送信する場合を除き、Output 引数を TRUE に、Lock 引数を FALSE に設定します。

あるいは **EsxBeginReport()** (Output および Lock 引数を前述のように設定)を呼び出して、次に **EsxSendString()**を呼び出し、レポート・スクリプトに 1 つの文字列ずつ送信します。最後に、**EsxEndReport()**を呼び出してレポート・シーケンスを終了します。

ファイルからレポート・スクリプトを実行するには、**EsxReportFile()**を呼び出します。

レポート出力を取得するには、null 値が戻されるまで **EsxGetString()**を繰り返し呼び出して、戻されてくる文字列を読み取ります。「null 値が戻される」とは、C においては null ポインタ値を意味し、Visual Basic においては空のバッファが戻されることを意味します。

これらすべての関数の詳細な説明は、`EssReport`、`EssReportFile`、`EssBeginReport`、`EsbReport`、`EsbReportFile` および `EsbBeginReport` を参照してください

45 ページの「API ライブラリ」を参照してください。

レポート出力の解析

レポートから戻されたデータを解析するには、最初にレポートのフォーマットを理解する必要があります。レポート・スクリプトに{TABDELIMIT}コマンドが含まれる場合、データは次のフォーマットで戻ります:

```
<token><tab><token><tab><token><tab>.....<token><newline>
```

```
<token><tab><token><tab><token><tab>.....<token><newline>
.....
<token><tab><token><tab><token><tab>.....<token><null>
```

たとえば、次のレポート・スクリプトを考えてみます:

```
   {SSFORMAT}{DECIMAL 0} <COL(Year) <ROW(Market) Budget Sales Cola <CHILD Qtr1
<ICHILD Market !
```

このレポート・スクリプトの出力は、通常、次のデータになります:

```
      Budget  Sales  Cola
    Jan   Feb   Mar
East   5200   5000   5300
West   5600   5350   5700
Central 4250   4050   4400
South  3800   3450   3800
 Market 18850  17850  19200
```

レポート・スクリプトに{TABDELIMIT}コマンドを組み込むと、レポート・スクリプトの出力データは次のとおりになります:

```
   <tab>Budget<tab>Sales<tab>Cola<newline>
<tab>Jan<tab>Feb<tab>Mar<newline>
East<tab>5200<tab>5000<tab>5300<newline>
West<tab>5600<tab>5350<tab>5700<newline>
Central<tab>4250<tab>4050<tab>4400<newline>
South<tab>3800<tab>3450<tab>3800<newline>
 Market <tab>18850<tab>17850<tab>19200<null>
```

このフォーマットでデータを解析するには、戻された文字列について、タブ、改行、NULL の有無をスキャンます。これらはそれぞれ、トークンの末尾を定義するものです。トークンは、次の 4 つのタイプのいずれかです:

- メンバー名(先頭は必ず英数字)
- データ値(先頭は必ず数字または負の記号)
- #Missing などの特殊な値(先頭は必ず#文字)
- 空のセル(前述のいずれにも該当しない場合)

レポートが、グリッドや配列などの内部データ構造体に保管されており、レポートの行数または列数が減少している場合(たとえば、ズーム・アウト操作後など)、新しいレポートの境界の調整が必要になることがあります。

数値と数値メンバー名との間の競合は、通常、先頭が数字のトークンをスキャンして、数値のパラメータ(小数点精度など)が適合しているかを検証すれば解決できます。適合していないトークンは、メンバー名として扱います。

より信頼できる方法は、レポートでのトークンの位置決めを使用して、メンバー名かデータ値かを判断する方法です。レポートの最初の x 行は、メンバー名のみ

になります(ここで、x は、列次元の数 + ページ・ヘッダー用の 1 行)。また、最初の y 列は、メンバー名のみになります(ここで、y は行次元の数)。トークンの座標が x および y の両方より大きい場合、トークンは、特別な値(#文字で始まる)または数値のいずれかです。

<QUOTEMBRNAMES コマンドを使用して、すべてのメンバー名の前後に二重引用符を付けることを強制する(それにより、識別の問題を避ける)ことができます。このコマンドを使用すると、先頭の二重引用符によって、メンバー名を認識できます。

戻されるレポート出力トークンを、ページ、列、行、データの各領域で解析することは、多くの場合有益であり、後続のレポートで容易に再利用できます(次の「レポート出力のスクリプトとしての使用」を参照)。

レポート出力のスクリプトとしての使用

レポート・スクリプトからの出力を、別のレポートへの入力として使用できます。レポート出力にはメンバー名とデータしか含まれていないため、ヘッダー・コマンド(前述で説明した)を使用して、新しいレポートにプリフェースを添付する必要があります。次に前のレポートによるメンバー名出力をレポート・ヘッダーに追加し(不要な情報をサーバーに送信しないように、戻されたデータは除外)、それをスクリプトとして実行します。たとえば、次のようなスクリプトが最初に実行されます:

```
   <COL("Year") <ROW("Market")
"Actual" "Sales" "Cola" <CHILD "Qtr1" <CHILD "East"
!
```

その結果として出力されるレポートは、次のようになります:

```
         Actual  Sales  Cola
        Jan   Feb  Mar
New York     36   32   39
Massachusetts   24   09   14
Florida      37   29   37
Connecticut    0    5   11
New Hampshire   12   10   11
```

今度は前のレポートからヘッダー(つまり、フォーマット・コマンドの先頭の 2 行)を送信し、レポート出力からすべてのデータを削除し、すべてのメンバー名を二重引用符で囲んで注意記号(!)を付加すると、次のようなレポート・スクリプトができます:

```
   {TABDELIMIT}{DECIMALS 0} <PAGE("Scenario", "Measures", "Product") <COL("Year")
<ROW("Market")
"Actual" "Sales" "Cola" "Jan" "Feb" "Mar" "New York" "Massachusetts" "Florida"
"Connecticut" "New Hampshire"
!
```

このスクリプトは最初のスクリプトによって生成されたものと同じレポートを生成します。この方法は、ビューに対するドリルダウンなど、一連のアドホック操作を実行する際には便利です。

Essbase によって、特定のメンバー名の前にスペースが挿入されます。<INDENTGEN レポート設定によって、挿入される内容が異なります。メンバーがその後もレポート・スクリプトの一部として使用される場合は、冒頭のスペースは削除されます。

レポート出力を使用したズーム操作の実行

ビュー内の 1 つのメンバーについて単純な(1 レベル)ズーム・インを行うには、ビューを作成したレポートからの出力を<CHILD (または<ICHILD)コマンドでスクリプトとして送信してからメンバーをズームします。複数レベルのズームインを行うには、<DESC または<IDESC コマンドを使用します。ズーム・アウトを実行するには、<PARENT (または場合によっては<ANCESTORS)コマンドを使用します。

たとえば、次のようなレポート出力を考えます:

```
      Actual  Sales  Cola
     Jan   Feb  Mar
East    109   85  112
```

ユーザーが East のドリル・ダウンを選択する場合、レポート・スクリプトは次のようになります:

```
   {SSFORMAT}{DECIMALS 0} <PAGE(Scenario, Measures, Product) <COL(Year) <ROW(Market)
Actual Sales Cola Jan Feb Mar <ICHILD East
!
```

このスクリプトを実行すると、次のレポート出力が生成されます:

```
        Actual  Sales  Cola
       Jan   Feb  Mar
New York    36   32   39
Massachusetts  24   09   14
Florida     37   29   37
Connecticut   0    5   11
New Hampshire  12   10   11
 East     109   85  112
```

表フォーマットのレポート出力の作成

レポートの出力をリレーショナル・データベース・クエリーに類似した表フォーマットに強制できます。次のレポート・ライター・コマンドで、このフォーマットを作成します:

- {ROWREPEAT}コマンド- ネストしたグループがある場合でも、メンバー名のリスト全体をレポートの各行に出力します。次の例では、Ohio が各行で繰り返されます:

```
             Actual  Sales
       Jan   Feb   Mar
Ohio   100-10  130   121   134
Ohio   100-20  118   104   123
Ohio   100-30  77    65    81
```

- {SUPCOLHEADING}コマンド- このコマンドを追加すると、レポート出力の列ヘッダーがレポート上で非表示になります。

```
             Actual  Sales
Ohio   100-10  130   121   134
Ohio   100-20  118   104   123
Ohio   100-30  77    65    81
```

- {SUPHEADING}コマンド- このコマンドを追加すると、レポート出力のページ見出しも非表示になります。次に例を示します:

```
     Ohio   100-10  130   121   134
Ohio   100-20  118   104   123
Ohio   100-30  77    65    81
```

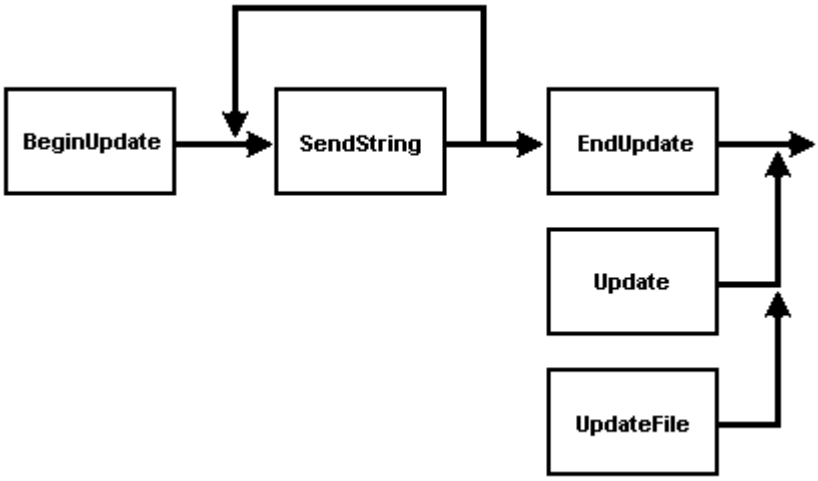
さらに、データを完全に正規化された形式で戻すことを確実にするため、すべての次元(または 1 つを除いたすべての次元)がレポートの<ROW コマンドに含まれるようにする必要があります。

データの更新

データの更新は、ビュー内のデータを変更し、データをサーバーに返送して行います。更新中はユーザーはビューに関連するブロックをロックする必要があります。これによって、プログラムがデータを取得してから、データベースに再度書き込まれるまでの間に、他のユーザーがデータを変更できなくなります。

更新に関するアクションの順序は、次のようになります:

1. レポート・スクリプトを実行して関連のブロックをロックし、更新対象のデータを取得します
2. ビュー内の一部またはすべてのデータを変更します
3. データをサーバーに送り返して、ブロックのロックを解除します



更新関数の制御フロー

EsxReport()または **EsxBeginReport()**を使用してブロックをロックします。これらの関数に渡される Lock 引数が TRUE になっていることを確認します。これによって取得するデータに関連するすべてのブロックがロックされます。これらの関数ではブロックをロックしてデータを取得することも、ブロックのロックのみ行うこともできます(新しいデータまたは最新のデータの場合)。データは取得せずにブロックのロックのみ行う場合は、渡される Output 引数を TRUE または FALSE に適宜変更します。

次に、ユーザーがビュー内のデータ・セルを編集できるようにします(使用している製品が提供する機構を使用します)。

最後に **EsxUpdate()**を呼び出して、これに(更新済の値を含む)ビューのコンテンツすべてを渡します。または **EsxBeginUpdate()**を呼び出し、次に **EsxSendString()**を呼び出すことによって、一度に 1 つの文字列ずつ、ビュー全体をサーバーに送信します。

サーバーに送信される各文字列には、更新を指定した各行の末尾に改行を入れる必要があります。

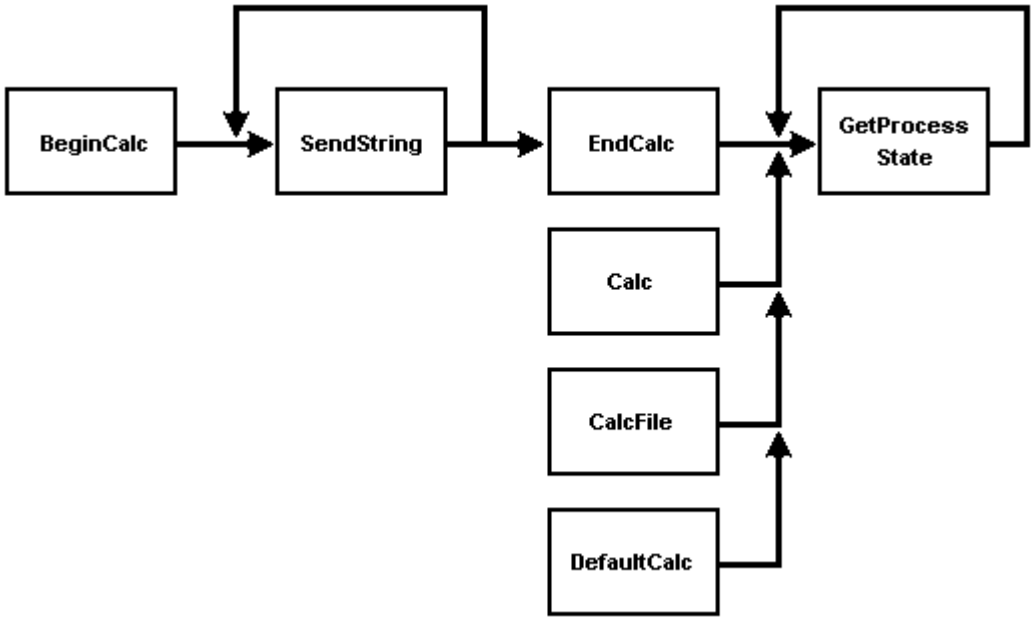
ファイルから更新を実行する場合は、前述のように、最初にブロックをロックしてから **EsxUpdateFile()**を呼び出します。

詳細な説明は、`EssUpdate`、`EssSendString`、`EssBeginUpdate`、`EsbUpdate`、`EsbSendString`、および `EsbBeginUpdate` を参照してください。

45 ページの「API ライブラリ」を参照してください。

データの計算

Essbase におけるデータの計算とは、データベース・アウトラインで定義された階層および式(デフォルト計算)、または計算スクリプトに含まれている式を使用してデータベースの一部または全部を集計することを意味しています。



計算関数の制御フロー

デフォルト計算はデータベースに保管されており、**EsxDefaultCalc()**を呼び出して実行します。デフォルト計算に使用されているスクリプトを入手および設定するには、**EsxGetDefaultCalc()**と、**EsxSetDefaultCalc()**または **EsxSetDefaultCalcFile()**を使用します。

レポートと同様に、計算には次の 3 つの方法があります:

- 計算の指定を文字列として **EsxCalc()**に渡す方法
- **EsxBeginCalc()**、**EsxSendString()**、**EsxEndCalc()**を使用して一連の文字列を渡す方法
- **EsxCalcFile()**を使用して計算スクリプト・ファイルを指定する方法

Essbase における計算は非同期の操作で、該当する計算関数を呼び出すと、API が計算の終了を待たずに、呼出し元にすぐ戻ります(たとえばレポートの実行とは異なります)。計算の完了には時間がかかることもあるため(数時間かかることも多い)、Essbase では非同期に計算を行っています。したがって、計算開始後は、計算が終了したかどうかをプログラムに定期的に確認させる必要があります (**EsxGetProcessState()**の呼出しによって行います)。

最も簡単な確認方法は、システム・タイマーを設定して、頻繁に(5 から 10 秒程度)プロセスをウェイク・アップし、計算のステータスを確認する方法です。計算中はプログラム内で別な操作を行えますが、同じコンテキスト・ハンドルを使用して Essbase API への関数呼出しを行うことはできません。

これらの関数の詳細な説明は、`EssCalc`、`EssBeginCalc`、`EssCalcFile`、`EsbCalc`、`EsbBeginCalc`、および `EsbCalcFile` を参照してください。

45 ページの「API ライブラリ」を参照してください。

セキュリティの使用

セキュリティを管理するために Administration Services が提供する機能はすべて、EssbaseAPI によって利用可能です。セキュリティ・システムの機能を十分に理解するには、『Oracle データベース管理者ガイド』を参照してください。

セキュリティ・システムを使用する関数の多くは、ログイン中のユーザーに利用可能な権限を必要とし、適切な権限なしにセキュリティ情報を変更しようとすると、エラーを戻します。通常、ログイン中のユーザーは管理者権限、アプリケーション・マネージャまたはデータベース・マネージャ権限を持っているはずです。しかし、セキュリティ関数を使用している場合、起こりえる問題に注意し、特に最初のテストの間には、このようなエラーに備えておく必要があります。

Essbase でユーザーまたはグループを作成または削除するためには、**EsxCreateUser()** および **EsxDeleteUser()**を使用します。ユーザーのパスワードを設定するためには、**EsxSetPassword()**を使用します。サーバーのユーザーのリストを取得するためには、**EsxListUsers()**を使用します。

ユーザーまたはグループのセキュリティ情報を取得して設定するには、**EsxGetUser()** および **EsxSetUser()**を呼び出します。

グループのメンバーであるユーザーのリスト(またはメンバーの所属先グループのリスト)を取得して設定するには、**EsxGetGroupList()**および **EsxSetGroupList()**を呼び出します。

アプリケーションに対するユーザー・アクセス権を取得するには、**EsxGetApplicationAccess()**を呼び出します。

セキュリティ関数は、名前付きアプリケーションにアクセスできるすべてのユーザーの名前、名前付きユーザーがアクセスできるすべてのアプリケーション、または特定のアプリケーションとユーザーの組合せのアクセス・レベルを戻せます。類似した関数がデータベース用に存在します。また、対応する Set 関数が、アプリケーションおよびデータベースへのアクセス権を設定するために存在します。

名前付きセキュリティ・フィルタのコンテンツを取得するには、最初に **EsxGetFilter()**を呼び出します。次に、NULL 文字列が戻されるまで、(フィルタの各行の説明を取得するため)**EsxGetFilterRow()**の呼出しを繰り返します。フィルタのコンテンツを設定するには、最初に **EsxSetFilter()**を呼び出します。次に、列がすべて送信される(シーケンスを終了するため、NULL 列ポインタを送信する)まで **EsxSetFilterRow()**の呼出しを繰り返します。

データベースの名前付きフィルタのリストを取得するには、**EsxListFilters()**を呼び出します。名前付きフィルタを割り当てられたユーザーのリストを取得するには、**EsxGetFilterList()**を使用します。

セキュリティ関連の関数の詳細な説明は、224 ページの「C のメイン API セキュリティ・フィルタ関数」および1310 ページの「VB のメイン API セキュリティ・フィルタ関数」に関する説明を参照してください。

アプリケーションおよびデータベースの保守

データベース・アウトラインの保守とは別に、API を使用して実行可能な管理関数が他にもいくつかあります。

アプリケーションに関する情報を入手するには、**EsxGetApplicationInfo()**を使用します。変更可能なアプリケーション状態パラメータを入手するには、**EsxGetApplicationState()**(これらのパラメータの更新用に対応する Set 関数もあります)を呼び出します。データベース用に類似の管理関数が提供されています。

アプリケーションまたはデータベースの Set 関数を使用するには、対応する Get 関数を呼び出して構造体のフィールドを初期化します。

アプリケーション・ログ・ファイルを取得するには、**EsxGetLogFile()**を呼び出します。

データベースの実行時統計値を選択するには、**EsxGetDatabaseStats()**を呼び出します。データベース・ノート(デフォルトのログイン・ダイアログ・ボックスから表示可能なテキスト文字列)を入手または設定するには、**EsxGetDatabaseNote()**および **EsxSetDatabaseNote()**を使用します。

データベースの一部またはすべてをデータベースにリロード可能なテキスト・ファイル・フォーマットにエクスポートするには、**EsxExport()**を使用します。

アプリケーションまたはデータベース間で Essbase ファイル・オブジェクト(アウトライン、計算スクリプト、ルール・ファイルなど)を移動するには、**EsxCopyObject()**を使用します。編集のためにクライアントとサーバー間でオブジェクトを移動するには、**EsxGetObject()**および **EsxPutObject()**を使用します。

オブジェクトを作成するには、**EsxCreateObject()**を呼び出します。オブジェクト名を変更するには、**EsxRenameObject()**を呼び出します。オブジェクトを削除するには、**EsxDeleteObject()**を呼び出します。アプリケーションまたはデータベース内の特定のタイプのオブジェクトをすべてリストするには、**EsxListObjects()**を呼び出します。

データベースとアプリケーションに対する管理関数の使用の詳細は、213 ページの「C のメイン API データベース関数」、212 ページの「C のメイン API アプリケーション関数」、1303 ページの「VB のメイン API データベース関数」、および 1302 ページの「VB のメイン API アプリケーション関数」を参照してください。

45 ページの「API ライブラリ」を参照してください。

メッセージの処理

API には、サーバーで生成されたエラー・メッセージおよび他のメッセージを遮断するための、また、クライアント・プログラムの画面上に適切なメッセージを自動的に表示するためのメカニズムが含まれます。通常、このメカニズムは、役に立ちますが、必要に応じてオフにできます。API では、プログラムが、それらのメッセージが表示されるのを防ぎ、プログラム内で処理するためにそれらをトラップできます。どのメッセージを表示するかを選択し、プログラムの内部メッセージおよびエラー処理と一貫した方法で、メッセージを表示できます。このメカニズムは、Essbase のプログラムとのシームレスな統合を提供します。

Essbase におけるデフォルトのメッセージ・プロセスはプラットフォームに依存しますが、通常はログ情報(アプリケーション名およびデータベース名、ユーザー名、タイムスタンプなど)付きのダイアログ・ボックスと、そのメッセージ・テキストが表示されます。

各 Essbase メッセージには、一意の識別番号、メッセージ・レベル番号および関連するテキスト文字列(必ずしも一意でない)が備わっています。デフォルトでは、Essbase は、重大なエラーのエラー・メッセージのみを表示し、警告や情報のエラー・メッセージは表示しません。

C におけるメッセージ処理

C の API では、カスタム・メッセージ処理関数を定義し、初期化呼出し、**EssInit()** の間に、ポインタをその関数へ渡せます。API がサーバーからメッセージを受け取ると、このカスタム関数が呼び出されます。カスタム関数は、関数の戻りコードを調べて、メッセージを内部で処理するか、またはデフォルトのメッセージ処理のために API にメッセージを戻します。詳細は、90 ページの「C のメイン API メッセージ処理」に関する説明を参照してください。

Windows および C のためのメッセージ処理関数の例を次に示します:

```
   /* C Example of a message handling function */
ESS_FUNC_M ErrorHandler (ESS_PVOID_T  myCtx,
           ESS_LONG_T   MsgNum,
           ESS_USHORT_T  Level,
           ESS_STR_T    LogStr,
           ESS_STR_T    MsgStr)
{
    ESS_STS_T    sts = 0;
    ESS_STR_T    ErrorStr;
    ESS_USHORT_T  len;
    HANDLE     hMem;
    /* Only display messages of level ERROR or above */
    if (Level >= ESS_LEVEL_ERROR)
    {
        /* Calculate combined length of Log and Message strings */
        len = 3;        /* allow for end of line characters + null */
        if (LogStr != NULL)
            len += strlen (LogStr);
        if (MsgStr != NULL)
            len += strlen (MsgStr);
        /* Concatenate the strings */
        if ((hMem = GlobalAlloc (GPTR, len)) != 0)
        {
            ErrorStr = GlobalLock (hMem);
            sprintf (ErrorStr, "%s\n%s", LogStr, MsgStr);
            /* Display message in a Windows message box */
            MessageBox ((HWND)NULL, ErrorStr, "Essbase Error",
                   MB_OK);
            GlobalUnlock (hMem);
            GlobalFree (hMem);
        }
    }
    return (sts);
}
```

Visual Basic におけるメッセージ処理

Visual Basic の API では、メッセージ処理メカニズムは多少異なります。ここでも、初期化呼出し、**EsbInit()**の間に、パラメータを API へ渡します。呼出しは、(Essbase

デフォルト処理を抑制して)カスタム・メッセージ処理を始め、メッセージ・スタックを設定します。その後、プログラムでエラーが発生した場合(API 関数呼出しからの 0 以外の戻り値によって示される)、内部エラー処理関数を呼び出す必要があります。その関数は次に、**EsbGetMessage()**を呼び出して、スタックからメッセージを取得し、ユーザーが選択した方法でメッセージを表示します。詳細は、1221 ページの「Visual Basic の API メッセージの処理」についての説明を参照してください。

Visual Basic におけるメッセージ処理関数の例を次に示します:

```
   ' VB Example of message handler
Dim hInst As Long
Dim hCtx As Long
Dim sts As Long
Dim Server As String * ESB_SVRNAMELEN
Dim User As String * ESB_USERNAMELEN
Dim Password As String * ESB_PASSWORDLEN
Dim Appname As String * ESB_APPNAMELEN
Dim Dbname As String * ESB_DBNAMELEN
Dim Access As Integer
Dim Init As ESB_INIT_T
' GetMessage Variables
Dim Count As Integer
Dim TestApp As String
Dim TestDb As String
Dim TestFtrName As String
Dim ErrMsg As String * 256
Dim ErrNum As Long
Dim ErrLev As Integer
ESB_TRUE = Chr$(1)
ESB_FALSE = Chr$(0)
Init.Maxhandles = 10
Init.ClientError = ESB_TRUE
Init.ErrorStack = 100
sts = EsbInit(Init, hInst)
sts = EsbAutoLogin(hInst, Server, User, Password, Appname, Dbname,
 ESB_AUTO_NOSELECT, Access, hCtx)
If sts <> 0 Then
 sts = EsbGetMessage(hInst, ErrLev, ErrNum, ErrMsg, 256)
 MsgBox ErrMsg & Chr(13) & "Program Ending"
End If
TestApp = "Sample"
TestDb = "Basic"
TestFtrName = "Anything"
'This function call should return an error and then be picked up by EsbGetMessage
sts = EsbGetFilterList(hCtx, TestApp, TestDb, TestFtrName, Count)
If sts <> 0 Then
 sts = EsbGetMessage(hInst, ErrLev, ErrNum, ErrMsg, 256)
 MsgBox "Program Ending" & Chr(13) & Chr(13) & ErrMsg
End If
sts = EsbLogout(hCtx)
sts = EsbTerm(hInst)
End
```

メモリーの管理

C の API の場合にかぎり、カスタム・メモリー管理関数を API 自体の内部で使用するように定義できるため、既存の内部メモリー管理の仕組みが API のメモリー管理の仕組みと競合することはありません。ここでも、カスタム関数によって、API をユーザー・プログラムに統合できます。

最初に、作成するコードの内部に次の 3 つの関数を組み込む必要があります:

- メモリー割当て関数。
- メモリー解放関数。
- メモリー再割当て関数。

次に、**EssInit()**による初期化呼出し中に、この 3 つの関数を API に渡す必要があります。そして API がメモリー・バッファの割当て、解放、再割当てを行う際に、API 内でこれらの関数が使用されます。API 内で割り当てられたすべてのアイテムはユーザー・プログラムに戻され、これらの関数の使用が保証されるため、メモリーの破壊や違反の危険性を伴わずに再割当てや解放が可能です。

API によるカスタム・メモリー管理使用の詳細は、88 ページの「C プログラムにおけるメモリーの使用」を参照してください。

5 Essbase APIプログラムでのUnicodeの問題

この章の内容

プログラミングに関する一般的な注意点

Unicode モードのクライアントのみ、Unicode モードのアプリケーションと完全に機能できます。一般に、Unicode 向け Essbase アプリケーション・プログラムを作成するには、クライアントとサーバーのモードを考慮する必要があります。この説明では、Essbase サーバーは完全に Unicode に対応し、Essbase サーバーは最新バージョンであることを前提としています。

Unicode 対応の Essbase サーバーとクライアントとの通信には、基本的に 3 タイプのシナリオがあります。クライアントのタイプは次の 3 つです:

- Unicode サーバーと通信する非 Unicode クライアント・プログラム
- 非 Unicode モードで Unicode サーバーと通信する Unicode 対応クライアント。
- Unicode サーバーと通信する Unicode 対応クライアント。

非 Unicode モードの Unicode 対応プログラムは Unicode サーバー上のすべてのデータにアクセスできますが、Unicode モード・アプリケーションのデータベース・アウトラインを変更できません。Unicode モードで動作する Unicode 対応クライアント・プログラムのみ、Unicode 対応サーバーのデータとデータベース・アウトラインの両方に完全にアクセスできます。

Unicode モード・クライアント・プログラムの定義

UTF-8 でエンコードしたデータを使用してサーバーと通信できるのは、Unicode モードのクライアント・プログラムのみです。Unicode モードのクライアント・プログラムを初期化するには、**EssInit()**に渡される **ESS_INIT_T** 構造体の usApiType

フィールドを使用します。このフィールドで使用できる値は、**ESS_API_NONUNICODE** と **ESS_API_UTF8** の 2 つです。

この API 初期化関数は、アプリケーション・プログラムのモードを指定する唯一の場所です。このトピックには、次のセクションが含まれます:

非 Unicode クライアント

非 Unicode クライアントは、Essbase API の以前のバージョンで機能する古いクライアントです。この種のクライアントは、短い文字列と非 Unicode エンコードのみ扱います。このような古いクライアントは長い文字列を扱えないため、非 Unicode 対応アプリケーションの処理に限定されています。

このタイプのクライアントでは、Unicode モード・サーバー上のアウトラインまたはルール・ファイルを編集できません。

Unicode 対応サーバーは非 Unicode クライアントと非 Unicode モードで通信できます。

非 Unicode クライアントは、サーバーに接続されていないときにアウトラインおよびルール・ファイルを編集できます。ただし非 Unicode クライアントによるルール・ファイルの編集と、Unicode クライアントによるルール・ファイルおよびアウトラインの編集においては、エンコードが問題になる場合があります。

Unicode モードのサーバーでルール・ファイルまたはアウトラインを編集する際、ユーザーは出力ファイルのフォーマットを選択するか、またはデフォルトで入力ファイルと同じフォーマットにできます。認められている出力ファイル・フォーマットは次のとおりです:

- 非 Unicode フォーマット - 短い文字列と非 Unicode エンコード方式
- Unicode フォーマット - 長い文字列と UTF-8 エンコード方式

これらのファイルは、非 Unicode クライアント内部では非 Unicode エンコード方式で編集され、Unicode クライアント内部では Unicode エンコード方式で編集されます。

入力ファイルを Unicode フォーマットから非 Unicode フォーマットに変換するときに、長すぎる文字列が含まれていて入力ファイルが変換できない場合は、変換が異常終了し、ユーザーに診断メッセージが戻されます。

非 Unicode モードの Unicode 対応クライアント

Unicode 対応クライアントが、組込みファイルと Unicode 対応の Essbase の DLL で構築されましたが、API との通信がネイティブ・エンコードです。API は Unicode 対応の Essbase から、クライアントの API DLL を、非 Unicode の Essbase からの組

込みファイルで構築されたクライアントに配置できません。クライアントを新しい DLL で実行するには、Unicode 対応の Essbase 組込みファイルで構築する必要があります。

非 Unicode モードの Unicode 対応のクライアントでは、Unicode モードのサーバー上のアウトラインまたはルール・ファイルを編集できません。

Unicode 対応の組込みファイルと DLL を使用するには、クライアントがサポートする最大文字列長を長くする必要があります。クライアントによっては、長い最大長を定義した Unicode 対応の Essbase 組込みファイルを使用して再コンパイルするのみでよい場合もあります。

他のクライアントでは、長い最大長をサポートするためにコードを変更する必要がある場合もあります。たとえば、クライアントはメンバー名のバイト長を保存するために 1 バイトを使用する場合があります。メンバー名の新しい最大バイト長(320 バイト)を保存するには 1 バイトでは足りないため、設計を変更して、クライアントが、より長い最大長をサポートできるようにする必要があります。

Unicode モードの Unicode 対応クライアント

Unicode クライアントは Unicode 対応 Essbase を使用して作成され、UTF-8 で API と通信します。

Unicode クライアントを実行するには、クライアントは新しいネイティブ・クライアントについてのサブセクションで前述したように、長い文字列の最大長を処理する必要があります。さらに、クライアントは UTF-8 で API と通信する必要があります。

クライアントが Java で書かれている場合、クライアントがその他の言語で書かれている場合よりも変換は容易です。ただし、いずれの場合も、大幅な変更が発生します。たとえば、クライアント・コードは非 Unicode エンコードのオペレーティング・システムと通信し、Essbase API と Unicode モードで通信する必要があります。

Unicode モードの指定

クライアント・プログラムの Unicode 関連モードは、初期化構造体 ESS_INIT_T でのみ指定できます。何も指定されていない場合、このプログラムは、非 Unicode モードで動作します。モードを指定するには、usApiType フィールドを使用します。

バイト・オーダー・エンコーディングの指定

Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、次の関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります:

- EssBeginReport

- EssBeginUpdate
- EssBeginDataload
- EssBeginDataloadASO
- EssBeginDataloadEx
- EssBeginStreamBuildDim
- EssBeginCalc

BOM を送信するには、次の例に示されているように use EssSendString を使用します:

```
void ESS_BeginUpdate()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_BOOL_T Store;
  ESS_BOOL_T Unlock;
  ESS_STR_T query = "";
  /* Begin Update */
  Store = ESS_TRUE;
  Unlock = ESS_FALSE;
  sts = EssBeginUpdate (hCtx, Store, Unlock);
  printf("EssBeginUpdate sts: %ld\n",sts);
  /* Send update specification */
  //String with BOM characters
  query = "\xEF\xBB\xBF 'marché' 'New York' 'Actual' 'Sales' '100-10' 5";
  if(!sts)
    sts = EssSendString(hCtx, query);
  /* End Update */
  if(!sts)
    sts = EssEndUpdate(hCtx);
}
void ESS_BeginReport()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_STR_T rString = ESS_NULL;
  ESS_STR_T query = ESS_NULL;
  sts = EssBeginReport (hCtx, ESS_TRUE, ESS_FALSE);
  printf("EssBeginReport sts: %ld\n",sts);
  if(!sts)
  {
    //String with BOM characters
    query = "\xEF\xBB\xBF 'New York' 'Actual' 'Sales' '100-10' 'marché' 'Jan' !";
    sts = EssSendString(hCtx, query);
  }
  if(!sts)
    sts = EssEndReport (hCtx);
  if(!sts)
    sts = EssGetString(hCtx,&rString);

  while ((!sts) && (rString != NULL))
  {
    printf("%s", rString);
    EssFree (hInst, rString);
```

```
    sts = EssGetString (hCtx, &rString);
  }
  printf("\n");
}
```

Unicode モードと Essbase サーバー

Essbase サーバーでは Unicode モードにあるときのみ、Unicode モード・アプリケーションの作成、または非 Unicode モード・アプリケーションの Unicode モードへの移行を実行できます。

詳細は、229 ページの「C のメイン API の Unicode モードの関数」を参照してください。

Unicode アウトライン

Unicode モードのアウトラインの操作に関する関数の詳細は、787 ページの「C のアウトライン API の Unicode モードの関数」を参照してください。

グリッド API

グリッド API を使用する Unicode モードのクライアント・プログラムを初期化するには、**EssGInit()**に渡される ESSG_INIT_T 構造体の **usApiType** フィールドを使用します。また、ESSG_DATA_T には、Unicode モードで操作するための usType フィールドの追加値があります。

第 I I 部
CのメインAPI

C のメイン API の内容：

6 CのメインAPIの使用

この章の内容

C のメイン API のインスタンス・ハンドル

インスタンス・ハンドル(概念的にはファイル・ハンドルと同様)は、API へのプログラム・アクセスを示し、API 内で使用されるプログラム固有のリソースと設定を識別します。この種の識別は、複数のプログラムによって同時にアクセスされることのある動的共有ライブラリでは必要です。**EssInit()**の呼出しによって、プログラムが API を初期化すると、インスタンス・ハンドルが戻されます。

アプリケーションにおけるインスタンス・ハンドルの使用

C プログラムでは、インスタンス・ハンドルは **ESS_HINST_T** 型として宣言します。

インスタンス・ハンドルは、**EssLogin()**を呼び出すときに渡す必要があります。この関数からはコンテキスト・ハンドルが戻されます。また、API で使用したプログラム固有のリソースをすべて解放するには、API 終了関数 **EssTerm()**に渡す必要があります。

インスタンス・ハンドルを他のプログラム、子プロセス、またはスレッドに渡して、同じ API リソースと設定を使用して個別にログインできます。同じインスタ

ンス・ハンドルを使用しているすべてのプログラム、プロセス、またはスレッドが API の終了前に必ずログアウトしていることを確認してください。

注： スレッドは別のスレッドのネットワーク・ステータス情報への上書きを防止するために、固有のインスタンス・ハンドル(phInstance)を必要とすることがあります。

C のメイン API のコンテキスト・ハンドル

コンテキスト・ハンドルは、システムへのユーザーによる単一の有効なログインを示します。**EssLogin**()の呼出しが正常終了すると、コンテキスト・ハンドルが戻されます。それを、引数としてコンテキスト・ハンドルを必要とする他の API 呼出しに渡すことができます:

- **アプリケーションでのコンテキスト・ハンドルの使用**- コンテキスト・ハンドルは、C プログラムではタイプ **ESS_HCTX_T** として定義されます。一般に、ユーザーがそのサーバーにログ・インしている間(つまり、**EssLogout**()の呼出しが成功するまで)、コンテキスト・ハンドルは有効です。ただし、サーバー・シャットダウンなどのような場合、コンテキスト・ハンドルが無効になる場合があります。そのため、プログラムは、セッションの間にユーザーが再びログ・インできる方法を(たとえばメニュー・オプションまたはファンクション・キーを通して)提供する必要があります。

  注： コンテキスト・ハンドルは、API のインスタンスに固有であり、適切なインスタンスのリソースおよび設定を暗黙的に参照します。

- **複数のコンテキスト・ハンドル**- API プログラムの単一のインスタンスは、1 つ以上の **Essbase** サーバー上で同じユーザー名または異なるユーザー名を使用して、EssLogin()を複数回呼び出すことができます。**EssLogin**()への呼出しはそれぞれ、独自のコンテキスト・ハンドルを戻します。また、プログラムは、戻された各コンテキスト・ハンドルを追跡する必要があります。使用している 1 つのクライアント・アプリケーション当たり 255 個までのコンテキスト・ハンドルを同時に持つことができます。ただし、プログラムが単一のサーバー上でその処理をすべて実行する場合、コンテキスト・ハンドルを 1 つのみ使用して、必要に応じて異なるアプリケーションやデータベースの間で切り替える方が一般に簡単です。このとき、**EssSetActive**()関数または **EssAutoLogin**()関数のいずれかを使用します。

- **コンテキスト・ハンドルの共有**- 一般に、複数のプログラム、プロセスまたはスレッドの間でコンテキスト・ハンドルを共有することは、その使用が排他的であることが保証される場合を除いて、望ましくありません。同じインスタンス・ハンドルを使用し、各プロセスまたはスレッドに別々にログ・インする方がよい方法です。Essbase では、同じサーバー上で同じユーザー名を使用する複数のログインは、そのサーバー上のポートを 1 つのみ使用します。

- **ローカルのコンテキスト・ハンドル** - (クライアント上の)ローカルのオブジェクトおよびファイルに対する操作は、ローカルのコンテキスト・ハンドルを使用できます(ローカル・コンテキスト・ハンドルの使用を参照)。

  ローカル・コンテキストも参照してください。

C のメイン API ファイル・オブジェクト

Essbase オブジェクトとは、データベース・アウトライン、計算スクリプトまたは他のデータのように、Essbase が使用する単なるファイルのことです(8×3 英数字フォーマット)。Essbase にはオブジェクト・システムがあり、これにより、名前、ファイル・タイプおよび関連付けられているアプリケーションやデータベースのみで、API を介してファイルを参照できます。これにより、基本となるファイル・システムとは無関係にオブジェクトを操作できます(ファイル・システムは、Essbase の異なるプラットフォームおよび実装の間で異なる場合があります)。

オブジェクトは、いかなる Essbase サーバーまたはクライアント上にも存在できます。また、その間でのコピーも可能です。サーバーのロック・メカニズムが、オブジェクトへのアクセスを制御し、十分な権限を持ったユーザーは、(**EssGetObject()** 関数を使用して)サーバー・オブジェクトをロックし、クライアントにそれらをコピーできます。また、(**EssPutObject()**関数を使用して)そのオブジェクトを編集したり、サーバーへ戻すことができます。サーバー・オブジェクトは、読取り専用のアクセスにはロックなしに開くことができますが、サーバーに戻して保存することはできません。ユーザーは、個人的利用のためにクライアント・ワークステーションでオブジェクトを作成または編集したり、他のユーザーが共有するためにサーバーに保存できます。

オブジェクトへのアクセス

API を通してオブジェクトにアクセスするとき、オブジェクト名は、オブジェクトのファイル名(拡張子なし)を示します。オブジェクト・タイプは、**ESS_OBJTYPE_xxx** の形式で API ヘッダー・ファイルで宣言されます(ここで、xxx は **ESS_OBJTYPE_REPORT** でのように、特定のタイプを示します)。ほとんどのオブジェクトは、アプリケーションおよびデータベースと関連付けられています。しかし、計算スクリプトやルール・ファイルなどのオブジェクトは、アプリケーション・レベルで保管したり、アプリケーション内のデータベースで使用できます。

データベース・アウトライン・ファイルは、他のオブジェクトとは異なり、API を使用しての削除、名前変更、コピーおよび保存を実行できません。

サーバー・オブジェクト・ファイルは、対応するアプリケーションまたはデータベースのサブディレクトリに物理的に位置します。ただし、サーバー・オブジェクト・ファイルを直接操作することは、通常望ましくありません。必ず、適切な API 関数を使用して、ファイルをローカルにコピーしてください。

クライアント・オブジェクト・ファイルも、デフォルトでは、**ESS_INIT_T** の LocalPath 設定が指定するディレクトリのアプリケーションおよびデータベースのサブディレクトリに保管されています。これらのファイルは自由に操作や編集ができます。ただし、クライアントで編集しているサーバー・オブジェクトをロックおよびロック解除するときにプログラムの行儀がよい(必ず、編集前にオブジェクトをロックし、変更が保存されていてもいなくても、後でロックを解除する)ことを確認してください。

アプリケーションおよびデータベースを NULL に設定することにより、クライアント・オブジェクト・システムをバイパスし、ファイル・システムに直接行くことができます。これによりオブジェクト・フィールドがパス全体になります。

ローカル・コンテキスト

API によってクライアント・マシン上のファイル・オブジェクトにアクセスする場合、使用する API オブジェクト関数のためにローカル・コンテキスト・ハンドルを作成する必要があります。ローカル・コンテキストを作成するには、`EssCreateLocalContext` 関数を使用します。この関数はコンテキスト・ハンドルを戻します。このハンドルは、ログイン・コンテキスト・ハンドルのかわりに、オブジェクト API 関数のいずれにも渡すことができます。また、サーバーではなくローカルのクライアント・オブジェクト・システム上で、要求された操作を API に実行させます。プログラムが最初に API を初期化した直後に、ローカル・コンテキストを 1 度作成する必要があるだけです。

ローカル・コンテキストを作成する場合、API を終了する前に `EssDeleteLocalContext` 関数を呼び出し、プログラムをクリーン・アップすることが必要です。

C プログラムにおけるメモリーの使用

プログラムはすべて、なんらかの形のメモリー割当てを行います。EssbaseAPI は、内部でメモリーを割り当てます。メモリーの一部は、呼出し元プログラムにポインタの形で戻されます。呼出し元プログラムも、メモリーを割り当てられ、メモリーは API へのポインタとして渡されます。異なるメモリー管理スキーム間の競合を避けるために、API では、アプリケーションでのメモリー管理を統合するために 2 つのメカニズムを提供します:

- アプリケーションで API のメモリー管理の仕組みを使用します
- アプリケーションのメモリー管理の仕組みを内部的に使用する API をカスタマイズします

C の API のメモリー管理スキームの使用

API は、メモリー管理関数 **EssAlloc()**、**EssRealloc()**および **EssFree()**を提供します。これらの関数(およびすべての内部 AP メモリー割当て)は、`ESS_INIT_T` 初期化構造体の `AllocFunc`、`ReallocFunc` および **FreeFunc** フィールドがポイントするメモリー割当てルーチンを呼び出します。これらのフィールドへ NULL を渡すと、API に付属のデフォルトの割当てルーチンを使用します。これは、対象プラットフォームに適したネイティブのメモリー・アプリケーション・ルーチンを使用します。

すべてのプットフォームによって呼び出されたネイティブのメモリー割当てルーチンは、C 標準ライブラリ呼出し **malloc()**、**realloc()**および **free()**を呼び出します。C 標準ライブラリ呼出しは、アウトライン API の操作に対応します。それは、通常の使用の間にメモリーの小さな割当てを多く使用します。**GlobalRealloc()**は、新規のバッファ領域を NULL へ初期化しますが、**realloc()**は初期化しません。

注： Intel X86 ベースの Microsoft Windows プラットフォーム用コンパイラを使用している場合に、API では大きいメモリー・モデルが排他的に使用されることにご注意ください。

メモリー管理スキームのカスタマイズ

API のメモリー管理関数を呼び出さない場合、または、同じ割当てスキームをアプリケーション全体に一貫して使用する場合、API が使用するメモリー管理関数の集合を独自に定義できます。これには、メモリーの割当て、再割当ておよび解放を行うカスタム関数を書き、その関数を API が利用できるようにします。通常、これらの関数は、ご使用のアプリケーション内で使用される、対応するメモリー管理関数を内部で呼び出します。

C プログラムでのカスタム・メモリー管理関数の定義

プログラムで独自のカスタム・メモリー管理関数を定義するには、それらの関数を書き、API 初期化構造体の `Allocfunc`、`ReallocFunc` および `FreeFunc` フィールドを設定して、カスタム関数をポイントしてから、**EssInit()**を呼び出します。これらの関数およびその引数には希望の名前を使用できますが、それらを宣言するには、次のフォームを使用する必要があります:

```
   ESS_FUNC_M CustomAlloc  (ESS_SIZE_T BufSize, ESS_PPVOID_T ppBuffer);
ESS_FUNC_M CustomRealloc (ESS_SIZE_T BufSize, ESS_PPVOID_T ppBuffer);
ESS_FUNC_M CustomFree   (ESS_PVOID_T pBuffer);
```

このコードでは、フィールドは次のように定義されています:

- `BufSize` 引数は、割当てまたは再割当てを行うメモリー・バッファの最小サイズです。
- `ppBuffer` 引数は、割当てまたは再割当てが行われたバッファのアドレスを受け取るメモリー・ポインタのアドレスです。
- `pBuffer` 引数は、解放するメモリー・バッファのアドレスです。これらの関数は、正常終了した場合は 0 を、失敗した場合は 0 でない値を戻します。

その後、これらの 3 つの関数へのポインタを、初期化構造体の `AllocFunc`、`ReallocFunc` および `FreeFunc` フィールドに割り当てます。次にこれを **EssInit()**関数に渡します(94 ページの「C のメイン API の初期化」を参照)。

注： 独自のカスタム・メモリー管理関数を定義するには、構造体の 3 つのすべてのフィールドに対して関数を作成し、割り当てる必要があります。

自身のカスタム・メモリー管理関数を定義した後は、そのアプリケーション内でデフォルトの API メモリー管理は使用できません。コード内からの Essbase メモリー管理 API 関数、**EssAlloc()**、**EssRealloc()**および **EssFree()**の呼出しにより、ユーザーが定義した同等のカスタム関数が自動的に呼び出されるからです。ただし、同時に API を使用する他のアプリケーションには影響ありません。**EssInit()**を呼び出す各アプリケーションは、自身のカスタム関数を定義するか、デフォルトの関数を使用するかを独立して選択できます。

注意: カスタム・メッセージ関数の内部から Essbase API 関数を呼び出そうとしないでください。ただし、メモリー管理 API 関数 **EssAlloc()**、**EssRealloc()**および **EssFree()**は例外です。

C のメイン API メッセージ処理

プログラムが API を呼び出すと、システム・メッセージおよびエラー・メッセージが生成されます。これらのメッセージの一部は Essbase サーバーによって戻され、その他は API 内部に渡されます。プログラムはなんらかの方法でこれらのメッセージを処理する必要があり、処理中の操作を中止させるエラーがある場合は、ユーザーに通知する必要があります。

この項では、API のメッセージ・プロセスの仕組みと、C プログラムの開発でカスタム・メッセージ・プロセスを実装する方法について説明します:

- 90 ページの「Essbase C メイン API のメッセージの処理方法」
- 90 ページの「C プログラムのカスタム・メッセージ関数の定義」

Essbase C メイン API のメッセージの処理方法

Essbase では次のメッセージ・レベルがサポートされています:

- 通知メッセージ(通知のみ)
- 警告メッセージ(操作は続行)
- エラー・メッセージ(操作は中断)
- 深刻なエラー(操作が中止され、システムが不安定な状態になります)
- 致命的なエラー(操作が中断され、システムが停止します)

プログラムが、Essbase API のデフォルトのメッセージ処理を使用する場合、エラーまたはエラー以上のレベルのメッセージ(深刻または致命的)は、すべて現行アプリケーションの画面に表示されます。

C プログラムのカスタム・メッセージ関数の定義

C API では、API によって処理される前に、エラー・メッセージをトラップするのに使用できる、カスタム・メッセージ処理関数を提供できます。カスタム・メッセージ処理関数をコーディングして、特定のエラー状態をトラップしたり、プログラム全体にわたりすべてのユーザー・メッセージを同じように処理および表示できます。カスタム・メッセージ関数を提供しない場合は、メッセージ処理はすべて、API のデフォルトのメッセージ・ハンドラによって処理されます。

プログラムでカスタム・メッセージ関数を定義するには、最初に関数を作成し、**EssInit()**呼出しの前に、API 初期化構造体のフィールド `MessageFunc` がカスタム関数を参照するように設定します。

カスタム・メッセージ処理関数のコーディング

カスタム・メッセージ処理関数とその引数には任意の名前を付けられますが、宣言する場合には次の形式を使用する必要があります:

```
   ESS_FUNC_M    CustomMessage (
ESS_PVOID_T   UserContext,   /* user context pointer */
```

```
ESS_LONG_T    MessageNumber,  /* Essbase message number */
ESS_USHORT_ T  Level,      /* message level */
ESS_STR_T     LogString,    /* message log string */
ESS_STR_T     MessageString   /* message string */
);
```

このコードでは、フィールドは次のように定義されています:

- UserContext 引数は、API 初期化の間に初期化構造体の UserContext フィールドで **EssInit()**関数へ渡されたポインタのコピーです(94 ページの「C のメイン API の初期化」を参照)。このポインタを使用して、カスタム・メッセージ処理の間に必要な、アプリケーション特有のコンテキスト情報を含めることができますが、通常、このポインタは、プログラムの状態情報を含む構造体を渡すために使用されます。
- MessageNumber 引数は、特定のエラー状態によって戻されたメッセージをトラップするために使用します(各エラー・メッセージ・コードは、ヘッダー・ファイル(esserror.h)で定義されています)。
- Level 引数は、メッセージの種類(情報、警告、エラー)を表すメッセージ・レベルに基づいてメッセージをトラップする場合に使用します。
- LogString 引数は、サーバー・ログ項目情報を文字列として受領します。次のフォームの文字列を渡します:

  ```
  [Date & Time] Server/Application/Database/Username/Thread/Message#
  ```

  例:

  ```
  [Fri Feb 04 11:51:18 1994]Elm/Sample/Basic/Admin//1012550
  ```

- MessageString 引数は、文字列としてメッセージ・テキストを含んでいます。次のようなメッセージ・テキストを渡します:

  ```
  Total Calc Elapsed Time : [46] seconds
  ```

- デフォルトの API メッセージ・ハンドラでは、ログ文字列とメッセージ文字列が連続して表示されます(メッセージ・ダイアログ内に表示されるか、または stdout ストリームに書き込まれます)。例:

  ```
  [Fri Feb 04 11:51:18 1994]Elm/Sample/Basic/Admin//1012550 Total Calc Elapsed
  Time : [46] seconds
  ```

関数をポイントするように MessageFunc フィールドを設定

カスタム・メッセージ関数へのポインタは、**EssInit()**関数に渡される初期化構造体の MessageFunc フィールドに割り当てる必要があります(94 ページの「C のメイン API の初期化」を参照)。

カスタム関数を使用したメッセージ処理のコントロール

カスタム・メッセージ関数が呼び出されてから、Essbase サーバーがメッセージを戻すか、EssbaseAPI がエラーを戻します。関数が呼び出されると、渡された引数には、そのメッセージのメッセージ番号、メッセージ・レベル、ログ文字列およびエラー文字列が含まれています。各メッセージについて、関数は、これらの引数値を使用して、メッセージを処理するか、無視するか、またはデフォルト処理のために API にそれを戻すかを決めます:

- **API に対する戻りコードの意味**- 戻り値が 0 の場合、関数がメッセージを正常に処理し、それ以上のアクションを API が実行する必要はないことを示します。戻りコードが 0 でない場合、メッセージは、さらなる処理および表示のために、デフォルトの API メッセージ処理関数に渡されます。プログラムにメッセージを無視させるには、カスタム・メッセージ関数から 0 を戻します。

  注： メッセージを処理し終えると、API は、自動的にログおよびメッセージ文字列を解放します。コード内でそれらを解放しようとしないでください。

- **関数がどの戻りコードを生成しなければならないかの決定**- どの戻りコードを生成するか決めるには、カスタム・メッセージ関数をコーディングして、`MessageNumber` 引数または `Level` 引数、あるいはその両方を確認します。たとえば、プログラムは、すべての情報メッセージ、またおそらくはすべての警告メッセージを無視する場合があります(これをユーザー定義可能な設定にできます)。それには、ESSAPI.H に定義された適切な定数(たとえば `ESS_LEVEL_WARNING`)に対して **Level** 引数をテストし、値が必要な値以下の場合に 0 を戻します。その他のメッセージの場合、関数はこれらを内部で処理して 0 の値を戻すか、0 以外の値を戻して、これらのメッセージがデフォルトの API メッセージ・ハンドラで処理されるようにします。

  注： カスタム・メッセージ関数から Essbase API 関数を呼び出さないでください。ただし、メモリー管理 API 関数 **EssAlloc()**、**EssRealloc()**および **EssFree()**を除きます。

  独自のカスタム・メッセージ処理関数を定義しても、同時に API を使用する他のアプリケーションには影響しません。**EssInit()**を呼び出す各アプリケーションでは、カスタム・メッセージ関数を作成するか、またはデフォルトのメッセージ・ハンドラを使用するかを独自に選択できます。

ネットワーク・プロトコルの選択

Essbase は異なる Essbase ネットワーク・ドライバを提供して、複数のネットワーク・プロトコルとネットワーク・ベンダー実装をサポートします。インストールが必要なドライバは、ハードウェア、オペレーティング・システム、クライアント・マシンおよび接続する Essbase サーバー・マシンのネットワーク・プラットフォームによって異なります。

必要なネットワーク構成を決定し、正しいドライバ・ファイルをインストールしてください。

C のメイン API 関数の呼出し

この項では、インスタンス・ハンドルとコンテキスト・ハンドルを使用して戻りコードを戻す API 関数呼び出しについて説明します。

関数の宣言

API は ESS_FUNC_M マクロを使用して C の API 関数を宣言します。これによって、サポートしているすべてのプラットフォームに対して符号なしのロング・タイプとして宣言することになります。カスタム・メモリー管理やメッセージ処理関数など、API に渡すカスタム関数の宣言にもこのマクロを使用する必要があります。

インスタンス・ハンドルまたはコンテキスト・ハンドルの提供

EssInit()への最初の呼出しで戻されたインスタンス・ハンドルを、**EssLogin()**または **EssTerm()**の呼出しに渡す必要があります。**EssLogin()**で戻されたコンテキスト・ハンドルは、特定のログインに関連付けられたすべての関数呼出しに渡す必要があります。

戻りコードの処理

すべての Essbase API 関数では、ESS_STS_T のタイプのステータス・コードが戻されます。0 の戻りコードは関数が正常に実行されたことを、0 以外の値はエラーの状態を示します。エラーの戻り定数をすべて網羅したリストは、ヘッダー・ファイル `esserror.h` に含まれています。対応するメッセージ・テキストは `messages.txt` に含まれています。

注： あらゆる Essbase API 関数の戻りコードを常に確認する必要があります。戻りコードが 0 以外の値だった場合、関数から戻されるポインタや値は未定義となります。

内部メッセージ処理

Essbase ではカスタム以外のメッセージ処理用に、内部メッセージ処理関数を使用しています。32 ビットの Windows システムでエラー・イベントが発生すると、エラー・メッセージが生成されます。

C のメイン API タスクの一般的な順序

API では、プログラムによって特定の関数を他の関数より先に呼び出す必要があります。基本的な順序のルールは次のとおりです:

- プログラムでは、他の API 関数を呼び出す前に **EssInit()**を呼び出すことが必要です。
- プログラムは、コンテキスト・ハンドル引数を必要とする API 関数(ほとんどの API 関数)より前に、**EssLogin()**または **EssAutoLogin()**を呼び出す必要があります。さらに、使用する API オブジェクト関数のローカル・コンテキストを

作成する場合は、コンテキスト・ハンドル引数を必要とする API 関数の前に、**EssCreateLocalContext()**を呼び出す必要があります。

- 一部の API 関数では、アクティブなアプリケーションとデータベースの設定が必要です。プログラムで呼出し前に **EssSetActive()**または **EssAutoLogin()**を呼び出すことで、この種の設定が可能になります。
- C プログラムでは、カスタム・メッセージ処理関数から呼び出せるのはメモリー管理関数のみです。
- また、C プログラムでは、カスタム・メモリー管理関数からいずれの API 関数も呼び出せません。
- プログラムでコンテキスト・ハンドルに対して **EssLogout()**を呼び出した後は、ハンドルを API 関数へ渡さないでください。
- プログラムで **EssTerm()**を呼び出した後は、いずれの API 関数も呼び出さないでください。

以下は、単純な API アプリケーションの通常の操作順序です:

1. **ESS_INIT_T** 構造体を作成し初期化します。
2. **EssInit()**を呼び出して API を初期化します。
3. ローカルの静的な構造体またはグローバル構造体を割り当てます。
4. **EssLogin()**または **EssAutoLogin()**を呼び出して必要なサーバーにログインします。
5. **EssSetActive()**または **EssAutoLogin()**を呼び出してアクティブなアプリケーションおよびデータベースを選択します。
6. **EssReport()**や関連する関数を呼び出してデータを取得またはロックします。
7. **EssUpdate()**や関連する関数を呼び出してデータを更新します。
8. **EssCalc()**や関連する関数を呼び出してデータベースを再計算します。
9. **EssReport()**や関連する関数を呼び出してデータのレポートを作成します。
10. **EssLogout()**を呼び出してサーバーからログアウトします。
11. ローカルの静的な構造体またはグローバル構造体を解放します。
12. **EssTerm()**を呼び出して、API を終了します。

C のメイン API の初期化

プログラムは他の Essbase API 関数を呼び出す前に、**EssInit()**を呼び出して、API の初期化を行う必要があります。**EssInit()**ではすべての内部 API 変数を初期化し、プログラム要件に合わせて API のカスタマイズを行うことができます。

呼出しプログラムは **EssInit()**関数に初期化構造体を渡す必要があります。この構造体は ESSAPI.H で148 ページの「ESS_INIT_T」タイプとして定義されます。API のカスタマイズに使用される一連のフィールドが含まれており、特定の API デフォルトを設定します。**EssInit()**を呼び出す前に、この構造体のインスタンスを宣言して、該当するフィールドを初期化する必要があります。

EssInit()関数はインスタンス・ハンドルを戻します。インスタンス・ハンドルは、API ログイン関数へ引数として渡す必要があります。

初期化構造体の宣言

EssInit()に渡される初期化構造体は通常、呼出し関数内でローカル(すなわちスタック)として宣言されます。これは、一旦 **EssInit()**に渡されたら通常は必要ないからです。あるいは、**EssInit()**を呼び出す前に構造体を割り当てて、戻ってきたら解放することもできます。

初期化呼び出しで初期化構造体がカスタム・メモリー管理関数を参照する場合、正しいメモリー割当ての仕組みによって構造体が解放されるプログラムを作成してください。

初期化構造体のフィールドに 0 または NULL ポインタが設定されていると、フィールドでは API 内部のデフォルト値が使用されます。

フィールドを設定して API 関数を呼び出す前に、すべての構造体を消去(0 に設定)することをお薦めします。

Essbase サーバーへのログイン

一般に、**EssInit()**を呼び出した後にプログラムが最初に実行すべきことは、ユーザーにサーバー名、ユーザー名、パスワード(または定義済のデフォルトを使用)の入力を求め、**EssLogin()**を呼び出してサーバーにログインすることです。または、カプセル化されたログイン関数 **EssAutoLogin()**を使用します。この呼出しが正常に終了した場合、戻されるコンテキスト・ハンドルは保管され、すべての後続の API 呼出しに対して使用されます。

アクティブなアプリケーションとデータベースの選択

コンテキスト・ハンドルに加えて、ログイン関数はログインしたユーザーがアクセスできるアプリケーションとデータベースのリストも戻します(プログラムは **EssListDatabases()**関数を呼び出していつでもこのリストを取得できます)。プログラムでは、**EssSetActive()**関数を呼び出して、ユーザーが特定のアプリケーションおよびデータベースを選択できます。

ログインに **EssAutoLogin()**を使用する場合、アクティブなアプリケーションとデータベースをオプションで設定できます。

Essbase アプリケーション(すでにロード済かどうかに関係なく)の情報を取得するには、**EssGetApplicationState()**または **EssGetApplicationInfo()**関数を呼び出します。特定のデータベースの情報を取得するには、**EssGetDatabaseState()**または **EssGetDatabaseInfo()**関数を呼び出します。これらの関数は、アクティブなアプリケーションおよびデータベースを設定する前に呼び出せます。

データの取得と更新

データの取得

レポートまたはその後の更新のために Essbase データベースからデータを取得するには、プログラムでレポート指定を使用する必要があります。レポート指定は、単一のテキスト文字列(長さが 32KB 未満の場合)、一連のテキスト文字列、またはファイルの形式を使用できます。レポート・ファイルはクライアント・マシンまたは Essbase サーバー上に配置できます。

- **単一文字列としてレポート指定を送信**- 単一文字列としてレポート指定を送信するには、プログラムに **EssReport()**を呼び出させて、長さが 32KB 以下のレポート文字列全体を引数として渡します。**EssReport()**に対する呼出しで `Output` フラグが TRUE に設定されている場合、プログラムは、NULL 文字列が戻されるまで **EssGetString()**を繰り返し呼び出し、戻されたレポート・データを読み出す必要もあります。その後、戻されたデータは、必要に応じて、表示、ファイルへの書込みまたは印刷が可能です。
- **一連の文字列としてレポート指定を送信** - 一連の文字列としてレポート指定を送信するには、最初に **EssBeginReport()**を呼び出します。次に、レポート指定の各文字列を送信するために **EssSendString()**を繰り返し呼び出します(Windows では、各文字列の長さが 32KB を超えないように注意してください)。最後に、**EssEndReport()**を呼び出して、レポート指定を終了します。**EssBeginReport()**に対する呼出しにおいて、`Output` フラグが TRUE に設定されている場合。プログラムは、NULL 文字列が戻されるまで **EssGetString()**を繰り返し呼び出すことにより、戻されたレポート・データを読み出す必要があります。その後、戻されたデータは、必要に応じて、表示、ファイルへの書込みまたは印刷が可能です。
- **レポート指定としてファイルを送信**- レポート指定としてファイルを送信するには、**EssReportFile()**関数を使用して、レポート・ファイル名を渡します。**EssReportFile()**に対する呼出しにおいて、`Output` フラグが TRUE に設定されている場合、プログラムは、NULL 文字列が戻されるまで **EssGetString()**を繰り返し読み出す必要があります。その後、戻されたデータは、必要に応じて、表示、ファイルへの書込みまたは印刷が可能です。

データの更新

データベース内のデータを更新するには、最初に更新対象となるデータベース内のブロックをロックしてください。

➤ データベースのブロックをロックするには、次のいずれかの方法を選択します:

- `Output` フラグを TRUE、`Lock` フラグを TRUE に設定して、前述のようにレポート指定を送信します。このレポートによるデータ出力を変更し、更新としてデータベースに送信できます。
- または、ロードの準備ができた新規データまたは変更データがある場合、プログラムは最初にそのデータをレポート指定として使用することで、適切なレポート関数を呼び出す時点で `Output` フラグを FALSE、`Lock` フラグを TRUE に設定して、データ・ブロックをロックできます。

データベースは、単一文字列、一連の文字列またはファイルのいずれかから更新できます。更新データ・ファイルはクライアント・マシンまたは Essbase サーバー上で存在できます:

- **更新データを単一文字列として送信** - 更新を単一文字列として送信するには、**EssUpdate()**)を呼び出し、引数として文字列全体を渡します。(MS-Windows では、文字列の長さが 32KB を越えないように注意してください)。データベースが更新されるように、**EssUpdate()**に対する呼出しにおいては `Store` フラグを TRUE に設定します。`Unlock` フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックのロックが解除され、他のユーザーがそれらのブロックを更新できます。
- **一連の文字列として更新データを送信** - 一連の文字列として更新データを送信するには、最初に **EssBeginUpdate()**を呼び出します。次に、全データを送信するために **EssSendString()**を繰り返し呼び出します(MS-Windows では、各文字列の長さが 32KB を越えないように注意してください)。最後に、**EssEndUpdate()**を呼び出して、更新を終了します。データベースが更新されるように、**EssUpdate()**に対する呼出しにおいては `Store` フラグを TRUE に設定します。`Unlock` フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックもロックが解除されます。
- **更新データとしてファイルを送信**- ファイルとして更新を送信するには、**EssUpdateFile()**関数を使用して、データ・ファイル名を渡します。データベースが更新されるように、**EssUpdate()**に対する呼出しにおいては `Store` フラグを TRUE に設定します。`Unlock` フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックもロックが解除されます。

データベースの再計算

データベースのデータを更新した後は、連結した合計が正しくなるように再計算する必要があります。データベースを再計算するためには、デフォルト計算を実行、または特定の計算スクリプトを送信できます。また、計算スクリプトをデフォルトの計算スクリプトとして設定できます。計算スクリプトは、単一文字列、連続文字列またはファイルとして送信できます。計算スクリプト・ファイルは、クライアント・マシンまたは Essbase サーバーのいずれにあってもかまいません。

単一文字列としての計算スクリプトの送信

単一文字列として計算スクリプトを送信するためには、**EssCalc()**を呼び出し、引数として文字列全体を渡します(MS Windows では、文字列の長さが 32KB を超えないように注意してください)。計算スクリプトが実行されるように、**EssCalc()**への呼出しでは、`Calculate` フラグを TRUE に設定します。その後、一定の間隔で計算の進行状況を確認する必要があります。

連続文字列としての計算スクリプトの送信

連続文字列として計算スクリプトを送信するためには、最初に **EssBeginCalc()**を呼び出し、その後、計算スクリプトの文字列をすべて送信するために **EssSendString()**を繰り返し呼び出します(MS Windows では、各文字列の長さが 32KB を超えない

ように注意してください)。最後に、**EssEndCalc()**を呼び出して、スクリプトを終了します。データベースが再計算されるように、**EssBeginCalc()**への呼出しでは `Calculate` フラグを TRUE に設定します。その後、一定の間隔で計算の進行状況を確認する必要があります(「計算の進行状況のチェック」を参照)。

ファイルとしての計算スクリプトの送信

ファイルとして計算スクリプトを送信するためには、**EssCalcFile()**関数を使用して、計算スクリプトのファイル名を渡します。データベースが再計算されるように、**EssCalcFile()**への呼出しにおいて `Calculate` フラグを TRUE に設定します。その後、一定の間隔で計算の進行状況を確認する必要があります(「計算の進行状況のチェック」を参照)。

デフォルトの計算スクリプトの使用

現在のデフォルトの計算スクリプトを使用して、データベースを再計算するためには、**EssDefaultCalc()**関数を使用します。データベースにデフォルトの計算スクリプトを設定するためには、**EssSetDefaultCalc()**を使用して、単一文字列として計算スクリプトを渡します。ファイルからデフォルトの計算スクリプトを設定するためには、**EssSetDefaultCalcFile()**関数を使用して、計算スクリプト・ファイル名を渡します。**EssGetProcessState()**を使用して、計算がいつ終了したか判断します(「計算の進行状況の確認」を参照)。

計算の進行状況の確認

データベースの計算が始まったら、一定の間隔(推奨は 5 秒)で、**EssGetProcessState()**関数を呼び出すことにより、計算の進行状況を確認します。この関数は、計算状態を示す構造体を戻します。計算が終了したことまたはエラーが発生したことが示されるまで、**EssGetProcessState()**を呼び出します。また、**EssCancelProcess()**関数を使用して、進行中の計算を取り消せます。

注意 計算の進行中は、計算操作が正常に終了または取り消されるまで、同じコンテキスト・ハンドルを使用して、**EssGetProcessState()**または**EssCancelProcess()**以外の API 関数を呼び出そうとしないでください。計算が終了したことを **EssGetProcessState()**が示した後、プログラムは、そのコンテキスト・ハンドルで他の API 操作の実行を続けることができます。

Essbase サーバーからのログアウトと C メイン API の終了

すべてのデータベース操作が完了すると、アプリケーションは **EssLogout()**を呼び出してログアウトします。これによって、データベース内に予約されていた内部リソースが解放され、サーバー上のログイン・ポートも解放されて別のユーザーが使用できるようになります。

アプリケーション・プログラムが終了する際、**EssTerm()**関数を呼び出し、**EssInit()**への元の呼出しから戻されたインスタンス・ハンドルを渡します。これによって、Essbase API が使用するすべてのリソースが解放されます。この関数を呼び出した

後、**EssInit()**を再度呼び出して API を再初期化しないかぎり、これ以上 API 呼出しはできません。

C のメイン API 共通の問題と解決策

Essbase API を使用すると、Essbase 管理サーバーおよび MaxL が使用する関数の多くに無制限にアクセスできます。

この項は、最も一般的な障害を特定して解決するためのクイック・リファレンスです。

問題	解決策
メモリーの割当て時または解放時に、プログラムで保護違反が発生しています。	C プログラムでは、次の点を確認します: ● **EssFree()**関数を使用して API から戻されたメモリーが解放されていることを確認します。 ● API に渡すポインタの宣言された間接レベルを確認します。 ● Bounds Checker(TM)または Purify(TM)などのメモリー・チェッカを使用して、影響のあるモジュールを判別します。 Essbase が使用しないメモリーにアクセスするときにエラーが発生していても、Essbase メモリー管理スキームとユーザーのスキームの間に、干渉がある場合があります。自身のカスタム・メモリー管理関数を定義することを検討できます。

問題	解決策
API 関数を呼び出すと、ユーザーのプログラムは Essbase エラーを生成します。	ほとんどの Essbase エラー・メッセージは説明を必要としません。問題がどこにあるかはかなり明白なはずです。ただし、次に注意すべきよくあるエラーをいくつか示します(%n は、コンテキストに特有の文字列と置換されるメッセージ引数を示します): ● 「NULL の引数(%1)が ESSAPI 関数%2 に渡されました」。このメッセージは、API 関数%2 に渡された 1 つ以上の引数が NULL だったことを示します。%1 は、最初の NULL 引数の数(1 から始まる)を示します。 ● 「ESSAPI 関数%1 の呼出しシーケンスが無効です」。このメッセージは、別の関数呼出しが必要なときに、API 関数(%1)への呼出しを行ったことを示します。たとえば、**EssReport()**などレポート関数を実行した場合は、NULL 文字列が戻されるまで **EssGetString()**を繰り返し呼び出します。または、**EssCalc()**などの計算関数を実行した場合、戻される値が計算が完了したことを示すまで、**EssGetProcessState()**を呼び出すことにより、繰り返し計算状態を確認します。 ● 「ESSAPI 関数%s ではローカル演算はできません」。ローカルのコンテキスト・ハンドルを、それを許可しない関数へ渡しました。ログイン・コンテキスト・ハンドルをそのかわりに使用します。 ● 「メッセージ・データベース%s を開けません」。メッセージ・データベースが、プログラムが動作しているマシン上でアクセスできません。Essbase が予期する場所にメッセージ・データベースがあることを確認してください。Essbase は、最初に `EssInit()` に渡された初期化構造体の **MessagePath** フィールドを調べます。次に、ARBORMSGPATH 環境変数によって指定されたディレクトリ名およびファイル名を調べ、最後に、`$ESSBASEPATH\BIN` ディレクトリを調べます。ここで、$ESSBASEPATH は環境変数です。メッセージ・データベースがこれらのディレクトリのいずれでも利用できない場合、Essbase は実行時にエラー・メッセージを戻します。どの設定を Essbase が使用するか確認し、その後、指定された場所にメッセージ・データベースがあることを確認します。詳細は、第 3 章「Essbase と使用中の製品との統合」に関する説明を参照してください。
プログラムが確実に API 関数から Essbase エラーの戻りコードを受け取っていてもメッセージが表示されないか、または「メッセージ・データベース内のメッセージ #%1 に対するメッセージはありません」というメッセージが生成されます。	内部 API エラーにはメッセージを表示できないものがありますが、これは通常、メッセージが発生した際にユーザーのコンテキスト情報が使用できないためです。この場合には、関数から戻ったエラー・コードのノートを作成し、`messages.txt` のエラー・メッセージのリストを参照して、対応するメッセージ・テキストを見つけます。エラー定数自体は `esserror.h` に含まれています。
API で定義された構造体のフィールドへアクセスした結果、このフィールドの値が誤っているか、数バイトずれているように見えます。	お使いのコンパイラが、構造体のバイト境界が揃っていることをデフォルトで確認するかどうか確認します。その問題がまだ発生する場合は、API ヘッダー・ファイルの最新バージョンでコンパイルしていて、最新の API DLL とリンクしていることを確認します。

7 CのメインAPIの宣言

この章の内容

標準 C 言語の型

次のデータ型は、C プログラミング言語用 Essbase API で定義されています:

- 101 ページの「単純なデータ型(C)」
- 102 ページの「その他のデータ型(C)」
- 102 ページの「ビットマスク・データ型(C)」
- 104 ページの「ポインタ型(C)」
- 105 ページの「その他の型(C)」
- 105 ページの「配列型(C)」
- 106 ページの「API 定義(C)」

単純なデータ型(C)

データ型	Essbase 型
typedef char	ESS_CHAR_T
typedef short	ESS_SHORT_T
typedef long	ESS_LONG_T
typedef unsigned char	ESS_UCHAR_T
typedef unsigned short	ESS_USHORT_T
typedef unsigned long	ESS_ULONG_T
typedef float	ESS_FLOAT_T

データ型	Essbase 型
typedef double	ESS_DOUBLE_T
win32 && _USE_32BIT_TIME_T が定義されている場合: typedef __time32_t それ以外の場合: typedef time_t	ESS_TIME_T *
typedef unsigned short	ESS_DATE_T
win32 && _USE_32BIT_TIME_T が定義されている場合: typedef __time32_t それ以外の場合: typedef time_t	ESS_DATETIME_T *

注： * Visual Studio 2005 以降のコンパイラの場合、C ライブラリのデータ型 `time_t` は、コンパイラ・マクロ `_USE_32BIT_TIME_T` に基づいて `long` または `int64` Windows データ型にできます。Essbase データ型 `ESS_TIME_T` および `ESS_DATETIME_T` は、32 ビット Windows プラットフォームでは `long` です。

その他のデータ型(C)

データ型	Essbase 型	説明
typedef void	*ESS_HCTX_T	API コンテキスト・ハンドル
typedef void	*ESS_HINST_T	API インスタンス・ハンドル
typedef unsigned char	ESS_BOOL_T	boolean
typedef size_t	ESS_SIZE_T	メモリー・ブロックのサイズ
typedef char	*ESS_STR_T	文字列(文字の配列)
typedef void	ESS_VOID_T	void

ビットマスク・データ型(C)

これらのデータ型に対する値は、適切である場合、他の値を提供するために組み合せられたビット値から構成されます。たとえば、データベースへの WRITE アクセスを必要とする呼出し元は、READ および WRITE 権限を持っている必要があります。したがって、**ESS_ACCESS_WRITE** は、**ESS_PRIV_READ** および **ESS__PRIV_WRITE** に対するビット値と等しくなります。同様に、**ESS_OBJTYPE_BACKUP** は、**ESS_OBJTYPE_ASCBACKUP** と **ESS_OBJTYPE_BINBACKUP** との組合せになります。

データ型	Essbase 型	説明
typedef unsigned short	ESS_ ACCESS_T	セキュリティ・アクセス・レベル。可能なビット値は次のとおりです: ● ESS_PRIV_NONE - 0x0000 - 権限なし ● ESS_PRIV_READ - 0x0001 - データの読取り ● ESS_PRIV_WRITE - 0x0002 - データの書込み ● ESS_PRIV_CALC - 0x0004 - データの計算 ● ESS_PRIV_DBLOAD - 0x0010 - データベースのロードおよびアンロード ● ESS_PRIV_DBDESIGN - 0x0020 - データベースの管理 ● ESS_PRIV_DBCREATE - 0x0040 - データベースの作成、削除および編集 ● ESS_PRIV_APPLOAD - 0x0100 - アプリケーションのロードおよびアンロード ● ESS_PRIV_APPDESIGN - 0x0200 - アプリケーションの管理 ● ESS_PRIV_APPCREATE - 0x0400 - アプリケーションの作成、削除および編集 ● ESS_PRIV_USERCREATE - 0x1000 - ユーザーの作成、削除および編集 アクセス・タイプは権限の組合せです。有効な値は次のとおりです: ● ESS_ACCESS_NONE - 0x0000 ● ESS_ACCESS_READ - 0x0111 ● ESS_ACCESS_WRITE - 0x0113 ● ESS_ACCESS_CALC - 0x0117 ● ESS_ACCESS_METAREAD - 0x0118 ● ESS_ACCESS_DBMANAGE - 0x0137(または ESS_ACCESS_DBDESIGN、下位互換性のために維持) ● ESS_ACCESS_DBCREATE - 0x0177 ● ESS_ACCESS_APPDESIGN - 0x0377 ● ESS_ACCESS_APPCREATE - 0x0777 ● ESS_ACCESS_FILTER - 0x0110 ● ESS_ACCESS_DBALL - 0x00ff - データベースへのフル・アクセス ● ESS_ACCESS_APPALL - 0x0fff - アプリケーションおよびデータベースへのフル・アクセス ● ESS_ACCESS_ADMIN - 0xffff - 管理者(無制限アクセス)(または ESS_ACCESS_SUPER、下位互換性のために維持) Oracle Hyperion Shared Services セキュリティの役割のマッピングは次のとおりです: ● ESS_USERPROVROLE_NONE = ESS_ACCESS_NONE = 0x0000 ● ESS_USERPROVROLE_USERCREATE = ESS_PRIV_USERCREATE = 0x1000 **注：** この役割は、Shared Services モードにおいて Essbase によって設定できません。Shared Services でのみ設定できます。 ● ESS_USERPROVROLE_APPCREATE = ESS_PRIV_APPCREATE = 0x0400 ● ESS_USERPROVROLE_APPMANAGER = ESS_ACCESS_APPMANAGE または ESS_ACCESS_APPDESIGN = 0x0377 ● ESS_USERPROVROLE_APPLOAD = ESS_PRIV_APPLOAD = 0x0100 ● ESS_USERPROVROLE_DBFILTER = ESS_ACCESS_FILTER = 0x0110 ● ESS_USERPROVROLE_DBREAD = ESS_ACCESS_READ = 0x0111 ● ESS_USERPROVROLE_DBWRITE = ESS_ACCESS_WRITE = 0x0113

データ型	Essbase 型	説明
		• ESS_USERPROVROLE_DBCALC = ESS_ACCESS_CALC = 0x0117 • ESS_USERPROVROLE_DBMANAGER = ESS_ACCESS_DBMANAGE または ESS_ACCESS_DBDESIGN = 0x0137 • ESS_USERPROVROLE_ADMINISTRATOR = ESS_ACCESS_ADMIN または ESS_ACCESS_SUPER = 0xffff
typedef unsigned long	ESS_OBJTYPE_T	ファイル・オブジェクト・タイプ。 単一オブジェクト・タイプは次のとおりです: • ESS_OBJTYPE_NONE • ESS_OBJTYPE_OUTLINE • ESS_OBJTYPE_CALCSCRIPT • ESS_OBJTYPE_REPORT • ESS_OBJTYPE_RULES • ESS_OBJTYPE_ALIAS • ESS_OBJTYPE_STRUCTURE • ESS_OBJTYPE_ASCBACKUP • ESS_OBJTYPE_BINBACKUP • ESS_OBJTYPE_EXCEL • ESS_OBJTYPE_LOTUS2 (サポート廃止) • ESS_OBJTYPE_LOTUS3 (サポート廃止) • ESS_OBJTYPE_TEXT • ESS_OBJTYPE_PARTITION • ESS_OBJTYPE_LOTUS4 (サポート廃止) • ESS_OBJTYPE_WIZARD • ESS_OBJTYPE_OTL_E • ESS_OBJTYPE_SELECTION • ESS_OBJTYPE_LRO #define ESS_OBJTYPE_MAX 0x08000000 /* maximum single object type value */複合オブジェクト・タイプは、次のとおりです: • ESS_OBJTYPE_BACKUP • ESS_OBJTYPE_WORKSHEET • ESS_OBJTYPE_DATA • ESS_OBJTYPE_ALL

ポインタ型(C)

データ型	Essbase 型	説明
char	*ESS_PCHAR_T	char 型に対するポインタ
unsigned char	*ESS_PUCHAR_T	符合なし char 型に対するポインタ
short	*ESS_PSHORT_T	short 型に対するポインタ

データ型	Essbase 型	説明
unsigned short	*ESS_PUSHORT_T	符号なし short 型に対するポインタ
long	*ESS_PLONG_T	long 型に対するポインタ
unsigned long	*ESS_PULONG_T	符号なし long 型に対するポインタ
double	*ESS_PDOUBLE_T	double 型に対するポインタ
float	*ESS_PFLOAT_T	float 型に対するポインタ
ESS_ACCESS_T	*ESS_PACCESS_T	セキュリティ・アクセス・レベルへのポインタ
ESS_BOOL_T	*ESS_PBOOL_T	boolean 型に対するポインタ
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_HINST_T	*ESS_PHINST_T	API インスタンス・ハンドルへのポインタ
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_SIZE_T	*ESS_PSIZE_T	メモリー・ブロックのサイズへのポインタ
ESS_STR_T	*ESS_PSTR_T	文字列へのポインタ
ESS_VOID_T	*ESS_PVOID_T	void 型に対するポインタ

その他の型(C)

データ型	Essbase 型	説明
typedef long	ESS_STS_T	API 関数からの戻り値
typedef ESS_STS_T	(*ESS_FUNC_T)()	関数へのポインタ

配列型(C)

次の配列のタイプは、適切な長さの文字列を使用して定義されます。たとえば、タイプ **ESS_USERNAME_T** は typedef char **ESS_USERNAME_T** [**ESS_USERNAME_LEN**] として定義されます。

データ型	Essbase 型	説明
typedef char	ESS_USERNAME_T	ユーザー名
typedef char	ESS_PASSWORD_T	パスワード
typedef char	ESS_SVRNAME_T	サーバー名
typedef char	ESS_APPNAME_T	アプリケーション名
typedef char	ESS_DBNAME_T	データベース名

データ型	Essbase 型	説明
typedef char	ESS_OBJNAME_T	オブジェクト名
typedef char	ESS_MBRNAME_T	メンバー名
typedef char	ESS_FTRNAME_T	フィルタ名
typedef char	ESS_ALIASNAME_T	別名テーブルの名前
typedef char	ESS_PATH_T	ファイル・パス名
typedef char	ESS_DESC_T	アプリケーションまたはデータベースの説明

API 定義(C)

Essbase 型	値
#define ESS_TRUE	1
#define ESS_FALSE	0
#define ESS_NULL	NULL
#define ESS_NATIVE_SECURITY	1
#define ESS_SS_SECURITY	2

定数の定義(C)

次の定数は、Essbase API で定義されています:

- 106 ページの「属性定数(C)」
- 109 ページの「次元タグ定数(C)」
- 110 ページの「情報フラグ定数(C)」
- 110 ページの「リスト・オプション定数(C)」
- 111 ページの「文字列の最大長(C)」
- 112 ページの「要求タイプの定数(C)」
- 112 ページの「サイズ・フラグ定数(C)」

属性定数(C)

次の定数は、クエリーの結果、765 ページの「ESS_ATTRIBUTEQUERY_T」構造体のフィールド usInputMemberType と usOutputMemberType へ戻されるメンバーのデータ型を定義します。

値	定義
ESS_BASE_DIMENSION	属性次元以外の次元
ESS_BASE_MEMBER	属性メンバー以外のメンバー
ESS_ATTRIBUTE_DIMENSION	属性次元
ESS_ATTRIBUTE_MEMBER	属性メンバー
ESS_ATTRIBUTED_MEMBER	属性が関連付けられた基本メンバーまたは次元。標準メンバーまたは標準次元とも呼びます。

次の定数は、768 ページの「ESS_MBRINFO_T」構造体の Status フィールドの属性メンバーのステータスを定義するための定数です。

値	定義
ESS_MBRSTS_ATTRIBUTE	属性メンバーのステータス

次の定数は、139 ページの「ESS_DIMENSIONINFO_T」構造体の DimTag フィールドの属性次元タグのタイプを定義するための定数です。

値	定義
ESS_TTYPE_ATTRIBUTE	属性タグ
ESS_TTYPE_ATTRCALC	属性計算タグ。集約のために内部的に使用されます。

次の定数は、124 ページの「ESS_ATTRIBUTEVALUE_T」構造体の usDataType フィールドおよび139 ページの「ESS_DIMENSIONINFO_T」構造体の DimDataType フィールドの属性メンバー・データ型を定義します。

値	定義
ESS_ATTRMBRDT_BOOL	ブール式のデータ型
ESS_ATTRMBRDT_DATETIME	日付時刻データ型
ESS_ATTRMBRDT_DOUBLE	倍精度浮動小数点数データ型
ESS_ATTRMBRDT_STRING	文字列データ型
ESS_ATTRMBRDT_NONE	データ型はありません

次の定数は、765 ページの「ESS_ATTRIBUTEQUERY_T」構造体の usOperation フィールドの属性クエリー演算のタイプを定義するための定数です。

値	定義
ESS_EQ	等しい
ESS_NEQ	等しくない

値	定義
ESS_GT	より大きい
ESS_LT	より小さい
ESS_GTE	以上
ESS_LTE	以下
ESS_TYPEOF	タイプ
ESS_ALL	すべて

表 6　C API 属性の用語

用語	定義
バケットのタイプ	次元を作成する場合、ESS_ATTRMBRDT_DOUBLE 型のゼロレベルの属性メンバーをリレーショナル・ソースのデータ範囲に関連付けることができます。 バケットのタイプは、データ範囲の上限と下限を指定します。 「usBucketingType」を参照してください。
ESS_ATTRIBUTE_DIMENSION ESS_ATTRIBUTE_MEMBER	ESS_ATTRIBUTE_DIMENSION は属性次元です。 ESS_ATTRIBUTE_MEMBER は属性次元のメンバーです。 765 ページの「ESS_ATTRIBUTEQUERY_T」を参照してください。 また、EssCheckAttributes も参照してください。
ESS_ATTRIBUTED_MEMBER	ESS_ATTRIBUTED_MEMBER は(基本次元の)メンバーの 1 つであり、メンバーに関連した属性メンバーがあります。 765 ページの「ESS_ATTRIBUTEQUERY_T」を参照してください。 また、EssCheckAttributes も参照してください。
ESS_BASE_DIMENSION ESS_BASE_MEMBER	ESS_BASE_DIMENSION は標準次元であり、標準次元に関連のある属性次元があります。 ESS_BASE_MEMBER は基本次元のメンバーです。 765 ページの「ESS_ATTRIBUTEQUERY_T」を参照してください。 また、EssCheckAttributes も参照してください。
ESS_STANDARD_DIMENSION ESS_STANDARD_MEMBER	ESS_STANDARD_DIMENSION は属性次元以外のすべての次元です。 ESS_STANDARD_MEMBER は標準次元のメンバーです。 765 ページの「ESS_ATTRIBUTEQUERY_T」を参照してください。 また、EssCheckAttributes も参照してください。

用語	定義
ロング名	ESS_ATTRMBRDT_STRING 型以外のゼロレベルの属性メンバーは、ロング名で一意に識別できます。 ESS_ATTRMBRDT_STRING 型のゼロレベルの属性メンバーは、常に一意であることが必要です。 次の構造体を参照してください: ● 125 ページの「ESS_ATTRSPECS_T」 ● 124 ページの「ESS_ATTRIBUTEINFO_T」 また、次の関数も参照してください: ● EssGetAttributeSpecifications ● EssOtlGetAttributeSpecifications ● EssOtlSetAttributeSpecifications さらに、「属性メンバーの追加に関する注意」も参照してください。
ショート名	ESS_ATTRMBRDT_STRING 型以外のゼロレベルの属性メンバーを、ショート名と呼びます。 ショート名は、ESS_STR_T 型のパラメータとして関数に渡されます。 EssOtlFindAttributeMembers を参照してください。

次元タグ定数(C)

次の定数は、139 ページの「ESS_DIMENSIONINFO_T」構造体の DimTag フィールドで使用される情報フラグを定義するための定数です。

定数	定義
ESS_TTYPE_NONE	次元タイプなし。ESS_DIMENSIONINFO_T の DimTag フィールドの値。
ESS_TTYPE_CCATEGORY	勘定科目: 通貨 ACCOUNTS タグ。ESS_DIMENSIONINFO_T の DimTag フィールドの値
ESS_TTYPE_CNAME	国: 通貨 COUNTRY タグ。ESS_DIMENSIONINFO_T の DimTag フィールドの値
ESS_TTYPE_CTIME	時刻: 通貨 TIME タグ。ESS_DIMENSIONINFO_T の DimTag フィールドの値
ESS_TTYPE_CTYPE	タイプ: 通貨 TYPE タグ。ESS_DIMENSIONINFO_T の DimTag フィールドの値
ESS_TTYPE_CPARTITION	通貨 PARTITION タグ。ESS_DIMENSIONINFO_T の DimTag フィールドの値

暗黙の共有設定(C)

暗黙の共有設定は、EssOtlGetImpliedShare および EssOtlSetImpliedShare 関数を使用して特定のアウトラインに適用できます。

アウトラインが保存され、再構築されるまで、変更は反映されません。

注： 明示的な設定は、アプリケーションを後でコピーする場合には特に有用です。これは、明示的な設定が Essbase.cfg ファイルで設定したアプリケーション名に固有のエントリとは関係なく、アウトラインに付いて回るからです。

設定に指定可能な値:

値	説明
ESS_IMPLIEDSHARE_DEFAULT	EssOtlSetImpliedShare を使用して設定できます。 設定すると、_ON または_OFF に即座に変換されます。 戻された場合: ● アウトラインには暗黙の共有設定はありません ● 暗黙の共有が ON
ESS_IMPLIEDSHARE_DEFAULT_ON	EssOtlGetImpliedShare でのみ戻り値があります。 戻された場合: ● アウトラインが Essbase.cfg 内の暗黙の共有のデフォルト設定を使用します ● Essbase.cfg に暗黙の共有エントリ(ON)が含まれている可能性があります ● Essbase.cfg にはエントリは含まれていません ● 暗黙の共有が ON
ESS_IMPLIEDSHARE_DEFAULT_OFF	EssOtlGetImpliedShare でのみ戻り値があります。 戻された場合: ● アウトラインが Essbase.cfg 内の暗黙の共有のデフォルト設定を使用します ● Essbase.cfg には暗黙の共有エントリ(OFF)が含まれています ● 暗黙の共有は OFF
ESS_IMPLIEDSHARE _FORCE_ON	EssOtlSetImpliedShare を使用して設定できます。 アウトラインを示す明示的な設定では、常に暗黙の共有が ON です。
ESS_IMPLIEDSHARE _FORCE_OFF	EssOtlSetImpliedShare を使用して設定できます。 アウトラインを示す明示的な設定では、常に暗黙の共有が OFF です。

情報フラグ定数(C)

次の定数は、133 ページの「ESS_DBREQINFO_T」構造体の DbReqFlags(データのロード)フィールドで使用される情報フラグを定義するための定数です。

定数	定義
ESS_DBREQFLAG_CALCDEF	DbReqFlags フィールドのデフォルト・フラグ。デフォルトの計算スクリプトで使用されます。値: 0x0000001。
ESS_DBREQFLAG_CALCDSCR	DbReqFlags フィールドのカスタム計算スクリプト・フラグ。カスタム計算スクリプトで使用されます。値: 0x0000002。

リスト・オプション定数(C)

次の定数は、EssListTransactions 関数の ListOption フィールドが使用する要求タイプを定義します。

定数	定義
ESS_LIST_TRANSACTIONS_TOCLIENT	出力を画面に書き込みます。
LIST_TRANSACTIONS_TOFILE	● 出力を CSV ファイルに書き込みます。 ● 出力は ppResults に戻されません。 ● pCount と ppResults は NULL になります。 ● コンテンツはカンマ区切りファイルとして FileName に書き込まれます。 ● 指定したファイル名が存在する場合、コマンドは失敗します。
ESS_LIST_TRANSACTIONS_ FORCETOFILE	● 出力を CSV ファイルに書き込みます。 ● 出力は ppResults に戻されません。 ● pCount と ppResults は NULL になります。 ● コンテンツはカンマ区切りファイルとして FileName に書き込まれます。 ● 指定したファイル名が存在する場合、新しい出力で上書きされます。

文字列の最大長(C)

次の定数は、Essbase API での各種文字列タイプの最大長を定義します。これらの定数はいずれも、最後に NULL 文字を含みます:

定数	定義
ESS_ALIASNAMELEN	別名テーブル名の最大長
ESS_APPNAMELEN	アプリケーション名の最大長
ESS_CRDB_MAXIMUM	通貨データベースの最大次元数
ESS_DBNAMELEN	データベース名の最大長
ESS_DESCLEN	アプリケーションまたはデータベースの記述の最大長
ESS_FTRNAMELEN	フィルタ名の最大長
ESS_LINELEN	レポートの 1 行の最大長
ESS_MBRCOMMENTEXLEN	拡張メンバー・コメントの最大長
ESS_MBRNAMELEN	メンバー名の最大長
ESS_NAMELEN	一般の名前の最大長
ESS_PASSWORDLEN	ユーザー・パスワードの最大長
ESS_PATHLEN	ファイル・パス名の最大長
ESS_OBJNAMELEN	オブジェクト名の最大長
ESS_SVRNAMELEN	サーバー名の最大長
ESS_USERNAMELEN	ユーザーまたはグループ名の最大長

要求タイプの定数(C)

次の定数は、200 ページの「ESS_TRANSACTION_REQSPECIFIC_T」構造体の ucReqType フィールドが使用する要求タイプを定義します。

定数	定義
ESS_TRLOG_CALCSCRIPT_SERVER	計算スクリプト名
ESS_TRLOG_CALCSCRIPT_IMMEDIATE	データなし
ESS_TRLOG_CALCSCRIPT_DEFAULT	データなし
ESS_TRLOG_CALCSCRIPT_SETDEFAULT	データなし
ESS_TRLOG_DATALOAD_SERVER	サーバー側データ・ロード・ファイル
ESS_TRLOG_DATALOAD_IMMEDIATE	データはクライアント側から入力されました
ESS_TRLOG_DATALOAD_SQL	SQL データ・ソース
ESS_TRLOG_DATALOAD_CLEARDB	すべてのデータの消去
ESS_TRLOG_DATALOAD_RESETDB	すべてのデータとアウトラインの消去
ESS_TRLOG_SSUPDATE	スプレッドシートの更新
ESS_TRLOG_DATALOAD_FTP	FTP データ・ソース

サイズ・フラグ定数(C)

次の定数は、ESS_DBSTATE_T 構造体のフィールド MaxMemIndex と134 ページの「ESS_DBSTATE_T」の最大サイズと最小サイズを定義します。

定数	定義
ESS_INDEXCACHEMIN_SIZE	ESS_DBSTATE_T 構造体の MaxMemIndex フィールドの最小インデックス・キャッシュ・サイズ。値: 1048576。最大値は定義されていません。
ESS_INDEXPAGEMAX_SIZE	ESS_DBSTATE_T 構造体の IndexPageSize フィールドの最大インデックス・ページ・サイズ。値: 8192
ESS_INDEXPAGEMIN_SIZE	ESS_DBSTATE_T 構造体の IndexPageSizeMin フィールドの最小インデックス・ページ・サイズ。値: 1024

Unicode モードの定数(C)

次の定数は、Unicode モードのクライアント・プログラムを使用可能にします。これらの定数は **ESS_INIT_T** 構造体の **usApiType** フィールドの有効な値です。**ESS_INIT_T** 構造体は **EssInit**()で使用され、クライアント・プログラムが Unicode モードかどうかを定義します。Unicode モードのクライアント・プログラムのみ UTF-8 でエンコードされてテキストを Essbase サーバーに送信できます。

定数	定義	説明
ESS_API_NONUNICODE	0x0002	プログラムは非 Unicode モードのクライアント・プログラムです。クライアント・プログラムは非 Unicode エンコードの引数を API に渡します。これはデフォルト値です。
ESS_API_UTF8	0x0003	プログラムは Unicode モードのクライアント・プログラムです。クライアント・プログラムは UTF-8 エンコードの引数を API に渡します。

LRO の定数と構造体の定数(C)

次の定数および構造体は、特にリンク・レポート・オブジェクト(LRO)の使用のために定義されています:

LRO の定数(C)

次の定数は、Essbase API の LRO 関数および構造体で使用される様々な値を定義します。

データ型	フィールド	説明
ESS_LRODESCLEN_API	79	オブジェクトの記述の最大長
ESS_LRONOTELEN_API	599	セル・ノートの最大長
ESS_ONAMELEN_API	511	ファイル名とパスからなるオブジェクト名の長さ
ESS_DATESIZE	12	日付文字列のサイズ
ESS_STORE_OBJECT_API	0x0010	サーバーでリンク・オブジェクトを保管するための値
ESS_NOSTORE_OBJECT_API	0x0001	サーバーでリンク・オブジェクトを保管しないための値
ESS_LROTYPE_CELLNOTE_API	0	リンク・オブジェクトがセル・ノートであることを指定する値
ESS_LROTYPE_WINAPP_API	1	リンク・オブジェクトが Windows アプリケーションであることを指定する値
ESS_LROTYPE_URL_API	2	リンク・オブジェクトが URL であることを指定する値

ESS_CELLADDR_API_T

Essbase データベース内のデータ・セルのアドレスに関する情報を含んでいます。Essbase はメンバーの組合せからセル・アドレスを派生させ、アドレスを使用して

データ・セルにリンクされたオブジェクトを追跡します。この構造体のフィールドは API では変更できません。次にフィールドについて説明します:

データ型	フィールド	説明
ESS_ULONG_T	cellOffset	データ・ブロックにおけるセルのオフセット
ESS_SECPART_T	blkOffset	ブロック・オフセット
ESS_SECPART_T	segment	セグメント番号

ESS_LRODESC_API_T

Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを説明する情報を含んでいます。次にフィールドについて説明します:

データ型	フィールド	説明
struct ESS_LRODESC_API_T	next	(next フィールドは内部使用のみです。)
ESS_USHORT_T	usObjType	オブジェクト・タイプ
ESS_USHORT_T	status	カタログ・エントリのステータス
ESS_LROHANDLE_API_T	linkId	LRO のリンク ID
ESS_CHAR_T	userName[ESS_USERNAMELEN]	オブジェクトを最後に変更したユーザー名
ESS_TIME_T	updateDate	オブジェクトが最後に変更された日付
ESS_ACCESS_T	accessLevel	メンバーの組合せのアクセス・レベル
ESS_ULONG_T	memCount	メンバーの組合せに含まれるメンバー数
ESS_PMBRNAME_NONUNI_T	pMemComb	オブジェクトに関連付けられたメンバーの組合せ
ESS_LROINFO_API_T	lroInfo	ユニオンによって関連付けられた LRO 情報構造体
ESS_CHAR_T	note[ESS_LRONOTELEN_API]	ユニオンによって関連付けられたセル・ノート

ESS_LROHANDLE_API_T

リンク・オブジェクトの識別子を提供します。識別子はセル・アドレスと内部オブジェクト・ハンドルから構成されます。この構造体のフィールドにはリンク・オブジェクトに関連する情報が含まれているため、変更しないでください。次にフィールドについて説明します:

データ型	フィールド	説明
ESS_CELLADDR_API_T	cellKey	セル・アドレス
ESS_LONG_T	hObject	内部オブジェクト・ハンドル

ESS_LROINFO_API_T

Essbase データベースのデータ・セルにリンクされている特定のオブジェクトに関する情報を含んでいます。リンク・オブジェクトに関する情報を含んできるため、この構造体のフィールドは変更できません。次にフィールドについて説明します:

データ型	フィールド	説明
ESS_CHAR_T	objName[ESS_ONAMELEN_API]	データ・セルにリンクされたオブジェクトのソース・ファイル名。ESS_ONAMELEN_API はオブジェクト名の最大長を指定します。デフォルト値は 511 です。
ESS_CHAR_T	objDesc[ESSW_LRODESCLEN_API]	データ・セルにリンクされたオブジェクトの説明。ESS_LRODESCLEN_API は説明の最大長を指定します。デフォルト値は 79 です。

パーティションの定数および構造体の定義(C)

ドリルスルーの定数および構造体の定義

これらのトピックでは、特にドリルスルーで使用するために定義されている C メイン API の定数および構造体について説明します:

C のメイン API ドリルスルーの定数および構造体 (essdt.dll)

構造体

文字列の最大長の定数

次の定数は、Essbase API での各種文字列タイプの最大長を定義します。これらの定数はいずれも、最後に NULL 文字を含みます:

定数	定義
ESS_ALIASNAMELEN	別名テーブル名の最大長
ESS_APPNAMELEN	アプリケーション名の最大長
ESS_CRDB_MAXIMUM	通貨データベースの最大次元数
ESS_DBNAMELEN	データベース名の最大長
ESS_DESCLEN	アプリケーションまたはデータベースの記述の最大長
ESS_DESCRIPTION_LEN	ドリルスルーに使用する文字列の最大長(255)

定数	定義
ESS_DTREPORT_NAME	ドリルスルーに使用する文字列の最大長(80)
ESS_FTRNAMELEN	フィルタ名の最大長
ESS_LINELEN	レポートの 1 行の最大長
ESS_MAX_DATALEN	ドリルスルーに使用する文字列の最大長(255)
ESS_MAX_NAME	ドリルスルーに使用する文字列の最大長(30)
ESS_MBRCOMMENTEXLEN	拡張メンバー・コメントの最大長
ESS_MBRNAMELEN	メンバー名の最大長
ESS_NAMELEN	一般の名前の最大長
ESS_PASSWORDLEN	ユーザー・パスワードの最大長
ESS_PATHLEN	ファイル・パス名の最大長
ESS_OBJNAMELEN	オブジェクト名の最大長
ESS_SVRNAMELEN	サーバー名の最大長
ESS_USERNAMELEN	ユーザーまたはグループ名の最大長

ポインタ型

データ型	Essbase 型	説明
char	*ESS_PCHAR_T	char 型に対するポインタ
unsigned char	*ESS_PUCHAR_T	符合なし char 型に対するポインタ
short	*ESS_PSHORT_T	short 型に対するポインタ
unsigned short	*ESS_PUSHORT_T	符号なし short 型に対するポインタ
long	*ESS_PLONG_T	long 型に対するポインタ
unsigned long	*ESS_PULONG_T	符号なし long 型に対するポインタ
double	*ESS_PDOUBLE_T	double 型に対するポインタ
float	*ESS_PFLOAT_T	float 型に対するポインタ
ESS_ACCESS_T	*ESS_PACCESS_T	セキュリティ・アクセス・レベルへのポインタ
ESS_BOOL_T	*ESS_PBOOL_T	boolean 型に対するポインタ
ESS_DTAPIHINST_T	*ESS_PDTAPIHINST_T	ドリルスルー初期化構造体へのポインタ
ESS_DTHINST_T	*ESS_PDTHINST_T	ドリルスルー初期化構造体へのポインタ
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_HINST_T	*ESS_PHINST_T	API インスタンス・ハンドルへのポインタ

データ型	Essbase 型	説明
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_SIZE_T	*ESS_PSIZE_T	メモリー・ブロックのサイズへのポインタ
ESS_STR_T	*ESS_PSTR_T	文字列へのポインタ
ESS_VOID_T	*ESS_PVOID_T	void 型に対するポインタ

C のメイン・ドリルスルーの定数および構造体 (essdtapi.dll)

構造体

- 140 ページの「ESS_DTAPICOLUMN_T」
- 141 ページの「ESS_DTAPIDATA_T」
- 142 ページの「ESS_DTAPIHEADER_T」
- 142 ページの「ESS_DTAPIINFO_T」
- 143 ページの「ESS_DTAPIREPORT_T」

文字列の最大長の定数

次の定数は、Essbase API での各種文字列タイプの最大長を定義します。これらの定数はいずれも、最後に NULL 文字を含みます:

定数	定義
ESS_ALIASNAMELEN	別名テーブル名の最大長
ESS_APPNAMELEN	アプリケーション名の最大長
ESS_CRDB_MAXIMUM	通貨データベースの最大次元数
ESS_DBNAMELEN	データベース名の最大長
ESS_DESCLEN	アプリケーションまたはデータベースの記述の最大長
ESS_DESCRIPTION_LEN	ドリルスルーに使用する文字列の最大長(255)
ESS_DTREPORT_NAME	ドリルスルーに使用する文字列の最大長(80)
ESS_FTRNAMELEN	フィルタ名の最大長
ESS_LINELEN	レポートの 1 行の最大長
ESS_MAX_DATALEN	ドリルスルーに使用する文字列の最大長(255)
ESS_MAX_NAME	ドリルスルーに使用する文字列の最大長(30)
ESS_MBRCOMMENTEXLEN	拡張メンバー・コメントの最大長

定数	定義
ESS_MBRNAMELEN	メンバー名の最大長
ESS_NAMELEN	一般の名前の最大長
ESS_PASSWORDLEN	ユーザー・パスワードの最大長
ESS_PATHLEN	ファイル・パス名の最大長
ESS_OBJNAMELEN	オブジェクト名の最大長
ESS_SVRNAMELEN	サーバー名の最大長
ESS_USERNAMELEN	ユーザーまたはグループ名の最大長

ESS_DTAPIINFO_T の uInputOption に対するドリルスルー接続値

次の定数は、Oracle Essbase Studio へ接続し、ドリルスルーを実行するための入力値を定義します。

定数	定義
ESS_DTAPI_PROMPT_HISNAME	uInputOption フィールドの値。ユーザーは、Essbase Studio へ接続してドリルスルーを実行できます
ESS_DTAPI_PROMPT_LOGIN	uInputOption フィールドの値。Essbase Studio へ接続してドリルスルーを実行するには、パスワードが必要です

ポインタ型

データ型	Essbase 型	説明
char	*ESS_PCHAR_T	char 型に対するポインタ
unsigned char	*ESS_PUCHAR_T	符合なし char 型に対するポインタ
short	*ESS_PSHORT_T	short 型に対するポインタ
unsigned short	*ESS_PUSHORT_T	符号なし short 型に対するポインタ
long	*ESS_PLONG_T	long 型に対するポインタ
unsigned long	*ESS_PULONG_T	符号なし long 型に対するポインタ
double	*ESS_PDOUBLE_T	double 型に対するポインタ
float	*ESS_PFLOAT_T	float 型に対するポインタ
ESS_ACCESS_T	*ESS_PACCESS_T	セキュリティ・アクセス・レベルへのポインタ
ESS_BOOL_T	*ESS_PBOOL_T	boolean 型に対するポインタ
ESS_DTAPIHINST_T	*ESS_PDTAPIHINST_T	ドリルスルー初期化構造体へのポインタ
ESS_DTHINST_T	*ESS_PDTHINST_T	ドリルスルー初期化構造体へのポインタ

データ型	Essbase 型	説明
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_HINST_T	*ESS_PHINST_T	API インスタンス・ハンドルへのポインタ
ESS_HCTX_T	*ESS_PHCTX_T	API コンテキスト・ハンドルへのポインタ
ESS_SIZE_T	*ESS_PSIZE_T	メモリー・ブロックのサイズへのポインタ
ESS_STR_T	*ESS_PSTR_T	文字列へのポインタ
ESS_VOID_T	*ESS_PVOID_T	void 型に対するポインタ

C のメイン API の構造体

「コンテンツ」ペインで C のメイン API 構造体のリストを参照してください。

ESS_APPDB_T

このアプリケーション名とデータベース名の構造体は、アプリケーション名とデータベース名を戻します。フィールドは次のとおりです:

```
   typedef struct  ESS_APPDB_T
{
 ESS_APPNAME_T AppName;
 ESS_DBNAME_T  DbName;
} ESS_APPDB_T, *ESS_PAPPDB_T, **ESS_PPAPPDB_T;
```

データ型	フィールド	説明
ESS_APPNAME_T	AppName	アプリケーション名
ESS_DBNAME_T	DbName	データベース名

ESS_APPINFO_T

このアプリケーション情報構造体は、特定のアプリケーションに関する情報を戻します。この構造体のフィールドは、API を使用して変更できません。変更可能な追加のアプリケーション状態パラメータが含まれている123 ページの「ESS_APPSTATE_T」構造体を参照してください。フィールドは次のとおりです:

注： ロケール固有の拡張アプリケーション情報構造体である122 ページの「ESS_APPINFOEX_T」も参照してください。

```
   typedef struct  ESS_APPINFO_T
{
```

```
 ESS_APPNAME_T    Name;
 ESS_SVRNAME_T    Server;
 ESS_USHORT_T     Status;
 ESS_USHORT_T,    AppType;
 ESS_CHAR_T,     AppLocale, ESS_LOCALESTRING_LENGTH;
 ESS_USHORT_T     nConnects;
 ESS_TIME_T      ElapsedAppTime;
 ESS_USHORT_T     nDbs;
 ESS_DATA_STORAGE_T StorageType;
 ESS_DBNAME_T     DbNames[1];
} ESS_APPINFO_T, *ESS_PAPPINFO_T, **ESS_PPAPPINFO_T;
```

データ型	フィールド	説明
ESS_APPNAME_T	Name	アプリケーション名
ESS_SVRNAME_T	Server	サーバー名
ESS_USHORT_T	Status	アプリケーションのロード・ステータス(ロード済または未ロード)。このフィールドには次の値が含められます: ● ESS_STATUS_NOTLOADED ● ESS_STATUS_LOADING ● ESS_STATUS_LOADED ● ESS_STATUS_UNLOADING
ESS_USHORT_T	AppType	アプリケーションのタイプ。有効な値は次のとおりです: ● ESS_APP_UNICODE - 0x0003 - プログラムは Unicode クライアント・プログラムです。サーバーが Unicode モードでない場合、関数は失敗します。これはデフォルト値です。 ● ESS_APP_NONUNICODE - 0x0002 - プログラムは非 Unicode モード・クライアント・プログラムです。
ESS_CHAR_T	AppLocale	アプリケーションのロケール記述。データ型は ESS_LOCALESTRING_LENGTH です。
ESS_USHORT_T	nConnects	アプリケーションに現在接続しているユーザー数
ESS_TIME_T	ElapsedAppTime	アプリケーションをロードしてから経過した秒数
ESS_USHORT_T	nDbs	このアプリケーションのデータベース数
ESS_DATA_STORAGE_T	StorageType	ストレージ・タイプ。有効な値は次のとおりです: ● 0 - デフォルト(1 と同じ) ● 1 - 多次元(ブロック・ストレージ) ● 4 - 集約ストレージ ● 1000 - 未定義
ESS_DBNAME_T	DbNames [1]	アプリケーションのデータベースをすべてリストしたデータベース名文字列の動的配列(nDb 個の要素を持つ)。

ESS_APPINFOEX_T

この拡張アプリケーション情報構造体は、EssGetApplicationInfo に使用される標準の 120 ページの「ESS_APPINFO_T」とは若干異なります。この拡張構造体は EssGetApplicationInfoEx に使用されます。

フィールドは次のとおりです:

```
   typedef struct  ESS_APPINFOEX_T
{
 ESS_APPNAME_T    Name;
 ESS_SVRNAME_T    Server;
 ESS_USHORT_T,    AppType;
 ESS_CHAR_T,     AppLocale, ESS_LOCALESTRING_LENGTH;
 ESS_USHORT_T     Status;
 ESS_USHORT_T     nConnects;
 ESS_TIME_T      ElapsedAppTime;
 ESS_DATA_STORAGE_T  StorageType;
} ESS_APPINFOEX_T, *ESS_PAPPINFOEX_T, **ESS_PPAPPINFOEX_T;
```

データ型	フィールド	説明
ESS_APPNAME_T	Name	アプリケーション名
ESS_SVRNAME_T	Server	サーバー名
ESS_USHORT_T	AppType	アプリケーションのタイプ。 有効な値は次のとおりです: ● ESS_APP_UNICODE - 0x0003 - プログラムは Unicode クライアント・プログラムです。サーバーが Unicode モードでない場合、関数は失敗します。これはデフォルト値です。 ● ESS_APP_NONUNICODE - 0x0002 - プログラムは非 Unicode モード・クライアント・プログラムです。
ESS_CHAR_T	AppLocale	アプリケーションのロケール記述。データ型は ESS_LOCALESTRING_LENGTH です。
ESS_USHORT_T	Status	アプリケーションのロード・ステータス(ロード済または未ロード)。このフィールドには次の値が含められます: ● ESS_STATUS_NOTLOADED ● ESS_STATUS_LOADING ● ESS_STATUS_LOADED ● ESS_STATUS_UNLOADING
ESS_USHORT_T	nConnects	アプリケーションに現在接続しているユーザー数
ESS_TIME_T	ElapsedAppTime	アプリケーションをロードしてから経過した秒数

データ型	フィールド	説明
ESS_DATA_STORAGE_T	StorageType	ストレージ・タイプ。有効な値は次のとおりです: ● 0 - デフォルト(1 と同じ) ● 1 - 多次元(ブロック・ストレージ) ● 4 - 集約ストレージ ● 1000 - 未定義

ESS_APPSTATE_T

このアプリケーション状態構造体は、特定のアプリケーションの状態パラメータを取得および設定します。この構造体のすべてのフィールドは、API を使用して変更できます。ただし、フィールドには、集約ストレージ・データベースに適用されないものもあります。120 ページの「ESS_APPINFO_T」構造体も参照してください。これには、変更できない追加のアプリケーション情報が含まれます。フィールドは次のとおりです:

```
   typedef struct  ESS_APPSTATE_T
{
 ESS_DESC_T  Description;
 ESS_BOOL_T  Loadable;
 ESS_BOOL_T  Autoload;
 ESS_ACCESS_T Access;
 ESS_BOOL_T  Connects;
 ESS_BOOL_T  Commands;
 ESS_BOOL_T  Updates;
 ESS_BOOL_T  Security;
 ESS_ULONG_T  LockTimeout;
 ESS_ULONG_T  lroSizeLimit;
} ESS_APPSTATE_T, *ESS_PAPPSTATE_T, **ESS_PPAPPSTATE_T;
```

データ型	フィールド	説明
ESS_DESC_T	Description	アプリケーションの説明(最大 80 文字)
ESS_BOOL_T	Loadable	アプリケーションがロード可能かどうかを示すフラグ(ESS_TRUE の場合: アプリケーションがロード可能)
ESS_BOOL_T	Autoload	Essbase の起動時にアプリケーションが自動的にロードされるかどうかを示すフラグ(ESS_TRUE の場合、アプリケーションが自動的にロードされます)
ESS_ACCESS_T	Access	アプリケーション内のデータベースに対するデフォルトのアクセス権(すべてのユーザーに関する最低レベルのアクセス権)。このフィールドには次の値が含められます: ● ESS_PRIV_NONE ● ESS_PRIV_DBDESIGN ● ESS_PRIV_CALC ● ESS_PRIV_WRITE ● ESS_PRIV_READ

データ型	フィールド	説明
ESS_BOOL_T	Connects	ユーザーがアプリケーションに接続できるかどうかを示すフラグ(ESS_TRUE の場合、ユーザーは接続できます)。
ESS_BOOL_T	Commands	ユーザーがアプリケーションにコマンドを発行できるかどうかを示すフラグ(アプリケーションがユーザー・コマンドを受け入れる場合は ESS_TRUE)。
ESS_BOOL_T	Updates	ユーザーがアプリケーションのデータを更新できるかどうかを示すフラグ(アプリケーションがユーザーの更新コマンドを受け入れる場合は ESS_TRUE)。
ESS_BOOL_T	Security	アプリケーションのセキュリティが使用可能かどうかを示すフラグ(ESS_TRUE の場合、セキュリティが使用可能です)。
ESS_ULONG_T	LockTimeout	ブロックレベルのロックが自動的に解除されるまでのタイムアウト期間(秒)。このフィールドは集約ストレージ・データベースには適用されません。
ESS_ULONG_T	lroSizeLimit	LRO ファイルのサイズに対する制限値。この制限値は、各アプリケーションに設定され、管理者またはプログラムは、大きすぎるリンク・ファイルからサーバーを保護できます。Essbase 自体は、サイズを制限せず、デフォルト値もありません。この制限値は、LRO URL (512 文字までに制限)または LRO セル・ノート(599 文字までに制限)に適用されません。このフィールドは集約ストレージ・データベースには適用されません。

ESS_ATTRIBUTEINFO_T

特定のメンバーに関する属性情報を含んでいます。ESS_ATTRIBUTEINFO_T は、EssGetAttributeInfo によって使用されます。

```
   typedef struct ESS_ATTRIBUTEINFO_T
{
 ESS_MBRNAME_T     MbrName;
 ESS_MBRNAME_T     DimName;
 ESS_ATTRIBUTEVALUE_T  Attribute;
} ESS_ATTRIBUTEINFO_T, *ESS_PATTRIBUTEINFO_T, **ESS_PPATTRIBUTEINFO_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	MbrName	157 ページの「ESS_MEMBERINFO_T」または768 ページの「ESS_MBRINFO_T」の属性メンバー名。ロング名を含みます
ESS_MBRNAME_T	DimName	属性次元名
ESS_ATTRIBUTEVALUE_T 項	Attribute	属性値

ESS_ATTRIBUTEVALUE_T

属性メンバーの値およびタイプに関する情報を含んでいます。

```
   typedef struct ESS_ATTRIBUTEVALUE_T
{
 ESS_USHORT_T        usDataType;
 union
```

```
 {
  ESS_BOOL_T    bData;
  ESS_STR_T     strData;
  ESS_DATETIME_T  dtData;
  ESS_DOUBLE_T   dblData;
 }
 value;
} ESS_ATTRIBUTEVALUE_T, *ESS_PATTRIBUTEVALUE_T, **ESS_PPATTRIBUTEVALUE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usDataType	属性次元または属性メンバーのデータ型を指定する定数識別子。 属性次元またはゼロレベル(リーフ・ノード)の属性メンバーに対しては、次のいずれかの値になります: ● ESS_ATTRMBRDT_BOOL ● ESS_ATTRMBRDT_STRING ● ESS_ATTRMBRDT_DATETIME ● ESS_ATTRMBRDT_DOUBLE ● 属性次元ではなく、属性メンバーに対しては、次のいずれかの値になります: ● ESS_ATTRMBRDT_NONE ● ESS_ATTRMBRDT_AUTO
ESS_BOOL_T ESS_STR_T ESS_DATETIME_T ESS_DOUBLE_T	value value.bData value.strData value.dtData value.dblData	次の属性メンバー値に対するユニオン変数: ● ブール値 ● 文字列値 ● 日付と時刻の値 ● DOUBLE 値

ESS_ATTRSPECS_T

アウトラインの属性指定を設定する場合に **EssOtlSetAttributeSpecifications()**によって使用されます。また、アウトラインの属性指定を取得する場合には、**EssOtlGetAttributeSpecifications()**および **EssGetAttributeSpecifications()**によって使用されます。

```
   typedef struct ESS_ATTRSPECS_T
{
 ESS_USHORT_T  usGenNameBy;
 ESS_USHORT_T  usUseNameOf;
 ESS_CHAR_T   cDelimiter;
 ESS_USHORT_T  usDateFormat;
 ESS_USHORT_T  usBucketingType;
 ESS_STR_T    pszDefaultTrueString;
 ESS_STR_T    pszDefaultFalseString;
 ESS_STR_T    pszDefaultAttrCalcDimName;
 ESS_STR_T    pszDefaultSumMbrName;
 ESS_STR_T    pszDefaultCountMbrName;
 ESS_STR_T    pszDefaultAverageMbrName;
```

```
 ESS_STR_T   pszDefaultMinMbrName;
 ESS_STR_T   pszDefaultMaxMbrName;
} ESS_ATTRSPECS_T, *ESS_PATTRSPECS_T, **ESS_PPATTRSPECS_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usGenNameBy	ロング名を生成するときに接頭辞または接尾辞としてゼロレベルメンバーの世代を使用するかどうかを示す定数識別子: ● ESS_GENNAMEBY_PREFIX (デフォルト値) ● ESS_GENNAMEBY_SUFFIX
ESS_USHORT_T	usUseNameOf	ロング名を生成するときに使用するゼロレベルメンバーの世代を示す定数識別子: ● ESS_USENAMEOF_NONE (デフォルト値) ● ESS_USENAMEOF_PARENT ● ESS_USENAMEOF_GRANDPARENTANDPARENT ● ESS_USENAMEOF_ALLANCESTORS ● ESS_USENAMEOF_DIMENSION
ESS_CHAR_T	cDelimiter	ロング名を生成するときに使用する区切り記号を示す定数識別子: ● ESS_DELIMITER_UNDERSCORE (デフォルト値) ● ESS_DELIMITER_PIPE ● ESS_DELIMITER_CARET
ESS_USHORT_T	usDateFormat	日時属性のフォーマットを示す定数識別子: ● ESS_DATEFORMAT_MMDDYYYY (デフォルト値) ● ESS_DATEFORMAT_DDMMYYYY
ESS_USHORT_T	usBucketingType	数値属性のバケットのタイプを示す定数識別子: ● ESS_UPPERBOUNDINCLUSIVE (デフォルト値) ● ESS_UPPERBOUNDNONINCLUSIVE ● ESS_LOWERBOUNDINCLUSIVE ● ESS_LOWERBOUNDNONINCLUSIVE
ESS_STR_T	pszDefaultTrueString	TRUE を示すためブール属性と使用される文字列。デフォルト値は ESS_DEFAULT_TRUESTRING (「TRUE」)です。
ESS_STR_T	pszDefaultFalseString	FALSE を示すためブール属性と使用される文字列。デフォルト値は ESS_DEFAULT_FALSESTRING ("False")です。
ESS_STR_T	pszDefaultAttrCalcDimName	属性計算(集約)次元の名前。デフォルト値は ESS_DEFAULT_ATTRIBUTECALCULATIONS ("Attribute Calculations")です。
ESS_STR_T	pszDefaultSumMbrName	SUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESS_DEFAULT_SUM ("Sum")です。
ESS_STR_T	pszDefaultCountMbrName	COUNT を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESS_DEFAULT_COUNT ("Count")です。
ESS_STR_T	pszDefaultAverageMbrName	AVERAGE を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESS_DEFAULT_AVERAGE ("Average")です。

データ型	フィールド	説明
ESS_STR_T	pszDefaultMinMbrName	MINIMUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESS_DEFAULT_MIN ("Min")です。
ESS_STR_T	pszDefaultMaxMbrName	MAXIMUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESS_DEFAULT_MAX ("Max")です。

ESS_BLDDL_STATE_T

次元構築およびデータロードの進行状況に関する情報が含まれます。

```
typedef struct ESS_BLDDL_STATE_T
{
  ESS_USHORT_T   usProcessState;
  ESS_USHORT_T   usProcessStage;
  ESS_LONG_T    ilProcessStatus;
  ESS_ULONG_T   ulRecordsProcessed;
  ESS_ULONG_T   ulRecordsRejected;
} ESS_BLDDL_STATE_T, *ESS_PBLDDL_STATE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usProcessState	次元構築/データ・ロード・プロセスの状態: 進行中、最終段階または完了済。値については、「usProcessState の定数値」を参照してください。
ESS_USHORT_T	usProcessStage	次元構築/データ・ロード・プロセスの段階: データ・ソースを開いている最中、アウトラインの読取り中、次元の構築中、アウトラインの確認中またはアウトラインの書込み中。値については、「usProcessStage の定数値」を参照してください。
ESS_LONG_T	ilProcessStatus	次元構築/データ・ロード・プロセスのステータス(関数の戻りステータスと同じ)
ESS_ULONG_T	ulRecordsProcessed	現在までに処理されたデータ・レコード数
ESS_ULONG_T	ulRecordsRejected	現在までに拒否されたデータ・レコード数

usProcessState の定数値

```
#define ESS_BLDDL_STATE_DONE    0    /* No process, or process complete */
#define ESS_BLDDL_STATE_INPROGRESS  1    /* Process is in progress */
#define ESS_BLDDL_STATE_FINALSTAGE  5    /* Process at final stage */
```

usProcessStage の定数値

```
#define ESS_BLDDL_STAGE_NONE      0    /* No process */
#define ESS_BLDDL_STAGE_OPENDATASOURCE  1    /* Process at opening data source */
#define ESS_BLDDL_STAGE_OPENOTL     2    /* Process at reading outline */
#define ESS_BLDDL_STAGE_BUILDOTL     3    /* Process at building dimension */
#define ESS_BLDDL_STAGE_VERIFYOTL    4    /* Process at verifying outline */
#define ESS_BLDDL_STAGE_WRITEOTL     5    /* Process at writing outline */
#define ESS_BLDDL_STAGE_RESTRUCT     6    /* Process at restructuring database */
#define ESS_BLDDL_STAGE_DATALOAD     7    /* Process at loading data */
#define ESS_BLDDL_STAGE_FINALIZE     8    /* Process at finalizing*/
```

ESS_CONNECTINFO_T

特定のサーバーに接続しているプロセスに関する情報を保管します。

```
   typedef struct ESS_CONNECTINFO_T
{
  ESS_USERNAME_T  Name;      /* logged in user name */
  ESS_APPNAME_T  AppName;    /* connected application */
  ESS_DBNAME_T   DbName;     /* connected database */
  ESS_SVRNAME_T  LoginMachine;  /* login machine name */
  ESS_ULONG_T   LoginIP;     /* IPv4 address of the login machine */
  ESS_TIME_T    LastLogin;   /* login time */
}
 ESS_CONNECTINFO_T, *ESS_CONNECTINFO_T, **ESS_CONNECTINFO_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ログインしたユーザー名。
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)。
ESS_DBNAME_T	DbName	現在接続しているデータベースの名前(該当する場合)。
ESS_SVRNAME_T	LoginMachine	ログインしたマシンの名前。マシン名をネットワーク上で解決できない場合、このフィールドには文字列としてフォーマットされた IP アドレスが含まれます。アステリスク(*)は EssListLogins を呼び出したセッションを示します。
ESS_ULONG_T	LoginIP	ログインしたマシンの IP アドレス。
ESS_TIME_T	LastLogin	最後のログインの時刻。

ESS_CONNECTINFOEX_T

特定のサーバーに接続しているプロセスに関する情報を保管します。この構造体は ESS_CONNECTINFO_T に似ていますが、ProviderName および connparam フィールドが追加されています。

```
   typedef struct ESS_CONNECTINFOEX_T
{
  ESS_USERNAME_T  Name;
  ESS_USERNAME_T  ProviderName;
```

```
  ESS_CONNPARAM_T connparam;
  ESS_APPNAME_T  AppName;
  ESS_DBNAME_T   DbName;
  ESS_SVRNAME_T  LoginMachine;
  ESS_ULONG_T    LoginIP;
  ESS_TIME_T     LastLogin;
}
 ESS_CONNECTINFOEX_T, *ESS_PCONNECTINFOEX_T, **ESS_PPCONNECTINFOEX_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ログインしたユーザー名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: `native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER`
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)
ESS_DBNAME_T	DbName	データベース名
ESS_SVRNAME_T	LoginMachine	ログインしたマシンの名前。マシン名をネットワーク上で解決できない場合、このフィールドには文字列としてフォーマットされた IP アドレスが含まれます。アステリスク(*)は `EssListLogins` を呼び出したセッションを示します。
ESS_ULONG_T	LoginIP	ログインしたマシンの IP アドレス
ESS_TIME_T	LastLogin	最後のログインの時刻

ESS_DBFILEINFO_T

`EssListDbFiles` によって取得されるインデックスまたはデータ・ファイルの情報を含んでいます。

```
   typedef struct ess_dbfileinfo_t
{
 ESS_APPNAME_T  AppName;
 ESS_DBNAME_T   DbName;
 ESS_FILENAME_T FilePath;
 ESS_SIZE_T     FileSize;
 ESS_USHORT_T   FileSequenceNum;
 ESS_USHORT_T   FileCount;
 ESS_USHORT_T   FileType;
 ESS_BOOL_T     FileOpen;
}  ESS_DBFILEINFO_T, *ESS_PDBFILEINFO_T, **ESS_PPDBFILEINFO_T;
```

データ型	フィールド	説明
ESS_APPNAME_T	AppName	アプリケーション名
ESS_DBNAME_T	DbName	データベース名
ESS_FILENAME_T	FilePath	ファイル・パス
ESS_SIZE_T	FileSize	ファイルのサイズ(単位はバイト)
ESS_USHORT_T	FileSequenceNum	指定されたデータベースの FileType のファイルのセット内での 1 から始まるシーケンス番号
ESS_USHORT_T	FileCount	戻された FileType のファイル数
ESS_USHORT_T	FileType	次のいずれかのファイル・タイプ: ● ESS_FILETYPE_INDEX ● ESS_FILETYPE_DATA
ESS_BOOL_T	FileOpen	ファイルが開いているかどうかを示すフラグ: ファイルが閉じているときは 0、ファイルが開いているときはゼロ以外です

ESS_DBINFO_T

このデータベース情報構造体は、特定のデータベースに関する情報を入手します。この構造体のフィールドは、API を使用して変更できません。134 ページの「ESS_DBSTATE_T」構造体も参照してください。変更できる追加のデータベース状態パラメータを含んでいます。また、137 ページの「ESS_DBSTATS_T」構造体も参照してください。フィールドは次のとおりです:

```
   typedef struct  ESS_DBINFO_T
{
 ESS_APPNAME_T    AppName;
 ESS_DBNAME_T     Name;
 ESS_USHORT_T     DbType;
 ESS_USHORT_T     Status;
 ESS_USHORT_T     nConnects;
 ESS_USHORT_T     nLocks;
 ESS_ULONG_T     nDims;
 ESS_USHORT_T     Data;
 ESS_MBRNAME_T    Country;
 ESS_MBRNAME_T    Time;
 ESS_MBRNAME_T    Category;
 ESS_MBRNAME_T    Type;
 ESS_MBRNAME_T    CrPartition;
 ESS_TIME_T      ElapsedDbTime;
 ESS_ULONG64_T    DataFileCacheSetting;
 ESS_ULONG64_T    DataFileCacheSize;
 ESS_ULONG64_T    DataCacheSetting;
 ESS_ULONG64_T    DataCacheSize;
 ESS_ULONG_T     IndexCacheSetting;
 ESS_ULONG64_T    IndexCacheSize;
 ESS_ULONG_T     IndexPageSetting;
 ESS_ULONG_T     IndexPageSize;
```

```
 ESS_DBREQINFO_T  DbReqInfoAry[ESS_DBREQNUM];
 ESS_BOOL_T     bDbReadOnly;
 ESS_BOOL_T     bDataCompress;
 ESS_USHORT_T    usDataCompressType;
 ESS_ULONG_T    ulRetrievalBuffer;
 ESS_ULONG_T    ulRetrievalSortBuffer;
 ESS_BOOL_T     bCacheMemLocking;
 ESS_BOOL_T     bPreImage;
 ESS_USHORT_T    usIsolationLevel;
 ESS_LONG_T     lTimeOut;
 ESS_ULONG_T    ulCommitBlocks;
 ESS_ULONG_T    ulCommitRows;
 ESS_ULONG_T    ulDiskVolumeCount;
 ESS_DISKVOLUME_T  aDiskVolume[1];
} ESS_DBINFO_T, *ESS_PDBINFO_T, **ESS_PPDBINFO_T;
```

データ型	フィールド	説明
ESS_APPNAME_T	AppName	関連付けられたアプリケーション名
ESS_DBNAME_T	Name	データベース名
ESS_USHORT_T	DbType	データベースのタイプ。値: ● ESS_DBTYPE_NORMAL ● ESS_DBTYTPE_CURRENCY
ESS_USHORT_T	Status	データベースのロード・ステータス(ロード済または未ロード)。値: ● ESS_STATUS_NOTLOADED ● ESS_STATUS_LOADING ● ESS_STATUS_LOADED ● ESS_STATUS_UNLOADING
ESS_USHORT_T	nConnects	現在データベースに接続しているユーザーの数
ESS_USHORT_T	nLocks	排他的なロックが現在設定されているデータ・ブロックの数
ESS_ULONG_T	nDims	データベースの次元数
ESS_USHORT_T	Data	データベースからのデータのロード状態を表すフラグ。値: ESS_DBDATA_NONE: データがロードされてない ESS_DBDATA_LOADNOCALC: データはロードされたが計算されていない ESS_DBDATA_CLEAN: データ・ロードと計算が終了している
ESS_MBRNAME_T	Country	通貨国次元メンバー(ある場合)。ない場合、最初のバイトは NULL です。
ESS_MBRNAME_T	Time	通貨時間次元メンバー(ある場合)。ない場合、最初のバイトは NULL です。

データ型	フィールド	説明
ESS_MBRNAME_T	Category	通貨カテゴリ次元メンバー(ある場合)。ない場合、最初のバイトは NULL です。
ESS_MBRNAME_T	Type	通貨タイプ次元メンバー(通貨データベースのみ)。存在しない場合、最初のバイトは NULL です。
ESS_MBRNAME_T	CrPartition	通貨パーティション・メンバー(非通貨データベースのみ)
ESS_TIME_T	ElapsedDbTime	データベースがロードされている秒数
ESS_ULONG64_T	DataFileCacheSetting	現在有効なデータ・ファイル・キャッシュ・サイズの設定値。
ESS_ULONG64_T	DataFileCacheSize	現在データベースが使用しているランタイム・データ・ファイル・キャッシュ・サイズ(KB 単位)。データ・ファイル・キャッシュ・サイズを変更した後、新しいデータ・ファイル・キャッシュ・サイズを有効にするには、データベースを停止して再起動する必要があることに注意してください。
ESS_ULONG64_T	DataCacheSetting	現在有効なデータ・キャッシュ・サイズの設定値(KB)。
ESS_ULONG64_T	DataCacheSize	データ・キャッシュのランタイム・サイズ(KB)
ESS_ULONG_T	IndexCacheSetting	現在有効なインデックス・キャッシュ・サイズの設定値(KB)。
ESS_ULONG64_T	IndexCacheSize	インデックス・キャッシュのランタイム・サイズ(KB)。
ESS_ULONG_T	IndexPageSetting	現在有効なインデックス・ページ・サイズの設定(KB)。
ESS_ULONG_T	IndexPageSize	インデックス・ページのランタイム・サイズ(KB)。
133 ページの「ESS_DBREQINFO_T」	DbReqInfo, Ary[ESS_DBREQNUM]	最終計算日、最終データロード日、最終アウトライン更新日など、要求した情報の配列
ESS_BOOL_T	bDbReadOnly	データベースが読取り専用モードの場合は TRUE。それ以外の場合は FALSE。
ESS_BOOL_T	bDataCompress	オプションの圧縮フラグ(デフォルトは YES)。
ESS_USHORT_T	usDataCompressType	オプションの圧縮フラグを設定している場合のデータ圧縮タイプ(デフォルトは BitMap)。
ESS_ULONG_T	ulRetrievalBuffer	取得要求ごとに割り当てられる取得バッファのサイズ(デフォルトは 2048 バイト)。
ESS_ULONG_T	ulRetrievalSortBuffer	取得要求ごとに割り当てられる取得ソート・バッファのサイズ(デフォルトは 10240 バイト)。
ESS_BOOL_T	bCacheMemLocking	インデックス・キャッシュとデータ・キャッシュのメモリー・ページが、物理メモリー内でロックされている場合は TRUE。
ESS_BOOL_T	bPreImage	前にコミットされたデータを読み取るためのフラグ。
ESS_USHORT_T	usIsolationLevel	コミット設定(デフォルトは UNCOMMITTED)。

データ型	フィールド	説明
ESS_LONG_T	lTimeOut	COMMITTED アクセスのみに設定されるタイム・アウト間隔(秒)。
ESS_ULONG_T	ulCommitBlocks	計算およびスプレッドシートの更新中、明示コミットを実行する前に更新されたデータ・ブロックの数。
ESS_ULONG_T	ulCommitRows	データロード中、明示コミットをプロセスする前にプロセスされた入力ファイルの行数。
ESS_ULONG_T	ulDiskVolumeCount	このデータベース用に設定されたディスク・ボリュームの数。
ESS_DISKVOLUME_T	aDiskVolume[1]	ディスク・ボリューム設定の配列

ESS_DBREQINFO_T

EssGetDatabaseInfo()で使用されます。Essbase には情報が存在する要求のタイプとして、データのロード、計算、アウトラインの更新の 3 つがあります。次の Essbase API 定数は各タイプの要求を特定します:

```
   typedef struct ESS_DBREQINFO_T
{
 ESS_ULONG_T    DbReqType;
 ESS_USERNAME_T  User;
 ESS_TIMERECORD_T StartTimeRec;
 ESS_TIMERECORD_T EndTimeRec;
 ESS_ULONG_T    DbReqFlags;
} ESS_DBREQINFO_T, *ESS_PDBREQINFO_T;
```

データ型	値	説明
ESS_DBREQTYPE_DATLOAD	0	データのロード
ESS_DBREQTYPE_CALC	1	計算
ESS_DBREQTYPE_OTLUPD	2	アウトラインの更新

フィールドは次のとおりです:

データ型	フィールド	説明
ESS_ULONG_T	DbReqType	データベース要求のタイプ
ESS_USERNAME_T	User	ユーザー名
198 ページの「ESS_TIMERECORD_T」	StartTimeRec	要求の開始時間
198 ページの「ESS_TIMERECORD_T」	EndTimeRec	要求の終了時間

データ型	フィールド	説明
ESS_ULONG_T	DbReqFlags	データベース要求に関する追加情報を提供する情報フラグのビットマップ。DbReqType が CALC の場合に使用されます。次のフラグを使用できます: ● デフォルト(現在情報が含まれない)。ESS API 定数: ESS_DBREQFLAG_CALCDEF (デフォルト計算が実行された) ● カスタム計算スクリプト。ESS API 定数: ESS_DBREQFLAG_CALCDSCR (カスタム計算が実行された)

ESS_DBSTATE_T

このデータベース状態構造体は、特定のデータベース用に状態パラメータを入手し設定します。この構造体内のフィールドはすべて、API を使用して変更できます。130 ページの「ESS_DBINFO_T」および137 ページの「ESS_DBSTATS_T」構造体も参照してください。変更できない追加のデータベース情報を含んでいます。

```
   typedef struct  ESS_DBSTATE_T
{
 ESS_DESC_T     Description;
 ESS_BOOL_T     Loadable;
 ESS_BOOL_T     Autoload;
 ESS_ACCESS_T    Access;
 ESS_SHORT_T    IndexType;
 ESS_ULONG64_T   MaxMem;
 ESS_ULONG64_T   MaxMemDataFileCache;
 ESS_BOOL_T     CalcNoAggMissing;
 ESS_BOOL_T     CalcNoAvgMissing;
 ESS_BOOL_T     CalcTwoPass;
 ESS_BOOL_T     CalcCreateBlock;
 ESS_DBNAME_T    CrDbName;
 ESS_MBRNAME_T   CrTypeMember;
 ESS_USHORT_T    CrConvType;
 ESS_ULONG64_T   MaxMemIndex;
 ESS_ULONG_T    IndexPageSize;
 ESS_BOOL_T     DataCompress;
 ESS_USHORT_T    DataCompressType;
 ESS_ULONG_T    RetrievalBuffer;
 ESS_ULONG_T    RetrievalSortBuffer;
 ESS_BYTE_T     cIOAccessFlagInUse;    /* new for 6.5
 ESS_BOOL_T     bNoWaitIO;        /* new for 6.5
 ESS_USHORT_T    IsolationLevel;
 ESS_BOOL_T     PreImage;
 ESS_BYTE_T     cIOAccessFlagPending;   /* new for 6.5
 ESS_LONG_T     TimeOut;
 ESS_ULONG_T    CommitBlocks;
 ESS_ULONG_T    CommitRows;
 ESS_ULONG_T    nVolumes;
 ESS_DISKVOLUME_T  DiskVolume[1];
} ESS_DBSTATE_T, *ESS_PDBSTATE_T, **ESS_PPDBSTATE_T;
```

フィールドは次のとおりです:

データ型	フィールド	説明
ESS_DESC_T	Description	データベースの説明(シングル・バイト文字換算で最大 80 文字)
ESS_BOOL_T	Loadable	データベースをロードできるかどうかを示すフラグ(アプリケーションがロード可能な場合、ESS_TRUE)
ESS_BOOL_T	Autoload	アプリケーションの起動時に、データベースが自動的にロードされるかどうかを示すフラグ(データベースが自動的にロードされる場合は、ESS_TRUE)
ESS_ACCESS_T	Access	データベースへのデフォルトのアクセス・レベル。このフィールドが含むことができる値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
ESS_USHORT_T	IndexType	データベース・インデックス・タイプ(配列またはツリー)。このフィールドには次の値が含められます: ● ESS_INDEXTYPE_ARRAY ● ESS_INDEXTYPE_AVL API リリース 4 以降については、IndexType フィールドは廃止されています。
ESS_ULONG64_T	MaxMem	データベースで、圧縮されていないデータ・ブロックに対して予約されている最大メモリー(バイト単位)
ESS_ULONG64_T	MaxMemDataFileCache	データ・ファイル・キャッシュに対して予約されている最大メモリー(バイト単位)
ESS_BOOL_T	CalcNoAggMissing	メンバーの子がすべて欠落している場合に、メンバーを集約しないためのフラグ(欠落した値を集約しない場合は ESS_TRUE)
ESS_BOOL_T	CalcNoAvgMissing	平均値を計算する際に欠落しているメンバーを含めないようにするフラグ(欠落した値を含めない場合は ESS_TRUE)
ESS_BOOL_T	CalcTwoPass	データベースで完全な計算を実行するときに 2 パス計算を強制的に実行するためのフラグ(2 パス計算が使用可能な場合は、ESS_TRUE)
ESS_BOOL_T	CalcCreateBlock	定数割当て計算式でデータ・ブロックの作成を強制するフラグ(疎次元にのみ有効)。ブロックを強制的に作成する場合、ESS_TRUE に設定します。
ESS_DBNAME_T	CrDbName	関連付けられた通貨データベースの名前(通貨データベース以外で有効)。
ESS_MBRNAME_T	CrTypeMember	通貨換算タイプ・メンバーの名前(非通貨データベースで有効)
ESS_USHORT_T	CrConvType	通貨換算タイプ(通貨換算を乗算で計算するか、または除算で計算するか)。値: ● ESS_CRCTYPE_DIV ● ESS_CRCTYPE_MULT
ESS_ULONG64_T	MaxMemIndex	最小インデックス・キャッシュ・サイズ。値: 1048576。定数 ESS_INDEXCACHEMIN_SIZE を使用して設定します。

データ型	フィールド	説明
ESS_ULONG_T	IndexPageSize	バッファ・プールが作成されるインデックス・ページのサイズ(バイト単位)。 最小インデックス・ページ・サイズ。値: 1024。定数 ESS_INDEXPAGEMIN_SIZE を使用して設定します IndexPageSize フィールドの最大ページ・サイズ。値: 8192。定数 ESS_INDEXPAGEMAX_SIZE を使用して設定します
ESS_BOOL_T	DataCompress	このデータベースのブロックを圧縮するかどうか指定するオプションのフラグ。
ESS_USHORT_T	DataCompressType	オプションの圧縮フラグが設定されている場合、書込み操作で使用されるデータ圧縮のタイプ。 ● Bitmap - データ・セルを示すのにビットマップを使用します(デフォルト値)。 ● Run-Length Encoding - 連続して繰り戻される値をすべて圧縮します。 ● No Compression - データを圧縮しません。
ESS_ULONG_T	RetrievalBuffer	抽出された行のデータ・セルが RESTRICT、TOP または BOTTOM コマンドによって評価される前に、これらのセルを保管するサーバーのバッファのサイズを、バイト単位で指定します。デフォルトは 10240 バイトです。最小は 2048 バイトで、最大は 102400000 バイトです。
ESS_ULONG_T	RetrievalSortBuffer	取得の際にソートするデータを保管するサーバーのバッファのサイズを、バイト単位で指定します。最小は 2048 バイトで、最大は 102400000 バイトです。
ESS_BYTE_T	cIOAccessFlagInUse	アクティブな現在のデータベースが使用している I/O アクセスのタイプ。アクセスの 2 つのタイプは、ESS_IO_ACCESS_BUFFERED および ESS_IO_ACCESS_DIRECT です。cIOAccessFlagPending が ESS_IO_ACCESS_DIRECT に設定されている場合、いくつかの操作はまだバッファを必要とする場合があります。また、直接アクセスは一定のプラットフォーム上でサポートされない場合があります。このフィールドは読取り専用です。
ESS_BOOL_T	bNoWaitIO	これは、Essbase が直接 I/O 操作の終了を待つかどうかを制御します。直接 I/O をサポートするプラットフォーム上で、cIOAccessFlag が ESS_IO_ACCESS_DIRECT である場合のみ、これは適用されます。このフィールドは読取り専用です。デフォルトは TRUE です。
ESS_USHORT_T	IsolationLevel	コミット設定は次のとおりです: ● COMMITTED - トランザクションがコミットされるまで、影響を受けるすべてのデータ・ブロックに対して書込みロックをしてアクセスを制限します。 ● UNCOMMITTED (デフォルト)- トランザクションの実行中、必要に応じて書込みロックの取得や解放を行います。
ESS_BOOL_T	PreImage	読取りのみ要求の間に以前にコミットしたデータを読み取るためのフラグ。このフラグは、コミット・アクセスにのみ設定できます。デフォルトは YES です。
ESS_BYTE_T	cIOAccessFlagPending	Essbase が使用する I/O アクセスのタイプ(直接またはバッファ)。この設定は、次の DBLoad (開く操作)の後に有効になります。

データ型	フィールド	説明
ESS_LONG_T	TimeOut	タイムアウト間隔(秒単位)。これは、コミット・アクセスにのみ設定できます。 -1 は、待ち時間に制限がありません。 0 は即時アクセスで待機なし(デフォルト)。 n は、秒単位で指定された間隔です。
ESS_ULONG_T	CommitBlocks	明示コミットを実行する前に変更したデータ・ブロックの数(コミット設定が UNCOMMITTED の場合にのみ使用)。
ESS_ULONG_T	CommitRows	明示コミットを実行する前にデータ・ロードする入力ファイルの行数(コミット設定が UNCOMMITTED の場合についてのみ使用)。
ESS_ULONG_T	nVolumes	このデータベース用に設定されたディスク・ボリュームの数。
ESS_DISKVOLUME_T	DiskVolume[1]	ディスク・ボリューム設定の配列。

ESS_DBSTATS_T

このデータベース統計構造体は、特定のデータベースに関する実行時の統計情報を取得します。この構造体のフィールドは、API を使用して変更できません。変更できる追加のデータベース状態パラメータを含む、134 ページの「ESS_DBSTATE_T」構造体も参照してください。また、130 ページの「ESS_DBINFO_T」構造体も参照してください。フィールドは次のとおりです:

```
   typedef struct ESS_DBSTATS_T
{
 ESS_USHORT_T    IndexType;
 ESS_ULONG_T    nDims;
 ESS_ULONG_T    DeclaredBlockSize;
 ESS_ULONG_T    ActualBlockSize;
 ESS_DOUBLE_T    DeclaredMaxBlocks;
 ESS_DOUBLE_T    ActualMaxBlocks;
 ESS_DOUBLE_T    NonMissingLeafBlocks;
 ESS_DOUBLE_T    NonMissingNonLeafBlocks;
 ESS_DOUBLE_T    NonMissingBlocks;
 ESS_DOUBLE_T    PagedOutBlocks;
 ESS_DOUBLE_T    PagedInBlocks;
 ESS_DOUBLE_T    InMemCompBlocks;
 ESS_DOUBLE_T    TotalBlocks;
 ESS_DOUBLE_T    AverageFragmentationQuotient;
 ESS_DOUBLE_T    BytesOfRecoverableFreeSpace;
 ESS_DOUBLE_T    TotMemPagedInBlocks;
 ESS_DOUBLE_T    TotMemBlocks;
 ESS_DOUBLE_T    TotMemIndex;
 ESS_DOUBLE_T    TotMemInMemCompBlocks;
 ESS_DOUBLE_T    BlockDensity;
 ESS_DOUBLE_T    SparseDensity;
 ESS_DOUBLE_T    CompressionRatio;
 ESS_DOUBLE_T    ClusterRatio;
 ESS_DIMSTATS_T  DimStatsAry[1];
```

```
} ESS_DBSTATS_T, *ESS_PDBSTATS_T, **ESS_PPDBSTATS_T;
```

データ型	フィールド	説明
ESS_USHORT_T	IndexType	データベース・インデックス・タイプ(配列またはツリー)。このフィールドには次の値が含められます: ● ESS_INDEXTYPE_ARRAY ● ESS_INDEXTYPE_AVL
ESS_ULONG_T	nDims	データベース中の次元数。
ESS_ULONG_T	DeclaredBlockSize	宣言されたデータ・ブロックのサイズ。
ESS_ULONG_T	ActualBlockSize	実際のデータ・ブロックのサイズ
ESS_DOUBLE_T	DeclaredMaxBlocks	データベース内の宣言された最大ブロック数。
ESS_DOUBLE_T	ActualMaxBlocks	データベース内の実際の最大ブロック数。
ESS_DOUBLE_T	NonMissingLeafBlocks	データベース内の欠落していないリーフ(最下位レベル)ブロックの数。
ESS_DOUBLE_T	NonMissingNonLeafBlocks	データベース内の欠落していない非リーフ(上位レベル)ブロックの数。
ESS_DOUBLE_T	NonMissingBlocks	廃止。0 を戻します。
ESS_DOUBLE_T	PagedOutBlocks	現在ディスクにページ・アウトされているデータベース・ブロックの数。
ESS_DOUBLE_T	PagedInBlocks	現在メモリーにページ・インされているデータベース・ブロックの総数。
ESS_DOUBLE_T	InMemCompBlocks	現在、圧縮メモリーにページ・インされているデータベース・ブロックの数。
ESS_DOUBLE_T	TotalBlocks	既存のデータ・ブロックの総数(最大数でない)。
ESS_DOUBLE_T	AverageFragmentationQuotient	データ・ファイル内で、空き領域または Essbase が使用していない領域のパーセンテージです。
ESS_DOUBLE_T	BytesOfRecoverableFreeSpace	● 回復可能な空き領域の概算バイト数 ● 空き領域の回復が必要でない場合は-1
ESS_DOUBLE_T	TotMemPagedIn-Blocks	ページインされた(非圧縮)データベース・ブロックすべてに使用するメモリーの合計。
ESS_DOUBLE_T	TotMemBlocks	すべてのデータベース・ブロックに使用されているメモリーの合計。
ESS_DOUBLE_T	TotMemIndex	データベース・インデックスに使用されているメモリーの合計。
ESS_DOUBLE_T	TotMemInMemCompBlocks	現在、圧縮メモリーにページ・インされているデータベース・ブロックに使用されているメモリーの合計。
ESS_DOUBLE_T	BlockDensity	データベース・ブロックの平均密度(現在ロードされているすべてのブロックを使用して計算)。

データ型	フィールド	説明
ESS_DOUBLE_T	SparseDensity	データベース内の疎次元の平均密度。
ESS_DOUBLE_T	CompressionRatio	ディスク上のデータ・ブロックの平均圧縮率。
ESS_DOUBLE_T	ClusterRatio	ページ・ファイルの断片化のメジャー。1 に近い値は、断片化の程度が低いことを示します。0 に近い値は、計算およびクエリーのパフォーマンスに影響する可能性がある、高度の断片化を示します。
140 ページの「ESS_DIMSTATS_T」	DimStatsAry [1]	ESS_DIMSTATS_T というタイプを持つ次元統計構造体の(nDim 個の要素を持つ)動的配列。「構造体の定義」を参照してください。

ESS_DIMENSIONINFO_T

EssGetDimensionInfo()で使用されます。フィールドは次のとおりです:

```
   typedef struct ESS_DIMENSIONINFO_T
{
 ESS_MBRNAME_T DimName;
 ESS_DIMNUM_T  DimNumber;
 ESS_USHORT_T  DimType;
 ESS_USHORT_T  DimTag;
 ESS_ULONG_T   DeclaredDimSize;
 ESS_ULONG_T   ActualDimSize;
 ESS_DESC_T    Description;
 ESS_USHORT_T  DimDataType;
} ESS_DIMENSIONINFO_T, *ESS_PDIMENSIONINFO_T, **ESS_PPDIMENSIONINFO_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	DimName	次元名
ESS_DIMNUM_T	DimNumber	次元番号
ESS_USHORT_T	DimType	次元タイプ。値: ● ESS_DIMTYPE_DENSE ● ESS_DIMTYPE_SPARSE
ESS_USHORT_T	DimTag	次元タグ・タイプ。値: ● ESS_TTYPE_ATTRCALC ● ESS_TTYPE_ATTRIBUTE ● ESS_TTYPE_CCATEGORY ● ESS_TTYPE_CNAME ● ESS_TTYPE_CTIME ● ESS_TTYPE_NONE ● ESS_TTYPE_CPARTITION ● ESS_TTYPE_CTYPE

データ型	フィールド	説明
ESS_ULONG_T	DeclaredDimSize	宣言された次元サイズ
ESS_ULONG_T	ActualDimSize	実際の次元のサイズ
ESS_DESC_T	Description	予約済(現在サポートされていません)
ESS_USHORT_T	DimDataType	属性次元のデータ型。値: ● ESS_ATTRMBRDT_BOOL ● ESS_ATTRMBRDT_DATETIME ● ESS_ATTRMBRDT_DOUBLE ● ESS_ATTRMBRDT_STRING

ESS_DIMSTATS_T

特定のデータベース次元に関する情報を取得するため使用される次元統計構造体です。この構造体のフィールドは、API を使用して変更できません。これらの構造体の配列は 137 ページの「ESS_DBSTATS_T」構造体の末尾にあり、データベース内の各次元の情報を提供します。フィールドは次のとおりです:

```
   typedef struct ESS_DIMSTATS_T
{
 ESS_MBRNAME_T   DimName;
 ESS_USHORT_T    DimType;
 ESS_ULONG_T    DeclaredDimSize;
 ESS_ULONG_T    ActualDimSize;
} ESS_DIMSTATS_T, *ESS_PDIMSTATS_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	DimName	次元メンバー名
ESS_USHORT_T	DimType	次元タイプ(疎または密)。このフィールドには次の値が含められます: ● ESS_DIMTYPE_SPARSE ● ESS_DIMTYPE_DENSE
ESS_ULONG_T	DeclaredDimSize	宣言された次元サイズ(指定した次元のラベルのみまたは共有メンバーも含めた、その次元で宣言されたメンバーの数)
ESS_ULONG_T	ActualDimSize	実際の次元サイズ(指定した次元のラベルのみまたは共有メンバーを除いた、その次元で宣言されたメンバーの数)

ESS_DTAPICOLUMN_T

特定の列に関するヘッダー情報を定義します。

```
   typedef struct ESS_DTAPICOLUMN_T
{
```

```
 ESS_LONG_T  nColumnIdx;
 ESS_LONG_T  nDisplayOrder
 ESS_CHAR_T  sViewName[ESS_MBRNAMELEN];
 ESS_CHAR_T  sColumnName[ESS_MBRNAMELEN];
 ESS_USHORT_T uDatatype;
 ESS_LONG_T  nSortOrder;
 ESS_LONG_T  nSortSequence;
 ESS_BOOL_T  bFilterOnly;
 ESS_CHAR_T  sFilter[ESS_MAX_DATALEN + 1];
} ESS_DTAPICOLUMN_T, *ESS_PDTAPICOLUMN_T, **ESS_PPDTAPICOLUMN_T;
```

データ型	フィールド	説明
ESS_LONG_T	nColumnIdx	列の位置を示すゼロから始まるインデックス(読取り専用)
ESS_LONG_T	nDisplayOrder	列が表示される順番
ESS_CHAR_T	sViewName [ESS_MBRNAMELEN]	(読取り専用)
ESS_CHAR_T	sColumnName [ESS_MBRNAMELEN]	指定したデータ列のヘッダー・テキスト(読取り専用)
ESS_USHORT_T	uDataType	指定したデータ列のデータ型 ● ESS_DT_STRING ● ESS_DT_DATETIME ● ESS_DT_DOUBLE
ESS_LONG_T	nSortOrder	
ESS_LONG_T	nSortSequence	
ESS_BOOL_T	bFilterOnly	ESS_TRUE =フィルタのみ。
ESS_CHAR_T	sFilter [ESS_MAX_DATALEN + 1]	

ESS_DTAPIDATA_T

特定のデータ・セルのレポート・データを定義します。

```
   typedef struct ESS_DTAPIDATA_T
{
 ESS_ULONG_T nRowIdx;
 ESS_ULONG_T nColumnIdx;
 ESS_CHAR_T  sData[ESS_MAX_DATALEN + 1];
} ESS_DTAPIDATA_T, *ESS_PDTAPIDATA_T, **ESS_PPDTAPIDATA_T;
```

データ型	フィールド	説明
ESS_ULONG_T	nRowIdx	指定したデータ・ブロックで 0 から始まるインデックスを付けた行番号
ESS_ULONG_T	nColumnIdx	指定したデータ・ブロックで 0 から始まるインデックスを付けた列番号

データ型	フィールド	説明
ESS_CHAR_T	sData [ESS_MAX_ DATALEN + 1]	指定したデータ・ブロックのデータの値

ESS_DTAPIHEADER_T

特定の列に関するヘッダー情報を定義します。

```
   typedef struct __ess_dtapiheader_t
{
    ESS_ULONG_T       nColumnIdx ;
    ESS_CHAR_T        sViewName[ESS_MBRNAMELEN] ;
    ESS_CHAR_T        sColumnName[ESS_MBRNAMELEN] ;
    ESS_USHORT_T   uDatatype ;
} ESS_DTAPIHEADER_T, *ESS_PDTAPIHEADER_T, **ESS_PPDTAPIHEADER_T ;
```

データ型	フィールド	説明
ESS_ULONG_T	nColumnIdx	指定したデータ・ブロックで 0 から始まるインデックスを付けた列番号。
ESS_CHAR_T	sViewName [ESS_MBRNAMELEN]	
ESS_CHAR_T	sColumnName [ESS_DESCRIPTION_ LEN + 1]	指定したデータ・ブロックのデータの値。
ESS_USHORT_T	uDatatype	

ESS_DTAPIINFO_T

データ・セルの範囲に関する接続情報を定義します。

```
   typedef struct ESS_DTAPIINFO_T
{
 ESS_CHAR_T  sHisName[ESS_MAX_NAME + 1];
 ESS_CHAR_T  sUsername[ESS_MAX_NAME + 1];
 ESS_CHAR_T  sPassword[ESS_MAX_NAME + 1];
 ESS_USHORT_T uInputOption;
} ESS_DTAPIINFO_T, *ESS_PDTAPIINFO_T, **ESS_PPDTAPIINFO_T;
```

データ型	フィールド	説明
ESS_CHAR_T	sHisName [ESS_MAX_NAME + 1]	
ESS_CHAR_T	sUsername [ESS_MAX_NAME + 1]	
ESS_CHAR_T	sPassword [ESS_MAX_NAME + 1]	(書込み専用)
ESS_USHORT_T	uInputOption	(読取り専用)

ESS_DTAPIREPORT_T

特定の列に関するヘッダー情報を定義します。

```
   typedef struct ESS_DTAPIREPORT_T
{
 ESS_LONG_T nReportId;
 ESS_CHAR_T sName[ESS_DTREPORT_NAME + 1];
 ESS_LONG_T nCustomize;
 ESS_LONG_T nRowGovernor;
 ESS_LONG_T nTimeGovernor;
} ESS_DTAPIREPORT_T, *ESS_PDTAPIREPORT_T, **ESS_PPDTAPIREPORT_T;
```

データ型	フィールド	説明
ESS_LONG_T	nReportId	
ESS_CHAR_T	sName [ESS_DTREPORT_NAME + 1]	
ESS_LONG_T	nCustomize	
ESS_LONG_T	nRowGovernor	
ESS_LONG_T	nTimeGovernor	

ESS_DTBUFFER_T

レポート・データ・セルを定義します。

```
   typedef struct ESS_DTBUFFER_T
{
 ESS_ULONG_T row;
 ESS_ULONG_T column;
 ESS_CHAR_T  data[ESS_DESCRIPTION_LEN + 1];
} ESS_DTBUFFER_T, *ESS_PDTBUFFER_T, **ESS_PPDTBUFFER_T;
```

データ型	フィールド	説明
ESS_ULONG_T	row	指定したデータ・ブロックで 0 から始まるインデックスを付けた行番号。
ESS_ULONG_T	column	指定したデータ・ブロックで 0 から始まるインデックスを付けた列番号。
ESS_CHAR_T	data [ESS_DESCRIPTION_LEN + 1]	指定したデータ・ブロックのデータの値。

ESS_DTDATA_T

レポート・データ・セルを定義します。

```
   typedef struct ESS_DTDATA_T
{
 ESS_ULONG_T row;
 ESS_ULONG_T column;
 ESS_CHAR_T  data[ESS_DESCRIPTION_LEN + 1];
} ESS_DTDATA_T, *ESS_PDTDATA_T, **ESS_PPDTDATA_T;
```

データ型	フィールド	説明
ESS_ULONG_T	row	指定したデータ・ブロックで 0 から始まるインデックスを付けた行番号。
ESS_ULONG_T	column	指定したデータ・ブロックで 0 から始まるインデックスを付けた列番号。
ESS_CHAR_T	data [ESS_DESCRIPTION_LEN + 1]	指定したデータ・ブロックのデータの値。

ESS_DTHEADER_T

特定の列に関するヘッダー情報を定義します。

```
   typedef struct ESS_DTHEADER_T
{
 ESS_ULONG_T     colIndex;
 ESSDTREPORTDATATYPE dataType;
 ESS_CHAR_T      data[ESS_DESCRIPTION_LEN + 1];
} ESS_DTHEADER_T, *ESS_PDTHEADER_T, **ESS_PPDTHEADER_T;
```

データ型	フィールド	説明
ESS_ULONG_T	colIndex	列の位置を示すゼロから始まるインデックス。
ESSDTREPORTDATATYPE	dataType	指定したデータ列のデータ型。
ESS_CHAR_T	data [ESS_DESCRIPTION_LEN + 1]	指定したデータ列のヘッダー・テキスト。

ESS_DISKVOLUME_REPLACE_T

ソースおよび宛先ディスク・ボリューム・ラベルの名前を含んでいます。ソースは現在存在しており、宛先に置き換えられます。

```
   typedef_struct ess_diskvolume_replace_t
{
    ESS_FILENAME_T, szPartition_Src;
    ESS_FILENAME_T, szPartition_Dest;
} ESS_DISKVOLUME_REPLACE_T;
```

データ型	フィールド	説明
ESS_FILENAME_T	szPartition_Src	置き換えられるディスク・パーティションの名前。
ESS_FILENAME_T	szPartition_Dest	`szPartition_Src` を置き換えるディスク・パーティションの名前

ESS_DURLINFO_T

ドリルスルー URL の情報を取得します。

1904 ページの「ドリルスルー URL の制限」を参照してください。

```
   typedef struct url
{
 ESS_CHAR_T   bIsLevel0;
 ESS_STR_T    cpURLName;
 ESS_USHORT_T  iURLXmlSize;
 ESS_BYTE_T*   cpURLXml;
 ESS_USHORT_T  iCountOfDrillRegions;
 ESS_PSTR_T   cppDrillRegions;
}  ESS_DURLINFO_T;
```

データ型	フィールド	説明
ESS_CHAR_T	bIsLevel0	1 の場合、URL 定義はレベル 0 のデータに制限されます。0 の場合、制限はありません
ESS_STR_T	cpURLName	ドリルスルー URL 名
ESS_USHORT_T	iURLXmlSize	URL XML テキストのサイズ
ESS_BYTE_T*	cpURLXml	URL XML テキストへのポインタ
ESS_USHORT_T	icountOfDrillRegions	ドリルスルー URL によって参照される領域の数 ドリルスルー URL のドリル可能領域の数は、256 に制限されています。
ESS_PSTR_T	cppdrillRegions	ドリルスルー URL によって参照される領域のリスト

ESS_EXTUSERINFO_T

外部認証されたユーザーの情報を保管します。フィールドは次のとおりです:

```
   typedef struct ESS_EXTUSERINFO_T
{
 ESS_USERNAME_T  Name;
 ESS_APPNAME_T   AppName;
 ESS_DBNAME_T    DbName;
 ESS_BOOL_T     Login;
 ESS_USHORT_T    Type;
 ESS_ACCESS_T    Access;
 ESS_ACCESS_T    MaxAccess;
```

```
 ESS_DATE_T    Expiration;
 ESS_TIME_T    LastLogin;
 ESS_TIME_T    DbConnectTime;
 ESS_USHORT_T   FailCount;
 ESS_LOGINID_T  LoginId;
 ESS_DESC_T    Description;
 ESS_EMAIL_T    EMailID;
 ESS_BOOL_T    LockedOut;
 ESS_BOOL_T    PwdChgNow;
 ESS_USHORT_T   authType;
 ESS_PROTOCOL_T  protocol;
 ESS_CONNPARAM_T  connParam;
} ESS_EXTUSERINFO_T, *ESS_PEXTUSERINFO_T, **ESS_PPEXTUSERINFO_T,;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ユーザー名
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)
ESS_DBNAME_T	DbName	現在接続しているデータベースの名前(該当する場合)
ESS_BOOL_T	Login	ログイン・ステータスを示すフラグ。
ESS_USHORT_T	Typ	構造体のタイプ。このフィールドには次の値が含められます: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
ESS_ACCESS_T	Access	ユーザーに割り当てられたデフォルトのアクセス権限。値: 次のビット値を任意に組み合せられます: ● ESS_ACCESS_SUPER /* スーパーバイザ、全ビットを設定 */ ● ESS_PRIV_APPCREATE /* アプリケーションの作成/削除権限 */ ● ESS_PRIV_USERCREATE /* ユーザーの作成/削除権限 */
ESS_ACCESS_T	MaxAccess	ユーザーの最大アクセス権(グループのメンバーシップによる個別のアクセス権とアクセス・レベルを含む)。
ESS_DATE_T	Expiration	ユーザーのパスワードの失効日。
ESS_TIME_T	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付。
ESS_TIME_T	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。**EssSetUser** では設定できません。
ESS_USHORT_T	FailCount	最後に正常にログインしてからの、失敗したログインの回数。
ESS_LOGINID_T	LoginId	ユーザー・ログイン識別タグ。
ESS_DESC_T	Description	ユーザーの説明。
ESS_EMAIL_T	EMailID	ユーザーの電子メール・アドレス。
ESS_BOOL_T	LockedOut	ユーザーがロック・アウトされていることを示すフラグ。
ESS_BOOL_T	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ。

データ型	フィールド	説明
ESS_USHORT_T	authType	認証タイプ。
ESS_PROTOCOL_T	protocol	外部認証プロトコル: Shared Services モードの場合は CSS。
ESS_CONNPARAM_T	connParam	外部認証接続パラメータ。プロトコルが CSS の場合は NULL。

ESS_GENLEVELNAMEEX_T

世代名またはレベル名、および世代とレベルのメンバー名の一意性の設定に関する情報が含まれています。フィールドは次のとおりです:

```
   typedef_struct ESS_GENLEVELNAMEEX_T)
{
 ESS_USHORT_T,  usNumber;
 ESS_BOOL_T,   bNameUnique;
 ESS_MBRNAME_T,  szName;

} ESS_GENLEVELNAMEEX_T, ESS_PGENLEVELNAMEEX_T, ESS_GENLEVELNAMEEX_T **,
ESS_PPGENLEVELNAMEEX_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usNumber	世代番号またはレベル番号
ESS_BOOL_T	bNameUnique	世代またはレベルのメンバー名の一意性
ESS_MBRNAME_T	szName	世代名またはレベル名

ESS_GLOBAL_T

管理に使用されるグローバル・サーバー・システム・パラメータを含んでいます。この構造体の Currency を除くフィールドはすべて、API を使用して変更できます。フィールドは次のとおりです:

```
   typedef struct ESS_GLOBAL_T
{
 ESS_BOOL_T  Security;
 ESS_BOOL_T  Logins;
 ESS_ACCESS_T Access;
 ESS_USHORT_T Validity;
 ESS_BOOL_T  Currency;
 ESS_USHORT_T PwMin;
 ESS_TIME_T  InactivityTime;
 ESS_TIME_T  InactivityCheck;

 ESS_USHORT_T  InvalidAttempts;
 ESS_USHORT_T  InactivityLockout;
 ESS_USHORT_T  NumPwExpWarn;
 ESS_USHORT_T  PwStoredNum;
```

```
} ESS_GLOBAL_T, *ESS_PGLOBAL_T, **ESS_PPGLOBAL_T;
```

データ型	フィールド	説明
ESS_BOOL_T	Security	グローバルにセキュリティが使用可能にされているかどうかを示すフラグです(デフォルトは ESS_TRUE。この場合セキュリティは使用可能です)
ESS_BOOL_T	Logins	ユーザーのログインが使用可能かどうかを示すフラグです(デフォルトは ESS_TRUE。この場合ログインは使用可能です)。
ESS_ACCESS_T	Access	新しく作成されたアプリケーションのデフォルトのアクセス・レベルです(デフォルトは ESS_ACCESS_NONE)。使用できる値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
ESS_USHORT_T	Validity	デフォルトのパスワード有効期間(デフォルト値は 365 日間)。
ESS_BOOL_T	Currency	通貨オプションがサポートされているかどうかを示すフラグ(このフラグは読取り専用です)。通貨オプションが使用可能な場合は ESS_TRUE に設定されます。
ESS_USHORT_T	PwMin	パスワードの最小の長さ(デフォルトは 6 文字)。
ESS_TIME_T	InactivityTime	すべてのアプリケーションおよびエージェントから非アクティブなユーザーが自動的にログアウトされるまでの最大時間を秒数で表します。デフォルト値: 3600 秒。最小値: 300 秒。自動ログアウトを使用不可にするには、InactivityTime を 0 に設定します。
ESS_TIME_T	InactivityCheck	自動ログアウトの確認頻度を秒数で表します。デフォルト値: 300 秒。最小値: 30 秒。InactivityTime の設定より低くなければ、InactivityCheck は InactivityTime に設定され、警告メッセージが表示されます。自動ログアウトを使用不可にするには、InactivityCheck を 0 に設定します。
ESS_USHORT_T	InvalidAttempts	システム管理者に警告が通知され、ユーザーがロック・アウトされるまで、ユーザーに対して許可される無効な試行回数。
ESS_USHORT_T	InactivityLockout	ロック・アウトされるまで、ユーザーが非アクティブ(ログインからログインまでの間隔)な状態でいられる期間(デフォルトは 365 日)
ESS_USHORT_T	NumPwExpWarn	ユーザーがロック・アウトされるまで、ユーザーに対して発行されるパスワードの期限切れ警告の回数。
ESS_USHORT_T	PwStoredNum	任意のユーザーに対して保管されるパスワードの数。

ESS_INIT_T

API 初期化関数 **EssInit()**に渡され、API 開発者が API の使用をカスタマイズできるフィールドを含んでいます。構造体のフィールドのいずれかが 0(またはポインタについては NULL)に設定されている場合、API のデフォルトが使用されます。(詳細は、88 ページの「C プログラムにおけるメモリーの使用」を参照してください。)

```
typedef struct ESS_INIT_T
{
 ESS_ULONG_T  Version;
 ESS_PVOID_T  UserContext;
 ESS_USHORT_T MaxHandles;
 ESS_SIZE_T   MaxBuffer;
 ESS_STR_T    LocalPath;
 ESS_STR_T    MessageFile;
 ESS_PFUNC_T  AllocFunc;
 ESS_PFUNC_T  ReallocFunc;
 ESS_PFUNC_T  FreeFunc;
 ESS_PFUNC_T  MessageFunc;
 ESS_STR_T    HelpFile;
 ESS_ULONG_T  Ess_System;
#ifdef AD_UTF8
 ESS_USHORT_T, usApiType;
#endif
 ESS_PCATCHFUNC_T,  CatchFunc;
 ESS_PCATCH_INIT_FUNC_T,  CatchInitFunc;
 ESS_PCATCH_TERM_FUNC_T,  CatchTermFunc;
 ESS_PCOOKIE_CREATE_FUNC_T, CookieCreateFunc;
 ESS_PCOOKIE_DELETE_FUNC_T, CookieDeleteFunc;
} ESS_INIT_T, *ESS_PINIT_T;
```

データ型	フィールド	説明
ESS_ULONG_T	Version	アプリケーションのコンパイルに使用する Essbase API のバージョン。ESS_API_VERSION に設定します。下位互換性のために使用されます。
ESS_PVOID_T	UserContext	ユーザー定義のメッセージ・コンテキストへのオプションのポインタ(ユーザー定義のメッセージ関数に引数として渡されます)
ESS_USHORT_T	MaxHandles	API プログラムが必要とする同時コンテキスト・ハンドルの最大数(1-255)。デフォルトは 255 です。この数を減らすと、API 内でプログラムのために使用されるクライアント・メモリーの量が減る可能性があります。
ESS_SIZE_T	MaxBuffer	クライアント・プログラムで割り当てることができるバッファの最大サイズ(通常は 64KB)。デフォルトは 64KB です。
ESS_STR_T	LocalPath	クライアントでファイルやオブジェクトを操作するためのデフォルトのローカル・パス名。これが設定されていない場合、Essbase は、デフォルトでは ESSBASEPATH 環境変数を使用し、渡されるディレクトリ名には \CLIENT を追加します。
ESS_STR_T	MessageFile	メッセージ・データベース・ファイル、ESSBASE.MDB の修飾パス名。これが設定されていない場合、Essbase は最初に、ARBORMSGPATH 環境変数の完全修飾パスを使用しようとします。それ以外の場合は、(ESSBASEPATH)\BIN\SSBASE.MDB を使用します。ESSBASEPATH が定義されていない場合、実行時にエラーが戻されます。
ESS_PFUNC_T	AllocFunc	ユーザー定義のメモリー割当て関数へのポインタ。すべてのプラットフォーム: メモリー割当て関数は malloc()関数を使用します。
ESS_PFUNC_T	ReallocFunc	ユーザー定義のメモリー再割当て関数へのポインタ。すべてのプラットフォーム: メモリー割当て関数は realloc()関数を使用します。

データ型	フィールド	説明
ESS_PFUNC_T	FreeFunc	ユーザー定義のメモリー解放関数へのポインタ。すべてのプラットフォーム: メモリー割当て関数は free()関数を使用します。
ESS_PFUNC_T	MessageFunc	ユーザー定義のメッセージ・コールバック関数へのポインタ。ユーザー定義のコールバック関数に送信されたメッセージは、EssInit で Essbase に渡されます。リリース 6.2 より以前は、メッセージが NLS 文字(アクセント付き文字など外国語文字)を含んでいた場合、Essbase は、OEM (DOS) フォーマットでそれらを提供していました。リリース 6.2 以降は、これらのメッセージは、文字の誤解を避けるため、完全に文字(Windows)フォーマットになっています。これは、Essbase のローカライズ・バージョンのみに影響します。
ESS_STR_T	HelpFile	ユーザー定義のアプリケーション・ヘルプ・ファイルの完全修飾パス名で、「自動ログイン」ダイアログ・ボックスのヘルプに使用されます。ログイン・ヘルプ・コンテキストはヘルプ・ファイルで定義する必要があります。 第 3 章「Essbase と使用中の製品との統合」を参照してください。 デフォルトでは、「ヘルプ」ボタンをクリックすると、『Oracle Essbase Spreadsheet Add-in ユーザー・ガイド』オンライン・ヘルプに含まれている Essbase システムのログインに関するヘルプ・トピックが表示されます。 ESSBASEPATH が定義されていない場合、ヘルプ・ファイル名は NULL に設定されます。
ESS_ULONG_T	Ess_System	内部で使用するために予約されています。NULL に設定します
ESS_USHORT_T	usApiType	必須。プログラムが Unicode モードと非 Unicode モードのどちらであるかを定義します。有効な値は、112 ページの「Unicode モードの定数(C)」を参照してください。
typedef ESS_BOOL_T (*ESS_ PCATCHFUNC_T) (ESS_HCTX_T);	CatchFunc	クライアントによって実装されると、Essbase はクエリー中にこの関数を断続的(数秒ごと)に呼び出します。ルーチンが TRUE を戻すと、API 呼出しが取り消されます。
typedef ESS_STS_T (*ESS_PCATCH_ INIT_FUNC_T)(ESS_ HCTX _T);	CatchInitFunc	この関数は、CatchFunc 呼出しにどのような状態が必要であっても、リソースを初期化します。たとえば、ユーザーが[Esc]キーを押すかどうか、そして、CatchFunc がルーチンを呼び出してキーボードからのデータを取得するかどうかに基づいてクエリーを終了したい場合は、CatchFunc 呼出しのたびに初期化されないようにメモリーを事前初期化する必要がある場合があります。 Essbase は、クエリー中に次のプロセスを実行します: 1. 非 NULL の場合に、CatchInitFunc を呼び出します。 2. CatchFunc を断続的に呼び出しながら、クエリーを実行します。 3. 非 NULL の場合に、CatchTermFunc を呼び出します。
typedef ESS_STS_T (*ESS_PCATCH_ TERM_FUNC_T) (ESS_ HCTX _T);	CatchTermFunc	この関数は、CatchInitFunc によって初期化されたリソースを終了します。

データ型	フィールド	説明
typedef ESS_STS_T (*ESS_PCOOKIE_ CREATE_FUNC_T) (ESS_HCTX_T);	CookieCreateFunc	Essbase は、SetActive 時にこの関数を呼び出します。CatchFunc、CatchInitFunc、および CatchTermFunc 呼出しにユーザー情報が必要な場合に、この関数を使用します。たとえば、一定のユーザー・アクティビティに基づいてクエリーを終了したい場合は、CatchFunc 呼出しで使用される Cookie を作成する必要があります。EssGetCookie を呼び出して、Cookie を取得します。
typedef ESS_STS_T (*ESS_PCOOKIE_ DELETE_FUNC_T) (ESS_HCTX_T);	CookieDeleteFunc	この関数は、CookieCreateFunc によって作成された Cookie を削除します。Essbase は、ClearActive 時にこの関数を呼び出します。

Essbase API を使用したクエリー取消し

Essbase API を使用して開発されたプログラムは、オプションで初期化時にカスタム・クエリー取消し関数を登録できます。ESS_INIT_T には、クエリー取消しのためのカスタム・コールバック関数を作成できるフィールドが 5 つあります。そのフィールドは、CatchFunc、CatchInitFunc、CatchTermFunc、CookieCreateFunc、CookieDeleteFunc です。これらはデフォルトで NULL に設定されます。

クエリー取消しの使用例

次のコードは、[Esc]キーが押されたときにクエリー取消しを有効にします。KbdHitEx は、キーボードから入力された次のキーを取得して、キーの値を kbfinfo.chChar に書き込みます。

```
ESS_INIT_STRUCT InitStruct;
 InitStruct.CatchFunc = KillReqCatcher;

ESS_BOOL_T KillReqCatcher(ESS_HCTX_T hCtx)
{
 KBDINFO_T   kbinfo;
 if (KbdHitEx(&kbfinfo) && kbfinfo.chChar == KB_ESC)
  return ESS_TRUE;
 else
  return ESS_FALSE;
}
```

ただし、KdbHitEx ルーチンでは、最初に初期化ルーチン InitializeMyKeyboard が呼び出され、後で終了ルーチン TerminateMyKeyboard が呼び出される必要があると仮定します。この場合には、CatchInitFunc と CatchTermFunc を使用します。

```
    InitStruct.CatchInitFunc = InitKeyboard;
 InitStruct.CatchTermFunc = TerminateKeyboard;

ESS_STS_T InitKeyboard (ESS_HCTX_T hCtx)
{
 return InitializeMyKeyboard ();
}
```

```
ESS_STS_T TerminateKeyboard (ESS_HCTX_T hCtx)
{
 return TerminateMyKeyboard ();
}
```

今度は、InitializeMyKeyboard ルーチンと TerminateMyKeyboard ルーチンがステータス情報を保持する必要があると仮定します。Cookie を使用して、ステータスを保持できます。CookieCreateFunc によって作成された Cookie には、EssGetCookie によって CatchFunc、CatchInitFunc、および CatchTermFunc でアクセスできます。

```
    InitStruct.CatchInitFunc = InitKeyboard2;
 InitStruct.CatchTermFunc = TerminateKeyboard2;
 InitStruct.CookieCreateFunc = AllocKeyboardState;
 InitStruct.CookieDeleteFunc = FreeKeyboardState;

ESS_STS_T InitKeyboard2 (ESS_HCTX_T hCtx)
{
 ESS_PVOID_T cookie;
 ESS_STS_T sts;
 sts = EssGetCookie(hCtx, &cookie);
 if (sts)
  return sts;
 return InitializeMyKeyboard (cookie);
}

ESS_STS_T TerminateKeyboard2 (ESS_HCTX_T hCtx)
{
 ESS_PVOID_T cookie;
 ESS_STS_T sts;
 sts = EssGetCookie(hCtx, &cookie);
 if (sts)
  return sts;

 return TerminateMyKeyboard (cookie);
}
ESS_STS_T AllocKeyboardState(ESS_PVOID_T pKbdState)
{
 *pKbdState = malloc(KBDSTRUCT_SIZE);
 if (*pKbdState)
  return 0;
 else
  return -1;
}
ESS_STS_T FreeKeyboardState (ESS_PVOID_T kbdState)
{
 if (kbdState)
  free(kbdState);
 return 0;
}
```

ESS_LOAD_BUFFER_T

集約ストレージ・データ・ロード・バッファに関する情報が含まれています。
EssListExistingLoadBuffers に使用されます。

```
   typedef struct ESS_LOAD_BUFFER_T
{
 ESS_ULONG_T     ulBufferId;
 ESS_ULONG_T     ulDuplicateAggregationMethod;
 ESS_ULONG_T     ulOptionFlags;
 ESS_ULONG_T     ulSize;
 ESS_BOOL_T     bInternal;
 ESS_BOOL_T     bActive;
 ESS_BOOL_T     bReserved01;
 ESS_BOOL_T     bReserved02;
 ESS_ULONG_T     ulReserved01;
 ESS_ULONG_T     ulReserved02;
 ESS_ULONG_T     ulReserved03;
} ESS_LOAD_BUFFER_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulBufferId	データ・ロード・バッファの ID (1 から 4294967296 までの数字)。
ESS_ULONG_T	ulDuplicateAggregationMethod	バッファ内の同じセルに対する複数の値を組み合せる方法について記述する次の定数の 1 つ: ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD: バッファが同じセルに対する複数の値を含んでいる場合に値を追加します。 ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ASSUME_EQUAL: 同じセルに対する複数の値が同一であることを確認し、同一である場合、重複を無視します。異なる場合は、エラー・メッセージを表示してデータ・ロードを停止します。 ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_USE_LAST: セルの最後の値として、バッファにロードされた最後の値を使用します。
ESS_ULONG_T	ulOptionFlags	次のロード・バッファ・オプションのいずれか(またはその組合せ): ● ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES ● ESS_ASO_DATA_LOAD_BUFFER_IGNORE_ZERO_VALUES
ESS_ULONG_T	ulSize	データ・ロード・バッファによる使用が許可されている集約ストレージ・キャッシュのパーセンテージ(1 から 100 までの数値)
ESS_BOOL_T	bInternal	Essbase によってバッファが作成された場合は ESS_TRUE、ユーザーによってバッファが作成された場合は ESS_FALSE
ESS_BOOL_T	bActive	現在データ・ロードでバッファを使用中の場合は ESS_TRUE
ESS_BOOL_T	bReserved01	未使用
ESS_BOOL_T	bReserved02	未使用
ESS_ULONG_T	ulReserved01	未使用

データ型	フィールド	説明
ESS_ULONG_T	ulReserved02	未使用
ESS_ULONG_T	ulReserved03	未使用

ESS_LOCKINFO_T

istLocks()関数から戻された排他的にロックされているデータ・ブロックに関する情報を含んでいます。この構造体のフィールドは、API を使用して変更できません。

```
   typedef struct ESS_LOCKINFO_T
{
 ESS_USERNAME_T  UserName;
 ESS_USHORT_T   nLocks;
 ESS_TIME_T    Time;
 ESS_LOGINID_T  LoginId;
} ESS_LOCKINFO_T, *ESS_PLOCKINFO_T, **ESS_PPLOCKINFO_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名
ESS_USHORT_T	nLocks	このユーザーによって排他的にロックされているブロックの数
ESS_TIME_T	Time	ブロックが排他的にロックされた最長時間(秒単位)
ESS_LOGINID_T	LoginId	ユーザー・ログイン識別タグ

ESS_LOCKINFOEX_T

ListLocks()関数から戻された排他的にロックされているデータ・ブロックに関する情報を含んでいます。この構造体は ESS_LOCKINFO_T に似ていますが、ProviderName および connparam フィールドが追加されています。この構造体のフィールドは、API を使用して変更できません。

```
   typedef struct ESS_LOCKINFOEX_T
{
 ESS_USERNAME_T  UserName;
 ESS_USERNAME_T  ProviderName;
 ESS_CONNPARAM_T connparam
 ESS_USHORT_T   nLocks;
 ESS_TIME_T    Time;
 ESS_LOGINID_T  LoginId;
} ESS_LOCKINFOEX_T, *ESS_PLOCKINFOEX_T, **ESS_PPLOCKINFOEX_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: `native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER`
ESS_USHORT_T	nLocks	ユーザーによって排他的にロックされているブロックの数
ESS_TIME_T	Time	ブロックが排他的にロックされた最長時間(秒単位)
ESS_LOGINID_T	LoginId	ユーザー・ログイン識別タグ

ESS_LOG_DATALOAD_T

データロードを記述するメタデータを含んでいます。

```
   typedef_struct ESS_LOG_DATALOAD_T
{
    ESS_OBJTYPE_T, datfile_type;
    ESS_UCHAR_T, datfile_loc;
    ESS_FILENAME_T, dat_filename;
    ESS_UCHAR_T, isRuleFile;
    ESS_UCHAR_T, rulfile_loc;
    ESS_FILENAME_T, rul_filename;
    ESS_USERNAME_T, sql_username;
    ESS_PASSWORD_T, sql_password;
    ESS_UCHAR_T, isAbortOnErr;
    ESS_ULONG_T, reserved0;
    ESS_ULONG_T, reserved1;
    ESS_ULONG_T, reserved2;
} ESS_LOG_DATALOAD_T;
```

データ型	フィールド	説明
ESS_OBJTYPE_T	datfile_type	データ・ファイルの型
ESS_UCHAR_T	datfile_loc	データ・ファイルの場所/SQL
ESS_FILENAME_T	dat_filename	データ・ファイル名
ESS_UCHAR_T	isRuleFile	ルール・ファイルがありますか
ESS_UCHAR_T	rulfile_loc	ルール・ファイルの場所
ESS_FILENAME_T	rul_filename	ルール・ファイルの名前
ESS_USERNAME_T	sql_username	SQL 接続ユーザー名

データ型	フィールド	説明
ESS_PASSWORD_T	sql_password	SQL 接続パスワード
ESS_UCHAR_T	isAbortOnErr	エラー・ファイル名が必要とされていますか
ESS_ULONG_T	reserved0-2	今後の使用に予約

ESS_MBRALT_T

特定のメンバー別名テーブルについての情報を含んでいます。この構造体のフィールドは、API を使用して変更できません。フィールドは次のとおりです:

```
   typedef struct  ESS_MBRALT_T
{
 ESS_MBRNAME_T     MbrName;
 ESS_MBRNAME_T     AltName;
} ESS_MBRALT_T, *ESS_PMBRALT_T, **ESS_PPMBRALT_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	MbrName	メンバー名
ESS_MBRNAME_T	AltName	関連付けられた別名

ESS_MBRERR_T

メンバー・エラーのリンク・リストに使用されます。**EssImport**()が使用します。

```
   typedef struct ESS_MBRERR_T
{
 struct ess_mbrerr_t *pNext;
 ESS_USHORT_T     ErrType;
 ESS_STR_T       Name;
 ESS_STR_T       Record;
} ESS_MBRERR_T, *ESS_PMBRERR_T, **ESS_PPMBRERR_T;
```

データ型	フィールド	説明
struct ESS_MBRERR_T	*pNext	リストに含まれる次の構造体へのポインタ
ESS_USHORT_T	ErrType	エラーのタイプ
ESS_STR_T	Name	メンバー名
ESS_STR_T	Record	エラーを含むファイル・レコード

ESS_MBRUSER_T

外部データ・ソース・ユーザー情報構造体。この構造体のフィールドは API を使用して修正できません。フィールドは次のとおりです:

```
   typedef struct ESS_MBRUSER_T
{
 ESS_STR_T User;
 ESS_STR_T Password;
} ESS_MBRUSER_T, *ESS_PMBRUSER_T;
```

データ型	フィールド	説明
ESS_STR_T	User	外部データ・ソースのユーザー名
ESS_STR_T	Password	外部データ・ソースのパスワード

ESS_MEMBERINFO_T

指定したデータベース・メンバーに関する情報を含んでいます。この構造体のフィールドは、API を使用して変更できません。フィールドは次のとおりです:

```
   typedef struct ESS_MEMBERINFO_T
{
 ESS_MBRNAME_T     MbrName;
 ESS_MEMNUM_T      MbrNumber;
 ESS_MBRNAME_T     DimName;
 ESS_DIMNUM_T      DimNumber;
 ESS_USHORT_T      Status;
 ESS_SHORT_T       Level;
 ESS_SHORT_T       Generation;
 ESS_SHORT_T       UnaryCalc;
 ESS_USHORT_T      MbrTagType;
 ESS_BOOL_T       CurrConvert;
 ESS_MBRNAME_T     CrMbrName;
 ESS_DESC_T       Description;
 ESS_MBRNAME_T     ParentMbrName;
 ESS_MBRNAME_T     ChildMbrName;
 ESS_MBRNAME_T     PrevMbrName;
 ESS_MBRNAME_T     NextMbrName;
 ESS_BOOL_T       fAttributed;
 ESS_ATTRIBUTEVALUE_T  Attribute;
 ESS_BOOL_T       fHasRelDesc;
 ESS_BOOL_T       fHasHAEnabled;
} ESS_MEMBERINFO_T, *ESS_PMEMBERINFO_T, **ESS_PPMEMBERINFO_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	MbrName	メンバー名
ESS_MEMNUM_T	MbrNumber	データベース・アウトラインのメンバー番号

データ型	フィールド	説明
ESS_MBRNAME_T	DimName	メンバーの次元名
ESS_DIMNUM_T	DimNumber	メンバーの次元数
ESS_USHORT_T	Status	メンバーの共有ステータスを得るには、このフィールドのコンテンツとフォームの各定数値との間の論理 AND を実行します ● ESS_MBRSTS_xxx: ● ESS_MBRSTS_NOTSET ● ESS_MBRSTS_NEVER ● ESS_MBRSTS_LABEL ● ESS_MBRSTS_REFER ● ESS_MBRSTS_REFNME ● ESS_MBRSTS_SHARE ● ESS_MBRSTS_VIRTSTORE ● ESS_MBRSTS_VIRTNOSTORE
ESS_SHORT_T	Level	指定したメンバーの一番下の子孫から数え上げた、メンバーのレベル番号(ゼロから始まる)
ESS_SHORT_T	Generation	指定したメンバーの次元メンバーから下に数えた、メンバーの世代番号(1 から始まる)
ESS_SHORT_T	UnaryCalc	このメンバーに対するデフォルトの単項ロールアップ。フォーム ESS_UCALC_xxx の値(add、subtract、multiply、divide、percent、none または never)。 ESS_UCALC_ADD ESS_UCALC_SUB ESS_UCALC_MULT ESS_UCALC_DIV ESS_UCALC_PERCENT ESS_UCALC_NOOP ESS_UCALC_NEVER
ESS_USHORT_T	MbrTagType	メンバーのタグ付きタイプの 16 ビット・マスク。フォーム ESS_ATYPE_xxx の値。
ESS_BOOL_T	CurrConvert	通貨換算。値: ESS_TRUE および ESS_FALSE
ESS_MBRNAME_T	CrMbrName	タグ付き通貨データベース・メンバーの名前。 ● 時間次元では、タグ付き時間メンバーの名前です。 ● 国次元では、タグ付き通貨メンバーの名前です。 ● 会計次元では、タグ付きカテゴリ・メンバーの名前です。
ESS_DESC_T	Description	メンバーの説明
ESS_MBRNAME_T	ParentMbrName	指定したメンバーの親メンバーの名前。メンバーに親がない場合は、空の文字列
ESS_MBRNAME_T	ChildMbrName	指定したメンバーの最初の子のメンバー名

データ型	フィールド	説明
ESS_MBRNAME_T	PrevMbrName	指定したメンバーの前の兄弟のメンバー名
ESS_MBRNAME_T	NextMbrName	指定したメンバーの次の兄弟のメンバー名
ESS_BOOL_T	fAttributed	メンバーに属性が関連付けられているかどうかを示します。 値: ESS_TRUE および ESS_FALSE。
124 ページの「ESS_ATTRIBUTEVALUE_T」	Attribute	属性値
ESS_BOOL_T	fHasRelDesc	メンバーにはリレーショナル上の子が 1 つ以上存在します。
ESS_BOOL_T	fHasHAEnabled	次元は、ハイブリッド分析リレーショナル・パーティションが使用可能です。 次元メンバーにのみ有効です。

ESS_NEWSHAREDSERVICESNATIVEUSERINFO_T

パスワードの割当ての自動生成オプションを使用して、Shared Services へのユーザーとグループに移行した結果生成されるユーザー名とそれに対応するパスワードが含まれます。

一致する名前が見つからないため、移行中に新しい Shared Services ユーザーとして作成されています。フィールドは次のとおりです:

```
   typedef struct ESS_NEWSHAREDSERVICESNATIVEUSERINFO_T)
{
  ESS_USERNAME_T,  Name;
   ESS_PASSWORD_T,   Password;
} ESS_NEWHUBNATIVEUSERINFO_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ユーザーまたはグループ名。
ESS_PASSWORD_T	Password	ユーザー・パスワード。

ESS_OBJDEF_T

要約のオブジェクト情報を提供します。EssImport および EssBuildDimension で使用されます。この構造体のフィールドは、API によって変更できません。

```
   typedef struct ESS_OBJDEF_T
{
 ESS_HCTX_T   hCtx;
 ESS_OBJTYPE_T ObjType;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;
```

```
} ESS_OBJDEF_T, *ESS_POBJDEF_T;
```

データ型	フィールド	説明
ESS_HCTX_T	hCtx	オブジェクトのコンテキスト・ハンドル
ESS_OBJTYPE_T	ObjType	オブジェクト・タイプ。オブジェクト・タイプのリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
ESS_STR_T	AppName	アプリケーション名
ESS_STR_T	DbName	データベース名
ESS_STR_T	FileName	拡張子のない、8 文字のオブジェクト・ファイル名。この名前は、次のすべてが当てはまる場合、ローカル・ファイル名になります: ● hCtx がローカル・コンテンツ・ハンドル ● AppName および DbName が NULL です ● FileName はローカル・ファイルのフル・パス名を指します

ESS_OBJINFO_T

特定のファイル・オブジェクトに関数情報を含んでいます。この構造体のフィールドは API では変更できません。フィールドは次のとおりです:

```
   typedef struct ESS_OBJINFO_T
{
 ESS_OBJNAME_T   Name;
 ESS_OBJTYPE_T   Type;
 ESS_APPNAME_T   AppName;
 ESS_DBNAME_T    DbName;
 ESS_ULONG_T     FileSize;
 ESS_BOOL_T      Locked;
 ESS_USERNAME_T  User;
 ESS_TIME_T      TimeStamp;
 ESS_TIMERECORD_T TimeModified;
} ESS_OBJINFO_T, *ESS_POBJINFO_T, **ESS_PPOBJINFO_T;
```

データ型	フィールド	説明
ESS_OBJNAME_T	Name	オブジェクト名
ESS_OBJTYPE_T	Type	オブジェクト・タイプ。オブジェクト・タイプのリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
ESS_APPNAME_T	AppName	アプリケーション名
ESS_DBNAME_T	DbName	データベース名
ESS_ULONG_T	FileSize	ディスク上に割り当てられたオブジェクトのファイル・サイズ (バイト単位)

データ型	フィールド	説明
ESS_BOOL_T	Locked	オブジェクトがサーバーでロックされているかどうかを示すフラグ(ESS_TRUE の場合、オブジェクトがロックされています)
ESS_USERNAME_T	User	オブジェクトをロックしたユーザー名(ロックされている場合)、それ以外の場合は未定義です
ESS_TIME_T	TimeStamp	オブジェクトがロックされた日付および時刻(ロックされている場合)、それ以外の場合は未定義です
198 ページの「ESS_TIMERECORD_T」	TimeModified	最後に行われた変更の日付と時刻

ESS_PART_T

メイン共有パーティション・データ構造体です。

```
   typedef struct ESS_PART_T
{
 ESS_PARTHDR_T    file_header;
 ESS_USHORT_T    part_count;
 ESS_PARTDEF_T   *parts;
 ESS_ULONG_T    maxserialno;
} ESS_PART_T, *ESS_PPART_T, **ESS_PPPART_T;
```

データ型	フィールド	説明
168 ページの「ESS_PARTHDR_T」	file_header	ファイル・ヘッダー。
ESS_USHORT_T	partition_count	共有パーティションの数。
166 ページの「ESS_PARTDEF_T」	partitions	共有パーティション定義の配列。
ESS_ULONG_T	maxserialno	シリアル番号に対する高いウォータ・マーク。

ESS_PART_CONNECT_INFO_T

データベースを指定します。

```
   typedef struct ESS_PART_CONNECT_INFO_T
{
 ESS_STR_T  pszHostName;
 ESS_STR_T  pszAppName;
 ESS_STR_T  pszDbName;
} ESS_PART_CONNECT_INFO_T, *ESS_PPART_CONNECT_INFO_T, **ESS_PPPART_CONNECT_INFO_T;
```

データ型	フィールド	説明
ESS_STR_T	pszHostName	ホスト名。

データ型	フィールド	説明
ESS_STR_T	pszAppName	アプリケーション名。
ESS_STR_T	pszDbName	データベース名。

ESS_PART_DEFINED_T

共有パーティションを指定します。

```
   typedef struct ESS_PART_DEFINED_T
{
 ESS_USHORT_T         usType;
 ESS_USHORT_T         usDirection;
 ESS_PART_CONNECT_INFO_T    HostDatabase;
} ESS_PART_DEFINED_T, *ESS_PPART_DEFINED_T, **ESS_PPPART_DEFINED_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usType	下に示した操作タイプ定数のいずれかになります。
ESS_USHORT_T	usDirection	下に示した方向定数のいずれかになります。
161 ページの「ESS_PART_CONNECT_INFO_T」	HostDatabase	ホスト・サーバー。

操作タイプ定数

```
   define ESS_PARTITION_OP_REPLICATED  0x0001
define ESS_PARTITION_OP_LINKED    0x0002
define ESS_PARTITION_OP_TRANSPARENT  0x0004
define ESS_PARTITION_OP_ALLTYPES (ESS_PARTITION_OP_REPLICATED |
               ESS_PARTITION_OP_LINKED |
               ESS_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   define ESS_PARTITION_DATA_SOURCE  0x0001
define ESS_PARTITION_DATA_TARGET  0x0002
define ESS_PARTITION_DATA_BOTH (ESS_PARTITION_DATA_SOURCE |
              ESS_PARTITION_DATA_TARGET)
```

ESS_PART_INFO_T

複数キューブの共有パーティション情報を保持します。

```
   typedef struct ESS_PART_INFO_T
{
 ESS_USHORT_T  OperationType;
 ESS_USHORT_T  DataDirection;
```

```
 ESS_USHORT_T  MetaDirection;
 ESS_SVRNAME_T SvrName;
 ESS_APPNAME_T AppName;
 ESS_DBNAME_T  DbName;
 ESS_TIME_T   LastMetaUpdateTime;
 ESS_TIME_T   LastRefreshTime;
 ESS_BOOL_T   AreaUpdatable;
 ESS_BOOL_T   IncrRefreshAllowed;
 ESS_TIME_T   LastUpdateTime;
} ESS_PART_INFO_T, *ESS_PPART_INFO_T, **ESS_PPPART_INFO_T;
```

データ型	フィールド	説明
ESS_USHORT_T	OperationType	このパーティションによってサポートされる操作のタイプ。
ESS_USHORT_T	DataDirection	リモート接続情報(ソース側またはターゲット側のいずれか?)。
ESS_SVRNAME_T	SvrName	パーティション定義の他の側のホスト。
ESS_APPNAME_T	AppName	パーティション定義の他の側のアプリケーション。
ESS_DBNAME_T	DbName	パーティション定義の他の側のデータベース。メタデータ変更情報。
ESS_TIME_T	LastMetaUpdateTime	メタデータの前回更新時刻。
次のフィールドは、複製データ・ターゲットにのみ適用されます		
ESS_TIME_T	LastRefreshTime	ターゲットのデータの前回リフレッシュ時刻。
ESS_BOOL_T	partitionUpdatable	複製されたデータの変更は許可されていますか?
次のフィールドは、複製データ・ソースにのみ適用されます		
ESS_BOOL_T	IncrRefreshAllowed	変更されたデータのみをリフレッシュできますか?
ESS_TIME_T	LastUpdateTime	パーティション内のデータの前回変更時刻。

操作タイプ定数

```
   #define ESS_PARTITION_OP_REPLICATED   0x0001
#define ESS_PARTITION_OP_LINKED     0x0002
#define ESS_PARTITION_OP_TRANSPARENT  0x0004
#define ESS_PARTITION_OP_ALLTYPES    (ESS_PARTITION_OP_REPLICATED |
                  ESS_PARTITION_OP_LINKED |
                  ESS_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   #define ESS_PARTITION_DATA_SOURCE    0x0001
#define ESS_PARTITION_DATA_TARGET    0x0002
#define ESS_PARTITION_DATA_BOTH     (ESS_PARTITION_DATA_SOURCE |
                  ESS_PARTITION_DATA_TARGET)
```

ESS_PART_REPL_T

共有パーティションにクエリーを行います。

```
   typedef struct ESS_PART_REPL_T
{
 ESS_LONG_T        lAreaCount;
 ESS_BOOL_T        bUpdatedOnly;
 ESS_PPART_CONNECT_INFO_T pHostDatabase;
} ESS_PART_REPL_T, *ESS_PPART_REPL_T, **ESS_PPPART_REPL_T;
```

データ型	フィールド	説明
ESS_LONG_T	lPartitionCount	リフレッシュするパーティションの数(-1 == ALL)
ESS_BOOL_T	bUpdatedOnly	前回のリフレッシュ操作以降、ソースで変更されたセルのみをリフレッシュします。
161 ページの「ESS_PART_CONNECT_INFO_T」	pHostDatabase	パーティション指定の配列。

ESS_PARTDEF_INVALID_T

これは、共有パーティション確認構造体です。

```
   typedef struct ESS_PARTDEF_INVALID_T
{
 ESS_USHORT_T error_type;
 ESS_ULONG_T  line_number;
 ESS_ULONG_T  overlap_number;
 ESS_CHAR_T   member_name[ESS_MBRNAMELEN];
 ESS_CHAR_T   error_message[ESS_LINELEN];
} ESS_PARTDEF_INVALID_T, *ESS_PPARTDEF_INVALID_T, **ESS_PPPARTDEF_INVALID_T;
```

データ型	フィールド	説明
ESS_USHORT_T	error_type	下に示したエラー定数のいずれかになります。
ESS_ULONG_T	line_number	エラーのある行の行番号。 パーティションの定義: 行番号 グローバル・マップ: 行番号 スライス・マップ: スライス番号。
ESS_ULONG_T	overlap_number	オーバーラップされたスライスの場合、スライス番号。オーバーラップされたパーティションの場合、パーティション番号。
ESS_CHAR_T	member_name[ESS_MBRNAMELEN]	エラーのあるメンバー名(ルールのマッピング専用)。
ESS_CHAR_T	error_message[ESS_LINELEN]	下に示したエラー定数のいずれかになります。

エラー定数

```
   define ESS_PARTITION_DEF_ERROR      = 1
define ESS_PARTITION_GLOBAL_MAP_ERROR   = 2
define ESS_PARTITION_AREA_MAP_ERROR    = 3
define ESS_PARITITON_AREA_OVERLAP_ERROR  = 4
define ESS_PARTITION_OVERLAP_ERROR    = 5
define ESS_PARTITION_CELLCOUNT_MISMATCH  = 6
define ESS_PARTITION_TYPE_CONFLICT    = 8
define ESS_PARTITION_DEFAULT_LOGIN_ERROR = 9
define ESS_PARTITION_INVALID_USER     = 10
define ESS_PARTITION_INVALID_PW      = 11
```

ESS_PARTDEF_CONNECT_T

接続情報を保持しています。

```
   typedef struct ESS_PARTDEF_CONNECT_T
{
 ESS_CHAR_T   svrname[ESS_SVRNAMELEN];
 ESS_CHAR_T   appname[ESS_APPNAMELEN];
 ESS_CHAR_T   dbname[ESS_DBNAMELEN];
 ESS_CHAR_T   username[ESS_USERNAMELEN];
 ESS_CHAR_T   password[ESS_PASSWORDLEN];
} ESS_PARTDEF_CONNECT_T, *ESS_PPARTDEF_CONNECT_T, **ESS_PPPARTDEF_CONNECT_T;
```

データ型	フィールド	説明
ESS_CHAR_T	svrname [ESS_SVRNAMELEN]	サーバー名。
ESS_CHAR_T	appname [ESS_APPNAMELEN]	アプリケーション名。
ESS_CHAR_T	dbname [ESS_DBNAMELEN]	データベース名。
ESS_CHAR_T	username [ESS_USERNAMELEN]	管理者ユーザー名。
ESS_CHAR_T	password [ESS_PASSWORDLEN]	管理者パスワード。

ESS_PARTDEF_MAP_T

マッピング情報を保持しています。

```
   typedef struct ESS_PARTDEF_MAP_T
{
 ESS_ULONG_T   mbr_count;
 ESS_STR_T    *src_mbrs;
 ESS_STR_T    *dest_mbrs;
} ESS_PARTDEF_MAP_T, *ESS_PPARTDEF_MAP_T, **ESS_PPPARTDEF_MAP_T;
```

データ型	フィールド	説明
ESS_ULONG_T	mbr_count	配列の再マッピングのサイズ。
ESS_STR_T	src_mbrs	ソースのメンバー名の配列。
ESS_STR_T	dest_mbrs	ターゲットのメンバー名の配列。

ESS_PARTDEF_T

パーティション定義を含んでいます。

```
   typedef struct ESS_PARTDEF_T
{
 ESS_PARTDEF_CONNECT_T    connection;
 ESS_STR_T          description;
 ESS_PARTDEF_AREAS_T     shape_defn;
 ESS_PARTDEF_TYPE_T     typedata;
 ESS_ULONG_T         serialno;
 ESS_TIME_T          meta_last_updated;
} ESS_PARTDEF_T, *ESS_PPARTDEF_T, **ESS_PPPARTDEF_T;
```

データ型	フィールド	説明
165 ページの「ESS_PARTDEF_CONNECT_T」	connection	接続情報。
ESS_STR_T	description	パーティションのユーザーの説明。
166 ページの「ESS_PARTDEF_AREAS_T」	shape_defn	シェイプの定義。
167 ページの「ESS_PARTDEF_TYPE_T」	typedata	タイプ別データ。
ESS_ULONG_T	serialno	共有パーティションの 1 から始まる ID。
ESS_TIME_T	meta_last_updated	このパーティションに影響を与える前回の再構築。

ESS_PARTDEF_AREAS_T

形状定義を保持しています。形状は複数のスライスから構成されています。

```
   typedef struct ESS_PARTDEF_AREAS_T
{
 ESS_USHORT_T  slice_count;
 ESS_STR_T   *slices;
} ESS_PARTDEF_AREAS_T, *ESS_PPARTDEF_AREAS_T, **ESS_PPPARTDEF_AREAS_T;
```

データ型	フィールド	説明
ESS_USHORT_T	slice_count	スライスの数。

データ型	フィールド	説明
ESS_STR_T	slices	スライス定義文字列の配列。

ESS_PARTDEF_TYPE_T

パーティション・タイプに固有の情報を保持します。

```
   typedef struct ESS_PARTDEF_TYPE_T
{
 ESS_USHORT_T     operation_type;
 ESS_USHORT_T     direction_type;
 ESS_USHORT_T     meta_direction_type;
 ESS_PARTDEF_MAP_T   area_map;
 ESS_PARTDEF_MAP_T  *slice_maps;
 ESS_TIME_T      last_refreshed;
 ESS_BOOL_T      incr_refresh;
 ESS_BOOL_T      updatable;
 ESS_CHAR_T      defaultuser[ESS_USERNAMELEN];
 ESS_CHAR_T      defaultpass[ESS_PASSWORDLEN];
} ESS_PARTDEF_TYPE_T, *ESS_PPARTDEF_TYPE_T, **ESS_PPPARTDEF_TYPE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	operation_type	下に示した操作タイプ定数のいずれかになります。
ESS_USHORT_T	direction_type	下に示した方向定数のいずれかになります。 **注：** SVR: のマークがあるフィールドは、サーバー・コードでのみ変更する必要があります。
ESS_USHORT_T	meta_direction_type	下に示した方向定数の 1 つによって識別される、メタデータのソース。
次のフィールドは複製ソースに適用されます		
ESS_BOOL_T	incr_refresh	SVR: 増分リフレッシュの可否
次のフィールドはすべてのターゲットに適用されます		
165 ページの「ESS_PARTDEF_MAP_T」	partition_map	メイン共有パーティション・メンバー・マップ。
165 ページの「ESS_PARTDEF_MAP_T」	slice_maps	スライス別のマップ。
次のフィールドは複製ターゲットに適用されます		
ESS_TIME_T	last_refreshed	SVR: 最終リフレッシュの時刻。
ESS_BOOL_T	updatable	ターゲットのデータは更新可能?

データ型	フィールド	説明
次のフィールドはリンクのターゲットに適用されます		
ESS_CHAR_T	defaultuser [ESS_USERNAMELEN]	デフォルトのユーザー名
ESS_CHAR_T	defaultpass [ESS_PASSWORDLEN]	デフォルトのパスワード

```
   define ESS_PARTITION_OP_REPLICATED  0x0001
define ESS_PARTITION_OP_LINKED    0x0002
define ESS_PARTITION_OP_TRANSPARENT  0x0004
define ESS_PARTITION_OP_ALLTYPES    (ESS_PARTITION_OP_REPLICATED |
                  ESS_PARTITION_OP_LINKED |
                  ESS_PARTITION_OP_TRANSPARENT)
```

```
   define ESS_PARTITION_DATA_SOURCE    0x0001
define ESS_PARTITION_DATA_TARGET    0x0002
define ESS_PARTITION_DATA_BOTH     (ESS_PARTITION_DATA_SOURCE |
                  ESS_PARTITION_DATA_TARGET)
```

ESS_PARTHDR_T

Essbase データベースおよびアプリケーションを指定します。

```
   typedef struct ESS_PARTHDR_T
{
 ESS_SVRNAME_T     zServer;
 ESS_APPNAME_T     zApplication;
 ESS_DBNAME_T     zDatabase;
 ESS_USERNAME_T     zUser;
 ESS_TIME_T      tTime;
} ESS_PARTHDR_T, *ESS_PPARTHDR_T, *ESS_PPPARTHDR_T;
```

データ型	フィールド	説明
ESS_SVRNAME_T	zServer	サーバー名。
ESS_APPNAME_T	zApplication	アプリケーション名。
ESS_DBNAME_T	zDatabase	データベース名。
ESS_USERNAME_T	zUser	ユーザー名。
ESS_TIME_T	tTime	このパーティションに影響を与える前回の再構築。

ESS_PARTOTL_CHANGE_API_T

```
   typedef struct ESS_PARTOTL_CHANGE_API_T
{
 ESS_ULONG_T          ulDimensionCount;
 ESS_PPARTOTL_DIMCHG_API_T   pDimchg;
 ESS_ULONG_T          ulAliasTableCount;
 ESS_PPARTOTL_NAMEMAP_API_T   pAliasTableChg;
}  ESS_PARTOTL_CHANGE_API_T, *ESS_PPARTOTL_CHANGE_API_T,
**ESS_PPPARTOTL_CHANGE_API_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulDimensionCount	次元の変更数。
172 ページの「ESS_PARTOTL_DIMCHG_API_T」	pDimchg	次元変更のリンク・リストを指すポインタ。
ESS_ULONG_T	ulAliasTableCount	別名テーブルの変更数。
177 ページの「ESS_PARTOTL_NAMEMAP_API_T」	pAliasTableChg	変更テーブルのリンク・リスト。

注意

ESS_PARTOTL_CHANGE_API_T 構造体は次元ごとにデータベース・アウトラインの変更を分類します。この構造体は EssSmDbOtlRestruct()の呼出しで渡されます。アウトラインの変更は、次元の一連の変更と、別名テーブルの一連の変更から構成されます。次元の変更は、pDimChg が示すリンク・リストとして渡されます。リンク・リスト内の各アイテムは、次元に対する変更を表します。またメンバーの変更へのリンク・リストを指すルート・ポインタである pMemberChange も含まれます。

別名テーブルの変更は、pAliasTableChg が指すリンク・リストとして渡されます。リンク・リスト内の各アイテムは、別名テーブル内の変更を示します。現在、追加および削除の操作のみサポートされています。別名テーブルの変更操作を下に示します

別名テーブルを削除すると、変更済のレコードが別名テーブルの削除を示します。別名テーブルには、削除される別名に関する変更レコードはありません。別名の変更はメンバー更新として記録されます。別名の変更は別名テーブルのステータスとは関係なく反映されます。つまり、別名テーブルがアクティブである必要はありません。

別名テーブルの名前変更は、古い名前の別名テーブルを削除して、新しい名前で別名テーブルを追加することを意味しています。名前が変更された別名テーブル内の別名は新しい別名です。

ESS_PARTOTL_CHG_FILE_T

メタデータ変更ファイルを指定します。

```
   typedef struct ESS_PARTOTL_CHG_FILE_T
{
```

```
  ESS_USHORT_T  usFileNum;
  ESS_PSTR_T   ppszFileName;
} ESS_PARTOTL_CHG_FILE_T, *ESS_PPARTOTL_CHG_FILE_T, **ESS_PPPARTOTL_CHG_FILE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usFileNum	メタ変更ファイルの数。
ESS_PSTR_T	ppszFileName	メタ変更ファイル名の配列。

ESS_PARTOTL_DIM_ATTRIB_API_T

指定された次元の属性を指定します。

```
   typedef struct ESS_PARTOTL_DIM_ATTRIB_API_T
{
  ESS_USHORT_T           usDimType;
  ESS_USHORT_T           usDimTag;
  ESS_ULONG_T           ulOldDimNo;
  ESS_ULONG_T           ulNewDimNo;
  ESS_ULONG_T           ulNamedLevNum;
  ESS_PARTOTL_NAMED_GENLEV_API_T  *pNamedLev;
  ESS_ULONG_T           ulNamedGenNum;
  ESS_PARTOTL_NAMED_GENLEV_API_T  *pNamedGen;
  ESS_STR_T            pszBasememberName;
  ESS_STR_T            pszOldName;
  ESS_STR_T            pszNewName;
} ESS_PARTOTL_DIM_ATTRIB_API_T, *ESS_PPARTOTL_DIM_ATTRIB_API_T,
**ESS_PPPARTOTL_DIM_ATTRIB_API_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usDimType	下に示した次元タイプ定数のいずれかになります。
ESS_USHORT_T	usDimTag	下に ESS_TTYPE_XXX として示した次元タグ定数のいずれかになります。
ESS_ULONG_T	ulOldDimNo	古いアウトラインの次元番号。
ESS_ULONG_T	ulNewDimNo	新しいアウトラインの次元番号。
ESS_ULONG_T	ulNamedLevNum	名前付きレベルの数。
176 ページの「ESS_PARTOTL_NAMED_GENLEV_API_T」	pNamedLev	名前付きレベル構造体の配列を指すポインタ。
ESS_ULONG_T	ulNamedGenNum	名前付き世代の数。
176 ページの「ESS_PARTOTL_NAMED_GENLEV_API_T」	pNamedGen	名前付き世代構造体の配列を指すポインタ。
ESS_STR_T	pszBasememberName	次元の追加および削除に対する基本メンバー名。

データ型	フィールド	説明
ESS_STR_T	pszOldName	古い次元名。
ESS_STR_T	pszNewName	新しい次元名である pszOldName と pszNewName は、名前の変更の場合にかぎり使用します。次元名の変更によって、次元と、その次元内の一番上のメンバーの両方の名前が変更される点に注意してください

次元タイプ定数(usDimType)

```
   define ESS_DIMTYPE_DENSE  0
define ESS_DIMTYPE_SPARSE 1
```

次元タグ定数(usDimTag)

```
   #define ESS_TTYPE_NONE     0
#define ESS_TTYPE_CCATEGORY   1    /* Accounts - currency ACCOUNTS tag */
#define ESS_TTYPE_CNAME     2    /* Country  - currency COUNTRY tag */
#define ESS_TTYPE_CTIME     3    /* Time - currency TIME tag */
#define ESS_TTYPE_CTYPE     4    /* Type - currency TYPE tag */
#define ESS_TTYPE_CPARTITION   5    /* Currency Partition tag */
#define ESS_TTYPE_ATTRIBUTE   6    /* Attribute tag */
#define ESS_TTYPE_ATTRCALC    7    /* Attribute calc tag(Internal) */
```

ESS_PARTOTL_DIMASSOCCHG_API_T

次元関連付け変更のタイプと同時に、属性次元の名前およびレベルに関する情報を含んでいます。

```
   typedef struct ESS_PARTOTL_DIMASSOCCHG_API_T
{
 ESS_SHORT_T             usDimAssocChgType;
 ESS_CHAR_T             *pszAttrDimName;
 ESS_SHORT_T             usLevel;
 struct ess_partotl_dimassocchg_api_t *pNext;
} ESS_PARTOTL_DIMASSOCCHG_API_T;
```

データ型	フィールド	説明
ESS_SHORT_T	usDimAssocChgType	次元関連付け変更のタイプ
ESS_CHAR_T	pszAttrDimName	属性次元名
ESS_SHORT_T	usLevel	次元関連付けのレベル
ESS_PARTOTL_DIMASSOCCHG_API_T	pNext	次の構造体へのポインタ

ESS_PARTOTL_DIMCHG_API_T

アウトラインの変更、特に次元に対する変更を指定します。

```
   typedef struct ESS_PARTOTL_DIMCHG_API_T
{
 ESS_USHORT_T           usDimChgType;
 ESS_PARTOTL_DIM_ATTRIB_API_T   DimAttribute;
 ESS_PARTOTL_MBR_RSRVD_API_T    MemberReserved;
 ESS_ULONG_T           ulMemberChanges;
 ESS_PPARTOTL_MBRCHG_API_T     pMemberChange;
 ESS_USHORT_T           usAttrType;
 ESS_USHORT_T           usDimAssocChgCnt;
 ESS_PARTOTL_DIMASSOCCHG_API_T  *pDimAssocChg;
 struct ess_partotl_dimchg_api_t *pNext;
} ESS_PARTOTL_DIMCHG_API_T, *ESS_PPARTOTL_DIMCHG_API_T, **ESS_PPPARTOTL_DIMCHG_API_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usDimChgType	下の次元変更(ESS_OTL_DIMCHG_T)の定数から 1 つを選択します
170 ページの「ESS_PARTOTL_DIM_ATTRIB_API_T」	DimAttribute	次元属性
173 ページの「ESS_PARTOTL_MBR_RSRVD_API_T」	MemberReserved	予約済
次の 2 つのフィールドは、ESS_PARTITION_OTLDIM_MBRCHG が次元変更のタイプの 1 つである場合にのみ有効です。		
ESS_ULONG_T	ulMemberChanges	メンバーの変更数
174 ページの「ESS_PARTOTL_MBRCHG_API_T」	pMemberChange	メンバー変更のリンク・リストへのポインタ
ESS_USHORT_T	usAttrType	属性タイプ
ESS_USHORT_T	usDimAssocChgCnt	次元の関連付けの数
171 ページの「ESS_PARTOTL_DIMASSOCCHG_API_T」	pDimAssocChg	次元の関連付けのリンク・リスト
ESS_PARTOTL_DIMCHG_API_T	pNext	次の次元変更に対するポインタ

次元の変更(ESS_OTL_DIMCHG_T)定数

ESS_PARTOTL_DIMCHG_API_T 構造体の usDimChgType フィールドのために、次の定数が定義されています:

```
   ESS_PARTITION_OTLDIM_ADD   /* Add dimensions   */
ESS_PARTITION_OTLDIM_DELETE  /* Delete dimensions */
ESS_PARTITION_OTLDIM_UPDATE  /* Update dimensions */
ESS_PARTITION_OTLDIM_MOVE   /* Move dimensions   */
ESS_PARTITION_OTLDIM_RENAME  /* Rename dimensions */
ESS_PARTITION_OTLDIM_MBRCHG  /*          */
ESS_PARTITION_OTLDIM_ALL   /* All of the above  */
```

ESS_PARTOTL_MBR_RSRVD_API_T

予約済のメンバー操作を指定します。

```
   typedef struct ESS_PARTOTL_MBR_RSRVD_API_T
{
 ESS_BOOL_T           breject;
 ESS_PARTOTL_OSN_RELATIVES_API_T *pSrcRelatives;
 ESS_PARTOTL_OSN_RELATIVES_API_T *pDstRelatives;
 ESS_VOID_T           *unused;
} ESS_PARTOTL_MBR_RSRVD_API_T, *ESS_PPARTOTL_MBR_RSRVD_API_T,
**ESS_PPPARTOTL_MBR_RSRVD_API_T;
```

データ型	フィールド	説明
ESS_BOOL_T	breject	TRUE の場合、このレコードは除外されます (アウトライン同期の場合のみ)。
177 ページの「ESS_PARTOTL_OSN_RELATIVES_API_T」	*pSrcRelatives	ソースの親と兄弟
177 ページの「ESS_PARTOTL_OSN_RELATIVES_API_T」	*pDstRelatives	宛先の親と兄弟
ESS_VOID_T	*unused	(未使用)

ESS_PARTOTL_MBRASSOCCHG_API_T

属性値と同時に、属性次元およびメンバー名に関する情報を含んでいます。

```
   typedef struct ESS_PARTOTL_MBRASSOCCHG_API_T
{
 ESS_CHAR_T             *pszAttrDimName;
 ESS_CHAR_T             *pszAttrMbrName;
 ESS_CHAR_T             *pszAttrParName;
 ESS_ATTRIBUTEVALUE_T         AttrValue;
 struct ess_partotl_mbrassocchg_api_t *pNext;
} ESS_PARTOTL_MBRASSOCCHG_API_T;
```

データ型	フィールド	説明
ESS_CHAR_T	pszAttrDimName	属性次元名
ESS_CHAR_T	pszAttrMbrName	属性メンバー名
ESS_CHAR_T	pszAttrParName	属性の親の名前
124 ページの「ESS_ATTRIBUTEVALUE_T」	AttrValue	属性値
ESS_PARTOTL_MBRASSOCCHG_API_T	pNext	次の構造体へのポインタ

ESS_PARTOTL_MBRATTR_API_T

メンバー属性情報を保管します。

```
   typedef struct ESS_PARTOTL_MBRATTR_API_T
{
 ESS_STS_T          status;
 ESS_SHORT_T         level;
 ESS_SHORT_T         generation;
 ESS_CHAR_T         *calc;
 ESS_SHORT_T         ucal;
 ESS_USHORT_T        atype;
 ESS_BOOL_T         nocconvert;
 ESS_CHAR_T         *crMbrName;
 ESS_PARTOTL_NAMECHG_API_T  *pUdaChange;
 ESS_PARTOTL_NAMECHG_API_T  *pAliasChange;
}
ESS_PARTOTL_MBRATTR_API_T,*ESS_PPARTOTL_MBRATTR_API_T,**ESS_PPPARTOTL_MBRATTR_API_T;
```

データ型	フィールド	説明
ESS_STS_T	status	メンバー・ステータス。
ESS_SHORT_T	level	レベル番号。
ESS_SHORT_T	generation	世代。
ESS_CHAR_T	calc	計算式。
ESS_SHORT_T	ucalc	このメンバーに対する単項計算記号。
ESS_USHORT_T	atype	ACCOUNT とタグ付けされた次元メンバーの 16 ビット・マスク。これは他では使用されません。デフォルトで、すべてオフです。
ESS_BOOL_T	nocconvert	デフォルト設定は FALSE で、通貨換算が行われます。
ESS_CHAR_T	crMbrName	タグ付き通貨データベース・メンバーの名前 時刻の場合 -- タグ付き時刻メンバー 国の場合 -- タグ付き通貨メンバー 勘定科目の場合 -- タグ付きカテゴリ・メンバー
176 ページの「ESS_PARTOTL_NAMECHG_API_T」	pUdaChange	ユーザー定義属性の変更。
176 ページの「ESS_PARTOTL_NAMECHG_API_T」	pAliasChange	別名の変更。

ESS_PARTOTL_MBRCHG_API_T

メンバー変更操作を指定します。

```
   typedef struct ESS_PARTOTL_MBRCHG_API_T
{
 ESS_ULONG_T           ulOperator;
 ESS_CHAR_T           *pszOperand1;
 ESS_CHAR_T           *pszOperand2;
```

```
  ESS_CHAR_T           *pszOperand3;
  ESS_CHAR_T           *pszOperand4;
  ESS_ULONG_T           ulOperand1;
  ESS_PARTOTL_MBRATTR_API_T    *pMemberAttribute;
  ESS_PARTOTL_MBR_RSRVD_API_T   MemberReserved;
  ESS_ULONG_T           ulMbrAssocChgCnt;
  ESS_PARTOTL_MBRASSOCCHG_API_T  *pMbrAssocChg;
  struct ess_partotl_mbrchg_api_t *pNext;
} ESS_PARTOTL_MBRCHG_API_T,*ESS_PPARTOTL_MBRCHG_API_T, **ESS_PPPARTOTL_MBRCHG_API_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulOperator	後述のメンバー変更(ESS_MBR_CHANGE_T)定数から 1 つを選択します
ESS_CHAR_T	pszOperand1	アルファベットのオペランド 1
ESS_CHAR_T	pszOperand2	アルファベットのオペランド 2
ESS_CHAR_T	pszOperand3	アルファベットのオペランド 3
ESS_CHAR_T	pszOperand4	アルファベットのオペランド 4
ESS_ULONG_T	ulOperand1	指定したメンバーの更新済属性を示すビットフィールドのオペランド。このフィールドは、メンバー変更演算子が ESS_PARTITION_OTLMBR_UPDATE の場合にかぎり使用されます。
173 ページの「ESS_PARTOTL_MBRATTR_API_T」	pMemberAttribute	メンバー属性構造体へのポインタ。削除と名前変更の場合は NULL 値です。
173 ページの「ESS_PARTOTL_MBR_RSRVD_API_T」	MemberReserved	予約済
ESS_ULONG_T	ulMbrAssocChgCnt	メンバーの関連付けの数
173 ページの「ESS_PARTOTL_MBRASSOCCHG_API_T」	pMbrAssocChg	メンバーの関連付けのリンク・リスト
ESS_PARTOTL_MBRCHG_API_T	pNext	次の構造体へのポインタ

メンバー変更(ESS_MBR_CHANGE_T)定数

ESS_PARTOTL_MBRCHG_API_T 構造体の ulOperator フィールドのために、次の定数が定義されています:

```
   ESS_PARTITION_OTLMBR_ADD       /* Add members          */
ESS_PARTITION_OTLMBR_DELETE     /* Delete members         */
ESS_PARTITION_OTLMBR_RENAME     /* Rename members         */
ESS_PARTITION_OTLMBR_MOVE      /* Move members          */
ESS_PARTITION_OTLMBR_UPDATE     /* Update members         */
ESS_PARTITION_OTLMBRATTR_STATUS    /* Status changes         */
ESS_PARTITION_OTLMBRATTR_ALIAS    /* Alias changes         */
ESS_PARTITION_OTLMBRATTR_UCALC    /* Unary calc symbol changes    */
ESS_PARTITION_OTLMBRATTR_ATYPE    /* Account type changes      */
ESS_PARTITION_OTLMBRATTR_CCONVERT  /* Currency conversion flag    */
ESS_PARTITION_OTLMBRATTR_CRMBRNAME  /* Tagged currency database member */
```

```
 ESS_PARTITION_OTLMBRATTR_UDA     /* User defined attribute changes  */
 ESS_PARTITION_OTLMBRATTR_CALC     /* Calc formula changes       */
 ESS_PARTITION_OTLMBRATTR_LEVEL    /* Level number changes       */
 ESS_PARTITION_OTLMBRATTR_GENERATION  /* Generation number changes    */
 ESS_PARTITION_OTLMBRATTR_ATTRIBUTE  /* Attribute changes        */
 ESS_PARTITION_OTLMBRATTR_ALL     /* All of the above         */
```

ESS_PARTOTL_NAMECHG_API_T

名前の変更を記録します。

```
   typedef struct ESS_PARTOTL_NAMECHG_API_T
{
 ESS_USHORT_T        usCount;
 ESS_PPARTOTL_NAMEMAP_API_T pNameMap;
} ESS_PARTOTL_NAMECHG_API_T, *ESS_PPARTOTL_NAMECHG_API_T,
**ESS_PPPARTOTL_NAMECHG_API_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usCount	変更数。
177 ページの「ESS_PARTOTL_NAMEMAP_API_T」	pNameMap	名前マップの配列。

ESS_PARTOTL_NAMED_GENLEV_API_T

レベルまたは世代の名前を指定します。

```
   typedef struct ESS_PARTOTL_NAMED_GENLEV_API_T
{
 ESS_USHORT_T             usOperator;
 ESS_SHORT_T             sGenLev;
 ESS_STR_T              pszName;
 struct ess_partotl_named_genlev_api_t *pNext;
} ESS_PARTOTL_NAMED_GENLEV_API_T, *ESS_PPARTOTL_NAMED_GENLEV_API_T,
**ESS_PPPARTOTL_NAMED_GENLEV_API_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usOperator	下に示した名前操作タイプ定数のいずれかになります。
ESS_SHORT_T	sGenLev	世代番号またはレベル番号。
ESS_STR_T	pszName	世代名またはレベル名。
ESS_PARTOTL_NAMED_GENLEV_API_T	pNext	次の構造体を指すポインタ。

名前操作タイプの定数

```
   #define ESS_NAME_ADD   0x01
```

```
#define ESS_NAME_DELETE 0x02
#define ESS_NAME_UPDATE 0x04
```

ESS_PARTOTL_NAMEMAP_API_T

名前の変更をチャートにします。

```
   typedef struct ESS_PARTOTL_NAMEMAP_API_T
{
 ESS_USHORT_T           usOperator;
 ESS_CHAR_T            *name;
 ESS_CHAR_T            *name2;
 struct ess_partotl_namemap_api_t  *pNext;
}  ESS_PARTOTL_NAMEMAP_API_T, *ESS_PPARTOTL_NAMEMAP_API_T,
**ESS_PPPARTOTL_NAMEMAP_API_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usOperator	下に示した名前操作タイプ定数のいずれかになります。
ESS_CHAR_T	name	uda 名。別名の変更の場合、別名テーブル名。
ESS_CHAR_T	name2	uda の変更では使用しません。別名の変更の場合は別名を使用します。
ESS__PARTOTL_NAMEMAP_API_T	pNext	次の構造体を指すポインタ。

名前操作タイプの定数

```
   #define ESB_NAME_ADD   0x01
#define ESB_NAME_DELETE 0x02
#define ESB_NAME_UPDATE 0x04
```

ESS_PARTOTL_OSN_RELATIVES_API_T

メンバー、親メンバーおよび兄弟メンバーの名前を含んでいます。

```
   typedef struct ESS_PARTOTL_OSN_RELATIVES_API_T
{
 ESS_UCHAR_T      statuses[ESS_PARTOTL_OSN_NUM_RELATIVES];
 ESS_PCHAR_T      names[ESS_PARTOTL_OSN_NUM_RELATIVES];
 ESS_ATTRIBUTEVALUE_T values[ESS_PARTOTL_OSN_NUM_RELATIVES];
} ESS_PARTOTL_OSN_RELATIVES_API_T;
```

データ型	フィールド	説明
ESS_UCHAR_T	statuses	各メンバーのステータスを含む配列

データ型	フィールド	説明
ESS_PCHAR_T	names	各メンバーの名前を含む配列
124 ページの「ESS_ATTRIBUTEVALUE_T」	values	各メンバーの属性値構造体を含む配列

ESS_PARTOTL_OSN_RELATIVES_API_T の定数

```
typedef enum ESS_PARTOTL_OSN_REL_TYPE_API_T (Indices for the
   statuses
   ,

   names
    and
   values
    arrays)
{
ESS_PARTOTL_OSN_MEMBER
ESS_PARTOTL_OSN_PARENT
ESS_PARTOTL_OSN_LSIBLING
ESS_PARTOTL_OSN_RSIBLING
ESS_PARTOTL_OSN_REGION_PARENT
ESS_PARTOTL_OSN_LEVEL_REGION_LSIBLING
ESS_PARTOTL_OSN_LEVEL_REGION_RSIBLING
ESS_PARTOTL_OSN_GENER_REGION_LSIBLING
ESS_PARTOTL_OSN_GENER_REGION_RSIBLING
ESS_PARTOTL_OSN_RESERVED1
ESS_PARTOTL_OSN_RESERVED2
ESS_PARTOTL_OSN_NUM_RELATIVES
}
#define ESS_PARTOTL_OSN_REGION_LSIBLING ESS_PARTOTL_OSN_GENER_REGION_LSIBLING
#define ESS_PARTOTL_OSN_REGION_RSIBLING ESS_PARTOTL_OSN_GENER_REGION_RSIBLING


typedef enum ESS_PARTOTL_OSN_REL_TYPE_API_T (Values for
   statuses
   )
{
ESS_PARTOTL_OSN_REL_NONE
ESS_PARTOTL_OSN_REL_SAME_AS_ADJACENT  /* The name of the region sibling is the
                  same as the name of the sibling. */
ESS_PARTOTL_OSN_REL_SHARED
ESS_PARTOTL_OSN_REL_REAL
}
```

ESS_PARTOTL_QUERY_T

メタデータの変更にクエリーを行います。

```
   typedef struct ESS_PARTOTL_QUERY_T
{
 ESS_PART_CONNECT_INFO_T  HostDatabase;
```

```
 ESS_USHORT_T       usOperationType;
 ESS_USHORT_T,       usDataDirectionType;
 ESS_PARTOTL_QRY_FILTER_T  MetaFilter;
} ESS_PARTOTL_QUERY_T, *ESS_PPARTOTL_QUERY_T, **ESS_PPPARTOTL_QUERY_T;
```

データ型	フィールド	説明
161 ページの「ESS_PART_CONNECT_INFO_T」	HostDatabase	ホスト・データベース。
ESS_USHORT_T	usOperationType	下に示した操作タイプ定数のいずれかになります。
ESS_USHORT_T	usDataDirectionType	下に示した方向タイプ定数のいずれかになります。
179 ページの「ESS_PARTOTL_QRY_FILTER_T」	MetaFilter	名前の詳細定義の基準。

操作タイプ定数

```
   #define ESS_PARTITION_OP_REPLICATED   0x0001
#define ESS_PARTITION_OP_LINKED     0x0002
#define ESS_PARTITION_OP_TRANSPARENT   0x0004
#define ESS_PARTITION_OP_ALLTYPES    (ESS_PARTITION_OP_REPLICATED |
                  ESS_PARTITION_OP_LINKED |
                  ESS_PARTITION_OP_TRANSPARENT)
```

方向タイプの定数

```
   #define ESS_PARTITION_DATA_SOURCE  0x0001
#define ESS_PARTITION_DATA_TARGET  0x0002
```

ESS_PARTOTL_QRY_FILTER_T

メタデータの検索基準を詳細に定義します。

```
   typedef struct ESS_PARTOTL_QRY_FILTER_T
{
 ESS_TIME_T     TimeStamp;
 ESS_ULONG_T     ulDimFilter;
 ESS_ULONG_T     ulMbrFilter;
 ESS_ULONG_T     ulMbrAttrFilter;
} ESS_PARTOTL_QRY_FILTER_T, *ESS_PPARTOTL_QRY_FILTER_T, **ESS_PPPARTOTL_QRY_FILTER_T;
```

データ型	フィールド	説明
ESS_TIME_T	TimeStamp	この時刻以降発生したメタデータの変更のクエリー。
ESS_ULONG_T	ulDimFilter	次元変更を選択するためのビット・フィールド。

データ型	フィールド	説明
ESS_ULONG_T	ulMbrFilter	メンバー変更を選択するためのビット・フィールド。
ESS_ULONG_T	ulMbrAttrFilter	メンバー属性変更を選択するためのビット・フィールド。

メンバー属性変更の定数

```
   #define ESS_PARTITION_OTLMBRATTR_STATUS     0x0001  /* status changes */
#define ESS_PARTITION_OTLMBRATTR_ALIAS      0x0002  /* alias changes */
#define ESS_PARTITION_OTLMBRATTR_UCALC      0x0004  /* unary calc symbol changes */
#define ESS_PARTITION_OTLMBRATTR_ATYPE      0x0008  /* account type changes */
#define ESS_PARTITION_OTLMBRATTR_CCONVERT    0x0010  /* currency conversion flag */
#define ESS_PARTITION_OTLMBRATTR_CRMBRNAME    0x0020  /* tagged currency db member */
#define ESS_PARTITION_OTLMBRATTR_UDA      0x0040  /* user defined attribute changes
*/
#define ESS_PARTITION_OTLMBRATTR_CALC      0x0080  /* calc formula changes */
#define ESS_PARTITION_OTLMBRATTR_LEVEL      0x0100  /* level number changes */
#define ESS_PARTITION_OTLMBRATTR_GENERATION   0x0200  /* generation number changes */
#define ESS_PARTITION_OTLMBRATTR_ALL   (ESS_PARTITION_OTLMBRATTR_STATUS |
                  ESS_PARTITION_OTLMBRATTR_ALIAS |
                  ESS_PARTITION_OTLMBRATTR_UCALC |
                  ESS_PARTITION_OTLMBRATTR_ATYPE |
                  ESS_PARTITION_OTLMBRATTR_CCONVERT |
                  ESS_MBRATTR_CRMBR_NAME |
                  ESS_PARTITION_OTLMBRATTR_UDA |
                  ESS_PARTITION_OTLMBRATTR_CALC |
                  ESS_PARTITION_OTLMBRATTR_LEVEL |
                  ESS_PARTITION_OTLMBRATTR_GENERATION)
#define ESS_ALLCHG    (ESS_PARTITION_OTLMBR_ALL | ESS_DIMCHG_ALL)
```

ESS_PARTOTL_READ_T

メタデータの変更を読み取ります。

```
   typedef struct ESS_PARTOTL_READ_T
{
 ESS_PPARTOTL_CHANGE_API_T  pOtlChg;
 ESS_TIME_T         SourceTime;
} ESS_PARTOTL_READ_T, *ESS_PPARTOTL_READ_T, **ESS_PPPARTOTL_READ_T;
```

データ型	フィールド	説明
169 ページの「ESS_PARTOTL_CHANGE_API_T」	pOtlChg	メタ変更レコード。
ESS_TIME_T	SourceTime	ソースのアウトラインの変更時刻。

ESS_PARTOTL_SELECT_APPLY_T

メタデータの変更を適用します。

```
   typedef struct ESS_PARTOTL_SELECT_APPLY_T
{
 ESS_STR_T         pszFileName;
 ESS_PPARTOTL_CHANGE_API_T  pOtlChg;
 ESS_TIME_T         SourceTime;
} ESS_PARTOTL_SELECT_APPLY_T, *ESS_PPARTOTL_SELECT_APPLY_T,
**ESS_PPPARTOTL_SELECT_APPLY_T;
```

データ型	フィールド	説明
ESS_STR_T	pszFileName	アウトライン変更ファイル名。
169 ページの「ESS_PARTOTL_CHANGE_API_T」	pOtlChg	アウトライン変更レコード (EssPartitionReadOtlChangeFile から)。
ESS_TIME_T	SourceTime	アウトライン変更ソースからのタイムスタンプ。

ESS_PARTOTL_SELECT_CHG_T

メタデータにクエリーを行います。

```
   typedef struct ESS_PARTOTL_SELECT_CHG_T
{
 ESS_STR_T        pszFileName;
 ESS_PARTOTL_QRY_FILTER_T QueryFilter;
} ESS_PARTOTL_SELECT_CHG_T, *ESS_PPARTOTL_SELECT_CHG_T, **ESS_PPPARTOTL_SELECT_CHG_T;
```

データ型	フィールド	説明
ESS_STR_T	pszFileName	メタ変更ファイル名。
179 ページの「ESS_PARTOTL_QRY_FILTER_T」	QueryFilter	条件を満たすレコードのみを読み取ります。

ESS_PARTSLCT_T

指定されたサイトの共有パーティションにクエリーを行います。

```
   typedef struct ESS_PARTSLCT_T
{
 ESS_USHORT_T    usOperationTypes;
 ESS_USHORT_T    usDirectionTypes;
 ESS_USHORT_T    usMetaDirectionTypes;
} ESS_PARTSLCT_T, *ESS_PPARTSLCT_T, **ESS_PPPARTSLCT_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usOperationTypes	下に示した操作タイプ定数のいずれかになります。

データ型	フィールド	説明
ESS_USHORT_T	usDirectionTypes	下に示した方向定数のいずれかになります。

操作タイプ定数

```
   #define ESS_PARTITION_OP_REPLICATED   0x0001
#define ESS_PARTITION_OP_LINKED     0x0002
#define ESS_PARTITION_OP_TRANSPARENT   0x0004
#define ESS_PARTITION_OP_ALLTYPES    (ESS_PARTITION_OP_REPLICATED |
                  ESS_PARTITION_OP_LINKED |
                  ESS_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   #define ESS_PARTITION_DATA_SOURCE    0x0001
#define ESS_PARTITION_DATA_TARGET    0x0002
#define ESS_PARTITION_DATA_BOTH     (ESS_PARTITION_DATA_SOURCE |
                  ESS_PARTITION_DATA_TARGET)
```

ESS_PARTSLCT_VALIDATE_T

確認するパーティションを指定します。

```
   typedef struct ESS_PARTSLCT_VALIDATE_T
{
 ESS_USHORT_T       usLoc;
 ESS_STR_T        pszFileName;
 ESS_PART_DEFINED_T    Part;
} ESS_PARTSLCT_VALIDATE_T, *ESS_PPARTSLCT_VALIDATE_T, **ESS_PPPARTSLCT_VALIDATE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usLoc	ESS_FILE_CLIENT または ESS_FILE_SERVER
ESS_STR_T	pszFileName	パーティション定義ファイル名。
162 ページの「ESS_PART_DEFINED_T」	Partition	確認対象のパーティション。

ESS_PERF_ALLOC_ARG_T

この構造体は、割当てまたはカスタム計算のエラーが発生した場所に関する情報を含んでいます。

```
   typedef_enum ESS_PERF_ALLOC_ARG_T
{
 ESS_PERF_ALLOC_ARG_NA,0,
 ESS_PERF_ALLOC_ARG_POV,1,
```

```
  ESS_PERF_ALLOC_ARG_AMOUNT,2,
  ESS_PERF_ALLOC_ARG_AMOUNTCONTEXT,3,
  ESS_PERF_ALLOC_ARG_AMOUNTTIMESPAN,4,
  ESS_PERF_ALLOC_ARG_TARGET,5,
  ESS_PERF_ALLOC_ARG_TARGETTIMESPAN,6,
  ESS_PERF_ALLOC_ARG_TARGETTIMESPANOPTION,7,
  ESS_PERF_ALLOC_ARG_OFFSET,8,
  ESS_PERF_ALLOC_ARG_DEBITMEMBER,9,
  ESS_PERF_ALLOC_ARG_CREDITMEMBER,10,
  ESS_PERF_ALLOC_ARG_RANGE,11,
  ESS_PERF_ALLOC_ARG_EXCLUDEDRANGE,12,
  ESS_PERF_ALLOC_ARG_BASIS,13,
  ESS_PERF_ALLOC_ARG_BASISTIMESPAN,14,
  ESS_PERF_ALLOC_ARG_BASISTIMESPANOPTION,15,
  ESS_PERF_ALLOC_ARG_ALLOCATIONMETHOD,16,
  ESS_PERF_ALLOC_ARG_SPREADSKIPOPTION,17,
  ESS_PERF_ALLOC_ARG_ZEROAMOUNTOPTION,18,
  ESS_PERF_ALLOC_ARG_ZEROBASISOPTION,19,
  ESS_PERF_ALLOC_ARG_NEGATIVEBASISOPTION,20,
  ESS_PERF_ALLOC_ARG_ROUNDMETHOD,21,
  ESS_PERF_ALLOC_ARG_ROUNDDIGITS,22,
  ESS_PERF_ALLOC_ARG_ROUNDTOLOCATION,23,
  ESS_PERF_ALLOC_ARG_SCRIPT,24,
  ESS_PERF_ALLOC_ARG_SOURCEREGION,25,
  ESS_PERF_ALLOC_ARG_GROUPID,26,
  ESS_PERF_ALLOC_ARG_RULEID,27
} ESS_PERF_ALLOC_ARG_T;
```

ESS_PERF_ALLOC_ERROR_T

この構造体は割当て関数によって戻される警告およびエラーに関する情報を戻します。この情報は、呼出し関数によってエラーがある引数およびエラーが発生した行番号とトークンを判別するために使用されます。エラー構造体は一部の警告またはエラーのみで生成されます。messageNumber は、構造体がどのメッセージと対応しているかを示します。複数のメッセージが同じ番号を持っている場合、対応するエラー構造体(ある場合)はメッセージが指定された順番と同じになります。

```
   typedef struct ESS_PERF_ALLOC_ERROR_T
{
  struct ESS_PERF_ALLOC_ERROR_T   *nextError;
  ESS_ULONG_T           messageNumber;
  ESS_PERF_ALLOC_ARG_T       argument;
  ESS_ULONG_T           lineNumber;
  ESS_CHAR_T           token[8192];
} ESS_PERF_ALLOC_ERROR_T;
```

データ型	フィールド	説明
ESS_PERF_ALLOC_ERROR_T	nextError	次のエラー構造体へのポインタ(ある場合)

データ型	フィールド	説明
ESS_ULONG_T	messageNumber	対応するエラーまたは警告メッセージの番号
ESS_PERF_ALLOC_ARG_T	(引数)	エラーを含むパラメータを示します
ESS_ULONG_T	lineNumber	エラーを含む引数の行を示します。0 の場合、これは該当しません。
ESS_CHAR_T	トークン(token)	解析エラーを含む引数の部分を示します。該当しない場合は空です

ESS_PERF_ALLOC_T

この構造体には、割当ての実行に使用される情報が格納されます。

```
   typedef struct ESS_PERF_ALLOC_T
{
  ESS_STR_T               pov;
  ESS_STR_T               amount;
  ESS_STR_T               amountContext;
  ESS_STR_T               amountTimeSpan;
  ESS_STR_T               target;
  ESS_STR_T               targetTimeSpan;
  ESS_ALLOCATION_TARGETTIMESPAN_OPTION  targetTimeSpanOption;
  ESS_STR_T               offset;
  ESS_STR_T               debitMember;
  ESS_STR_T               creditMember;
  ESS_STR_T               range;
  ESS_STR_T               excludedRange;
  ESS_STR_T               basis;
  ESS_STR_T               basisTimeSpan;
  ESS_ALLOCATION_BASISTIMESPAN_OPTION  basisTimeSpanOption;
  ESS_ALLOCATION_METHOD_OPTION     allocationMethod;
  ESS_ULONG_T              spreadSkipOption;
  ESS_ALLOCATION_ZEROAMT_OPTION     zeroAmountOption;
  ESS_ALLOCATION_ZEROBASIS_OPTION    zeroBasisOption;
  ESS_ALLOCATION_NEGBASIS_OPTION    negativeBasisOption;
  ESS_ALLOCATION_ROUND_OPTION     roundMethod;
  ESS_STR_T               roundDigits;
  ESS_STR_T               roundToLocation;
  ESS_ULONG64_T             groupID;
  ESS_ULONG64_T             ruleID;
} ESS_PERF_ALLOC_T;
```

データ型	フィールド	説明
ESS_STR_T	pov	データベース内の割当て領域を指定する MDX セット式
ESS_STR_T	amount	割り当てる金額を指定する MDX タプルまたは数値式

データ型	フィールド	説明
ESS_STR_T	amountContext	**オプション:** MDX タプル式: ● amount が数値式である場合に、amount のコンテキストを指定します ● amount がタプルまたは定数である場合は、この引数は空です
ESS_STR_T	amountTimeSpan	**オプション:** amount を割当ての前に集計する際の開始期間を指定する、レベル 0 のメンバーの MDX セット式
ESS_STR_T	target	割当てのターゲットの場所を指定する MDX タプル式
ESS_STR_T	targetTimeSpan	**オプション:** target の期間を指定する MDX セット式; targetTimeSpanOption とともに使用されます
ESS_ALLOCATION_TARGETTIMESPAN_OPTION_T	targetTimeSpanOption	**オプション:** targetTimeSpan メンバーへの値の割当て方法を指定します: ● ESS_ASO_ALLOCATION_TIMESPAN_DIVIDEAMT (分割) ● ESS_ASO_ALLOCATION_TIMESPAN_REPEATAMT (繰返し) ● 空白の場合は無視されます
ESS_STR_T	offset	**オプション:** オフセット項目の場所を指定する MDX タプル式
ESS_STR_T	debitMember	**オプション:** 正の結果値の書込み先を指定する MDX メンバー式。空の場合、借方/貸方処理は実行されません。
ESS_STR_T	creditMember	**オプション:** 負の結果値の書込み先を指定する MDX メンバー式。空の場合、借方/貸方処理は実行されません。
ESS_STR_T	range	割当てのデータベース領域を指定する MDX セット式
ESS_STR_T	excludedRange	**オプション:** range のサブセット(割当てには含まれているが書込み先ではない領域)を指定する MDX セット式
ESS_STR_T	basis	基準値の場所を指定する MDX タプル式。allocationMethod = ESS_ASO_ALLOCATION_METHOD_SPREAD で、spreadSkipOptions = 0 である場合、basis は空にする必要があります。
ESS_STR_T	basisTimeSpan	**オプション:** basis と同時に考慮される期間を指定する MDX セット式。basisTimeSpanOption と一緒に、割当ての基準を決定します。
ESS_ALLOCATION_BASISTIMESPAN_OPTION_T	basisTimeSpanOption	**オプション:** 複数の期間について基準を計算する方法を次のオプションから指定します: ● ESS_ASO_ALLOCATION_TIMESPAN_SPLITBASIS - 各期間について個別に基準値を処理します ● ESS_ASO_ALLOCATION_TIMESPAN_COMBINEBASIS - basisTimeSpan 内の複数の期間において合計によって合成した基準値を割当てに使用します
ESS_ALLOCATION_METHOD_OPTION_T	allocationMethod	割当てメソッド: ● ESS_ASO_ALLOCATION_METHOD_SHARE - 基準値に対する比率に応じて割り当てます ● ESS_ASO_ALLOCATION_METHOD_SPREAD - ターゲット領域全体で均等に割り当てます

データ型	フィールド	説明
ESS_ULONG_T	spreadSkipOption	**オプション:** ● allocationMethod = ESS_ASO_ALLOCATION_METHOD_SHARE の場合、この値は 0 になります。 ● allocationMethod = ESS_ASO_ALLOCATION_METHOD_SPREAD の場合、どの基準値をスキップするかを指定します。次のビット単位引数のうち、1 つ以上を選択してください: ❍ ESS_ASO_ALLOCATION_SPREAD_SKIPMISSING - allocationRange 内のセルのうち、basisMbr が#missing であるすべてのセルを除外します ❍ ESS_ASO_ALLOCATION_SPREAD_SKIPZERO - allocationRange 内のセルのうち、basisMbr が 0 であるすべてのセルを除外します ❍ ESS_ASO_ALLOCATION_SPREAD_SKIPNEGATIVE - allocationRange 内のセルのうち、basisMbr が負の値であるすべてのセルを除外します これらの引数は、たとえば ESS_ASO_ALLOCATION_SPREAD_SKIPZERO | ESS_ASO_ALLOCATION_SPREAD_SKIPNEGATIVE のように、ビット単位で組み合せることができます
ESS_ALLOCATION_ZEROAMT_OPTION_T	zeroAmountOption	amount の値が 0 または#MISSING である場合に実行する処理を指定します: ● ESS_ASO_ALLOCATION_ZEROAMT_DEFAULT - ゼロ値を割り当てます ● ESS_ASO_ALLOCATION_ZEROAMT_NEXTAMT - 次の amount 値にスキップします ● ESS_ASO_ALLOCATION_ZEROAMT_ABORT - 割当て全体を取り消します
ESS_ALLOCATION_ZEROBASIS_OPTION_T	zeroBasisOption	● allocationMethod=ESS_ASO_ALLOCATION_METHOD_SHARE の場合 - basis の集約がゼロであるときに実行する処理を Essbase に指示します ● allocationMethod=ESS_ASO_ALLOCATION_METHOD_SPREAD の場合 - すべての basis 値がスキップされたときに実行する処理を Essbase に指示します オプションを指定します: ● ESS_ASO_ALLOCATION_ZEROBASIS_NEXTAMT - 次の amount 値にスキップします ● ESS_ASO_ALLOCATION_ZEROBASIS_ABORT - 割当てを取り消します

データ型	フィールド	説明
ESS_ALLOCATION_NEGBASIS_OPTION_T	negativeBasisOption	負の basis 値が検出された場合に実行する処理を Essbase に指示します: ● ESS_ASO_ALLOCATION_NEGBASIS_DEFAULT - 通常どおりに計算します ● ESS_ASO_ALLOCATION_NEGBASIS_NEXTAMT - 次の amount 値にスキップします。データは現在の amount 値に割り当てられません。 ● ESS_ASO_ALLOCATION_NEGBASIS_ABORT - 割当てを取り消します。データは書き込まれません。 次の値は、allocationMethod==ESS_ASO_ALLOCATION_METHOD_SHARE の場合にのみ有効です ● ESS_ASO_ALLOCATION_NEGBASIS_ABS - 絶対値を使用します ● ESS_ASO_ALLOCATION_NEGBASIS_MISSING - basis を#missing として処理します ● ESS_ASO_ALLOCATION_NEGBASIS_ZERO - basis をゼロとして処理します
ESS_ALLOCATION_ROUND_OPTION_T	roundMethod	割り当てられた値の丸めメソッドは、次のとおりです: ● ESS_ASO_ALLOCATION_ROUND_NONE - 丸めを実行しません ● ESS_ASO_ALLOCATION_ROUND_DISCARDERRORS - 丸め誤差を無視して、丸めを実行します ● ESS_ASO_ALLOCATION_ROUND_ERRORSTOHIGHEST - 最大割当て値を含むターゲット・セルに丸め誤差を加算して、丸めを実行します ● ESS_ASO_ALLOCATION_ROUND_ERRORSTOLOWEST - 最小割当て値を含むターゲット・セルに丸め誤差を加算して、丸めを実行します ● ESS_ASO_ALLOCATION_ROUND_ERRORSTOLOCATION - 丸め誤差を roundToLocation に加算して、丸めを実行します
ESS_STR_T	roundDigits	roundMethod=ESS_ASO_ALLOCATION_ROUND_NONE の場合、空にする必要があります。MDX 数値式またはタプル式として指定する必要があります。値は、100 から-100 までの整数である必要があります。
ESS_STR_T	roundToLocation	**オプション:** roundMethod=ESS_ASO_ALLOCATION_ROUND_ERRORSTOLOCATION の場合、これは range 内の場所を指定する MDX タプル式です。それ以外の場合は空です
ESS_ULONG64_T	groupID	内部使用のみ。常に 0 を入力してください。
ESS_ULONG64_T	ruleID	内部使用のみ。常に 0 を入力してください。

ESS_PERF_CUSTCALC_T

この構造体は、集約ストレージ・データベースを使用したカスタム計算の実行に使用される情報を保管します。

カスタム計算スクリプトの作成および実行に関する完全な情報については、『Oracle Essbase データベース管理者ガイド』の集約ストレージ・データベースでのカスタム計算および割当ての実行に関する項を参照してください。

```
typedef struct ESS_PERF_CUSTCALC_T
{
  ESS_STR_T    pov;
  ESS_STR_T    script;
  ESS_STR_T    target;
  ESS_STR_T    debitMember;
  ESS_STR_T    creditMember;
  ESS_STR_T    offset;
  ESS_STR_T    sourceRegion;
  ESS_ULONG64_T   groupID;
  ESS_ULONG64_T   ruleID;
} ESS_PERF_CUSTCALC_T;
```

データ型	フィールド	説明
ESS_STR_T	pov	データベース内のスクリプト実行領域を指定する MDX セット式
ESS_STR_T	script	カスタム計算スクリプトのコンテンツ。複数割当ての形式 `Target := Formula` が含まれている必要があります。ここで、Target は MDX タプル式であり、Formula は MDX 数値式です。スクリプトには MDX タプル式および演算操作 (+ ,- , * , /)のみ使用できます。MDX 関数はサポートされていません。
ESS_STR_T	target	**オプション:** pov と計算結果が書き込まれる script の割当てステートメントの左側を組み合せて、場所を指定する MDX タプル式。次元が重複する場合、解決する競合の順番は最初に割当てステートメントで次に target、その次に pov です。
ESS_STR_T	debitMember	**オプション:** 借方メンバーを指定する MDX メンバー式。正の結果が保管されます。空の場合、借方/貸方処理は実行されません。
ESS_STR_T	creditMember	**オプション:** 貸方メンバーを指定する MDX メンバー式。負の結果が保管されます。空の場合、借方/貸方処理は実行されません。
ESS_STR_T	offset	**オプション:** オフセット・エントリ(ある場合)が書き込まれる場所を指定する MDX タプル式
ESS_STR_T	sourceRegion	スクリプトの式の右側で参照されるデータベース領域を示す MDX セット式
ESS_ULONG64_T	groupID	内部使用のみ。常に 0 を入力してください。
ESS_ULONG64_T	ruleID	内部使用のみ。常に 0 を入力してください。

ESS_PROCSTATE_T

非同期操作(計算など)を実行すると、この構造体は **EssGetProcessState()**への呼出しから戻されます。これによって、呼出し元は非同期操作のステータスを判定できます。

注： このリリースの C API では、設定されているのは State フィールドのみです。その他は今後使用するためのフィールドです。

```
   typedef struct ESS_PROCSTATE_T
{
 ESS_USHORT_T Action;
 ESS_USHORT_T State;
 ESS_USHORT_T Reserved1;
 ESS_ULONG_T  Reserved2;
 ESS_ULONG_T  Reserved3;
} ESS_PROCSTATE_T, *ESS_PPROCSTATE_T;
```

データ型	フィールド	説明
ESS_USHORT_T	Action	現在プロセスのアクション(使用されていません)
ESS_USHORT_T	Stat	現在プロセスの状態(終了または進行中)。値: ● ESS_STATE_DONE (0) ● ESS_STATE_INPROGRESS (1) ● ESS_STATE_FINALSTAGE (5)
ESS_USHORT_T	Reserved1	今後の使用に予約
ESS_ULONG_T	Reserved2	今後の使用に予約
ESS_ULONG_T	Reserved3	今後の使用に予約

ESS_RATEINFO_T

この通貨パーティション・レート情報構造体は **EssGetCurrencyRateInfo()**で使用されます。この構造体のフィールドは、API によって変更できません。

```
   typedef struct ESS_RATEINFO_T
{
 ESS_MBRNAME_T MbrName;
 ESS_MBRNAME_T RateMbr [ESS_CRDB_MAXDIMNUM];
} ESS_RATEINFO_T, *ESS_PRATEINFO_T, **ESS_PPRATEINFO_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	MbrName	メンバー名
ESS_MBRNAME_T	RateMbr[ESS_CRDB_MAXDIMNUM]	レート・メンバー名の配列

ESS_REQUESTINFO_T

セッションおよび要求に関する情報の表示、またはその終了に使用できる情報が含まれています。セッションとは、Essbase サーバーに接続されたユーザーがログ

インして、ログアウトするまでの時間を指します。要求とは、アプリケーションの起動やデータベース・アウトラインの再構築など、ユーザーまたは別なプロセスが Essbaseに送信するクエリーを指します。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。

```
   typedef struct ESS_REQUESTINFO_T
{
 ESS_LOGINID_T  LoginId;       user login identification tag
 ESS_USERNAME_T UserName;       user name
 ESS_SVRNAME_T  LoginSourceMachine;  Login machine name
 ESS_APPNAME_T  AppName;       connected application
 ESS_DBNAME_T   DbName;        connected database
 ESS_USHORT_T   DbRequestCode;     Request code
 ESS_DESC_T    RequestString;     Request string
 ESS_TIME_T    TimeStarted;      time started (in seconds)
 ESS_REQ_STATE_T State;         current process state
}  ESS_REQUESTINFO_T, *ESS_PREQUESTINFO_T, **ESS_PPREQUESTINFO_T;
```

データ型	フィールド	説明
ESS_LOGINID_T	LoginId	ユーザーのログイン時にユーザーに割り当てられる一意な番号。
ESS_USERNAME_T	UserName	要求元のユーザー名。
ESS_SVRNAME_T	LoginSourceMachine	セッションまたは要求の送信元のサーバー名
ESS_APPNAME_T	AppName	セッションまたは要求のアクティブなアプリケーション(ある場合)
ESS_DBNAME_T	DbName	セッションまたは要求のアクティブなデータベース(ある場合)
ESS_USHORT_T	DbRequestCode	アクティブなセッションを示す正の整数。例: 774896669
ESS_DESC_T	RequestString	要求のタイプを示す文字列。使用できる値については、次の要求タイプを参照してください。
ESS_TIME_T	TimeStarted	セッションまたは要求が処理された時間(秒単位)
196 ページの「ESS_REQ_STATE_T」	State	現在のセッションまたは要求の状態: 処理中、終了中、終了済のいずれかです。

要求タイプ

- Process xref request
- xref test
- Restructure
- GetCurrencyDb
- SetCurrencyType
- Export
- SQLImport

- SQLRetrieve
- Report
- SQLConnect
- SQLDatabases
- Calculate
- SetDefaultCalcScript
- ListCalcFunc
- VerifyFormula
- LoadAlias
- ListAliases
- DumpAlias
- BuildDimFile
- GetMbrInfo
- TestDriver
- GetSmStats
- OtlQueryMbrs
- OtlQueryAttrib
- CheckAttribute
- List location aliases
- ClearData
- SetCurrencyDb
- GetCurrencyType
- ParExport
- Import
- CancelUpdate
- SpreadsheetOperation
- SQLListDsn
- SQLTables
- ParseCalcScript
- GetDefaultCalcScript
- VerifyJavaSpec
- ListUdfs
- RemoveAlias
- SetAlias
- BuildDimStart

- GetDSInfo
- GetMbrCalc
- GetDimInfo
- PerfCommand
- OtlQueryMbrs
- OtlGetUpdateTime
- PutReplicatedCells
- Create location alias
- Validate
- GetStats
- SetCurrencyType
- GetCurrencyRate
- DataLoad
- StreamDataload
- ClearUserLocks
- SpreadsheetCellOperation
- SQLColumns
- SQLGetDsn
- RunDefaultCalcScript
- CalcStats
- UpdateCdfCdm
- UdfInfo
- ClearAliases
- GetAlias
- BuildDimension
- GetSelectedMbrInfo
- CheckMbrName
- GetAttributeNameSpecs
- GetOtlInfo
- OtlQueryUDAs
- GetAttrInfo
- GetReplicatedCells
- Delete location alias

ESS_REQUESTINFOEX_T

セッションおよび要求に関する情報の表示、またはその終了に使用できる情報が含まれています。セッションとは、Essbase サーバーに接続されたユーザーがログインして、ログアウトするまでの時間を指します。要求とは、アプリケーションの起動やデータベース・アウトラインの再構築など、ユーザーまたは別なプロセスが Essbase に送信するクエリーを指します。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。この構造体は ESS_REQUESTINFO_T に似ていますが、ProviderName および connparam フィールドが追加されています。

```
typedef struct ESS_REQUESTINFOEX_T
{
 ESS_LOGINID_T  LoginId;
 ESS_USERNAME_T  UserName;
 ESS_USERNAME_T  ProviderName;
 ESS_CONNPARAM_T connparam;

 ESS_SVRNAME_T  LoginSourceMachine;
 ESS_APPNAME_T  AppName;
 ESS_DBNAME_T   DbName;
 ESS_USHORT_T   DbRequestCode;
 ESS_DESC_T    RequestString;
 ESS_TIME_T    TimeStarted;
 ESS_REQ_STATE_T State;
}  ESS_REQUESTINFOEX_T, *ESS_PREQUESTINFOEX_T, **ESS_PPREQUESTINFOEX_T;
```

データ型	フィールド	説明
ESS_LOGINID_T	LoginId	ユーザーのログイン時にユーザーに割り当てられる一意な番号
ESS_USERNAME_T	UserName	要求元のユーザー名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: `native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER`
ESS_SVRNAME_T	LoginSourceMachine	セッションまたは要求の送信元のサーバー名
ESS_APPNAME_T	AppName	セッションまたは要求のアクティブなアプリケーション(ある場合)
ESS_DBNAME_T	DbName	セッションまたは要求のアクティブなデータベース(ある場合)
ESS_USHORT_T	DbRequestCode	アクティブなセッションを示す正の整数。例: 774896669
ESS_DESC_T	RequestString	要求のタイプを示す文字列。使用できる値については、193 ページの「ESS_REQUESTINFOEX_T」を参照してください。

データ型	フィールド	説明
ESS_TIME_T	TimeStarted	セッションまたは要求が処理された時間(秒単位)
ESS_REQ_STATE_T	State;	現在のセッションまたは要求の状態: 処理中、終了中、終了済のいずれかです。

要求タイプ

- Process xref request
- xref test
- Restructure
- GetCurrencyDb
- SetCurrencyType
- Export
- SQLImport
- SQLRetrieve
- Report
- SQLConnect
- SQLDatabases
- Calculate
- SetDefaultCalcScript
- ListCalcFunc
- VerifyFormula
- LoadAlias
- ListAliases
- DumpAlias
- BuildDimFile
- GetMbrInfo
- TestDriver
- GetSmStats
- OtlQueryMbrs
- OtlQueryAttrib
- CheckAttribute
- List location aliases
- ClearData
- SetCurrencyDb
- GetCurrencyType
- ParExport

- Import
- CancelUpdate
- SpreadsheetOperation
- SQLListDsn
- SQLTables
- ParseCalcScript
- GetDefaultCalcScript
- VerifyJavaSpec
- ListUdfs
- RemoveAlias
- SetAlias
- BuildDimStart
- GetDSInfo
- GetMbrCalc
- GetDimInfo
- PerfCommand
- OtlQueryMbrs
- OtlGetUpdateTime
- PutReplicatedCells
- Create location alias
- Validate
- GetStats
- SetCurrencyType
- GetCurrencyRate
- DataLoad
- StreamDataload
- ClearUserLocks
- SpreadsheetCellOperation
- SQLColumns
- SQLGetDsn
- RunDefaultCalcScript
- CalcStats
- UpdateCdfCdm
- UdfInfo
- ClearAliases

- GetAlias
- BuildDimension
- GetSelectedMbrInfo
- CheckMbrName
- GetAttributeNameSpecs
- GetOtlInfo
- OtlQueryUDAs
- GetAttrInfo
- GetReplicatedCells
- Delete location alias

ESS_REQ_STATE_T

ESS_REQUESTINFO_T によって使用されます。この構造体は現在のセッションまたは要求の状態に関する情報を戻します。この構造体のフィールドは、API を使用して変更できません。

```
   typedef ESS_USHORT_T         ESS_REQ_STATE_T;
#define ESS_REQ_IN_PROGRESS      0
#define ESS_REQ_TERMINATING      1
#define ESS_REQ_TERMINATED       2
```

データ型	フィールド	説明
ESS_USHORT_T	ESS_REQ_IN_PROGRESS (0)	現在のセッションまたは要求は処理中です。
ESS_USHORT_T	ESS_REQ_TERMINATING (1)	現在のセッションまたは要求は終了中です。
ESS_USHORT_T	ESS_REQ_TERMINATED (2)	現在のセッションまたは要求は終了しています。

ESS_RUNTIMESUBVARS_DESC_T

EssGetRuntimeSubVars API により使用されます。この構造体は、`RtSVList` 引数のデータ型で、計算スクリプト内のランタイム代替変数構造体のリスト(配列)です。各構造体には、ランタイム代替変数のキー/値ペアが含まれます。オプションで、ランタイム代替変数のデータ型とデータ入力制限(たとえば、100 以下の整数)を表す`<RTSV_HINT>rtsv_description</RTSV_HINT>`タグの文字列を各構造体で指定できます。

```
   typedef_struct (ESS_RUNTIMESUBVARS_DESC_T)
{
  (ESS_STR_T,  rtsvName);
  (ESS_STR_T,  rtsvVal);
  (ESS_STR_T,  rtsvDesc);
} (ESS_RUNTIMESUBVARS_DESC_T);
```

データ型	フィールド	説明
ESS_STR_T	rtsvName	ランタイム代替変数の名前
ESS_STR_T	rtsvVal	ランタイム代替変数の値
ESS_STR_T	rtsvDesc	ランタイム代替変数のデータ型およびデータ入力制限(たとえば、100 以下の整数)の説明

関連項目

EssGetRuntimeSubVars

ESS_SECURITY_MODE_T

EssGetEssbaseSecurityMode が使用します。このデータ型は Essbase サーバーのセキュリティ・モードに関する情報を戻します。

```
   typedef ESS_USHORT_T,  ESS_SECURITY_MODE_T;
#define ESS_NATIVE_SECURITY       1
#define ESS_SS_SECURITY     2
```

ESS_SEQID_T

シーケンス ID の配列を含んでいます。

```
   typedef_struct ESS_SEQID_T
{
 ESS_ULONG_T, seq_id_start;
 ESS_ULONG_T, seq_id_upper_start;
 ESS_ULONG_T, seq_id_end;
 ESS_ULONG_T, seq_id_upper_end;
} ESS_SEQID_T;
```

データ型	フィールド	説明
ESS_ULONG_T	seq_id_start	範囲の最初
ESS_ULONG_T	seq_id_upper_start	範囲の上部の最初
ESS_ULONG_T	seq_id_end	範囲の終了
ESS_ULONG_T	seq_id_upper_end	範囲の上部終了

ESS_TIMERECORD_T

```
   typedef struct ESS_TIMERECORD_T
{
 ESS_TIME_T  TimeValue;
 ESS_USHORT_T Seconds;
 ESS_USHORT_T Minutes;
 ESS_USHORT_T Hours;
 ESS_USHORT_T Day;
 ESS_USHORT_T Month;
 ESS_USHORT_T Year;
 ESS_USHORT_T Weekday;
} ESS_TIMERECORD_T, *ESS_PTIMERECORD_T;
```

133 ページの「ESS_DBREQINFO_T」構造体で使用されます。この構造体で示される時間は、通常はサーバー時刻です。フィールドは次のとおりです:

データ型	フィールド	説明
ESS_TIME_T	TimeValue	1/1/70 以降の秒単位の時間値
ESS_USHORT_T	Seconds	分の後の秒。値: 0-59。
ESS_USHORT_T	Minutes	時間の後の分。値: 0-59。
ESS_USHORT_T	Hours	深夜から数えた時間数。値: 0-23。
ESS_USHORT_T	Day	月の日付。値: 1-31。
ESS_USHORT_T	Month	1 月から数えた月数。値: 0-11。1 月=0。
ESS_USHORT_T	Year	1900 年から数えた年数。
ESS_USHORT_T	Weekday	日曜から数えた日数。値: 0-6。日曜=0。

ESS_TRANSACTION_ENTRY_T

含む

```
   typedef_struct ess_transaction_entry_t
{
    ESS_ULONG_T, seq_id;
    ESS_ULONG_T, seq_id_upper;
    ESS_TIME_T, time_start;
    ESS_TIME_T, time_end;
    ESS_USERNAME_T username;
    ESS_UCHAR_T, type;
    ESS_UCHAR_T, state;
    ESS_CHAR_T, reserved1;
    ESS_TRANSACTION_REQSPECIFIC_T, reqSpecDat;
} ESS_TRANSACTION_ENTRY_T
```

データ型	フィールド	説明
ESS_ULONG_T	seq_id	シーケンス ID
ESS_ULONG_T	seq_id_upper	今後使用されるシーケンス ID の上部
ESS_TIME_T	time_start	操作の開始時刻
ESS_TIME_T	time_end	操作の終了時刻
ESS_USERNAME_T	username	実行するユーザー
ESS_UCHAR_T	type	レコード・タイプ
ESS_UCHAR_T	state	このフィールドとクライアントを使用しないでください
ESS_CHAR_T	reserved1	今後の拡張用
ESS_TRANSACTION_REQSPECIFIC_T	reqSpecDat	特定のデータの要求

ESS_TRANSACTION_REPLAY_INP_T

トランザクション再実行に関する情報を含んでいます。

```
   typedef_struct ESS_TRANSACTION_REPLAY_INP_T
{
    ESS_UCHAR_T, InpType;
    ESS_UCHAR_T, reserved1;
    ESS_UCHAR_T, reserved2;
    ESS_UCHAR_T, reserved3;
    union
    {
        ESS_TIME32_T,    InpTime,  value;
        ESS_ULONG_T,    num_seq_id_range,  value;
    }value;
}ESS_TRANSACTION_REPLAY_INP_T);
```

データ型	フィールド	説明
ESS_UCHAR_T	InpType	時間ベースまたはシーケンス ID
ESS_UCHAR_T	reserved1	予約済
ESS_UCHAR_T	reserved2	予約済
ESS_UCHAR_T	reserved3	予約済
ESS_TIME32_T ESS_ULONG_T	InpTime num_seq_id_range	次の値に対するユニオン変数: ● 再実行を開始する時刻 ● 後続するシーケンス ID ベースの入力構造体の数

ESS_TRANSACTION_REQSPECIFIC_T

情報を含んでいます。

```
   typedef_struct ess_transaction_reqspecific_t
{
    ESS_UCHAR_T, ucReqType;
    ESS_UCHAR_T, reserved1;
    ESS_UCHAR_T, reserved2;
    ESS_UCHAR_T, reserved3;
    union
    {
        ESS_FILENAME_T,   calcname,  value;
        ESS_LOG_DATALOAD_T, dataload_info,  value;
        ESS_LOG_DIMBLD_T,     dimbld_info,  value);
        ESS_FILENAME_T,   tmpotlfilename,  value);
    }    value;
} ESS_TRANSACTION_REQSPECIFIC_T;
```

データ型	フィールド	説明
ESS_UCHAR_T	ucReqType	要求タイプ
ESS_UCHAR_T	reserved1	予約済
ESS_UCHAR_T	reserved2	予約済
ESS_UCHAR_T	reserved3	予約済
ESS_FILENAME_T ESS_LOG_DATALOAD_T ESS_LOG_DIMBLD_T ESS_FILENAME_T	calcname dataload_info dimbld_info tmpotlfilename	次の値に対するユニオン変数: ● 計算ファイル名 ● データ・ロードの詳細 ● 構築ロードの詳細 ● 一時アウトライン・ファイル名

ESS_USERAPP_T、ESS_GROUPAPP_T

ユーザーまたはグループ、および特定のアプリケーションに対するアクセス権情報が含まれています。この構造体の Access フィールドのみ、API を使用して変更できます。フィールドは次のとおりです:

```
   typedef struct ESS_USERAPP_T
{
 ESS_USERNAME_T  UserName;
 ESS_APPNAME_T   AppName;
 ESS_ACCESS_T    Access;
 ESS_ACCESS_T    MaxAccess;
} ESS_USERAPP_T, *ESS_PUSERAPP_T, **ESS_PPUSERAPP_T,
 ESS_GROUPAPP_T, *ESS_PGROUPAPP_T, **ESS_PPGROUPAPP_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名またはユーザー・グループ名
ESS_APPNAME_T	AppName	アプリケーション名
ESS_ACCESS_T	Access	ユーザーまたはグループに対して割り当てられたアプリケーションへのアクセス権。このフィールドの値は、次のビット値を任意に組み合せられます: ● ESS_PRIV_NONE ● ESS_PRIV_APPLOAD ● ESS_PRIV_APPDESIGN
ESS_ACCESS_T	MaxAccess	ユーザーまたはグループに割り当てられた、すべてのソースからのアプリケーションへの最大アクセス権

ESS_USERAPPEX_T、ESS_GROUPAPPEX_T

ユーザーまたはグループ、および特定のアプリケーションに対するアクセス権情報が含まれています。この構造体は ESS_USERAPP_T、ESS_GROUPAPP_T に似ていますが、ProviderName、Type および connparam フィールドが追加されています。

```
   typedef struct ESS_USERAPPEX_T
{
 ESS_USERNAME_T  UserName;
 ESS_USERNAME_T  ProviderName;
 ESS_CONNPARAM_T connparam;
 ESS_USHORT_T   Type;
 ESS_APPNAME_T   AppName;
 ESS_ACCESS_T   Access;
 ESS_ACCESS_T   MaxAccess;
} ESS_USERAPPEX_T, *ESS_PUSERAPPEX_T, **ESS_PPUSERAPPEX_T,
 ESS_GROUPAPPEX_T, *ESS_PGROUPAPPEX_T, **ESS_PPGROUPAPPEX_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名またはユーザー・グループ名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: `native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER`
ESS_USHORT_T	Type	構造体のタイプ。このフィールドには次の値が含められます: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
ESS_APPNAME_T	AppName	アプリケーション名

データ型	フィールド	説明
ESS_ACCESS_T	Access	ユーザーまたはグループに対して割り当てられたアプリケーションへのアクセス権。このフィールドの値は、次のビット値を任意に組み合せられます: ● ESS_PRIV_NONE ● ESS_PRIV_APPLOAD ● ESS_PRIV_APPDESIGN
ESS_ACCESS_T	MaxAccess	ユーザーまたはグループに割り当てられた、すべてのソースからのアプリケーションへの最大アクセス権

ESS_USERDB_T、ESS_GROUPDB_T

ユーザーまたはグループ、および特定のデータベースに対するアクセス権情報が含まれています。この構造体内のアクセスおよびフィルタフィールドのみが、API を使用して変更できるフィールドです。フィールドは次のとおりです:

```
   typedef struct ESS_USERDB_T
{
 ESS_USERNAME_T  UserName;
 ESS_APPNAME_T   AppName;
 ESS_DBNAME_T    DbName;
 ESS_ACCESS_T    Access;
 ESS_ACCESS_T    MaxAccess;
 ESS_FTRNAME_T   FilterName;
} ESS_USERDB_T, *ESS_PUSERDB_T, **ESS_PPUSERDB_T,
 ESS_GROUPDB_T, *ESS_PGROUPDB_T, **ESS_PPGROUPDB_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名またはユーザー・グループ名。
ESS_APPNAME_T	AppName	アプリケーション名。
ESS_DBNAME_T	DbName	データベース名。
ESS_ACCESS_T	Access	ユーザーまたはグループに対して割り当てられたデータベースへのアクセス権限。アクセス権限は、管理サービス・インタフェースを使用して設定されます。 このフィールドの値は、次のビット値を任意に組み合せられます: ● ESS_PRIV_NONE ● ESS_PRIV_READ ● ESS_PRIV_WRITE ● ESS_PRIV_CALC ● ESS_PRIV_DBLOAD ● ESS_PRIV_DBDESIGN これらの値は 102 ページの「ビットマスク・データ型(C)」のサブセットです。

データ型	フィールド	説明
ESS_ACCESS_T	MaxAccess	すべてのソースのユーザーまたはグループに割り当てられた、データベースへの最大のアクセス権限。アクセス権限は、管理サービス・インタフェースを使用して設定されます。
ESS_FTRNAME_T	FilterName	割り当てられたデータベース・フィルタの名前(ある場合)。ない場合、最初のバイトは NULL です。

ESS_USERDBEX_T、ESS_GROUPDBEX_T

ユーザーまたはグループ、および特定のデータベースに対するアクセス権情報が含まれています。この構造体は ESS_USERDB_T、ESS_GROUPDB_T に似てていますが、ProviderName、connparam および Type フィールドが追加されています。

```
   typedef struct ESS_USERDBEX_T
{
 ESS_USERNAME_T  UserName;
 ESS_USERNAME_T  ProviderName;
 ESS_CONNPARAM_T connparam;
 ESS_USHORT_T   Type;
 ESS_APPNAME_T   AppName;
 ESS_DBNAME_T   DbName;
 ESS_ACCESS_T   Access;
 ESS_ACCESS_T   MaxAccess;
 ESS_FTRNAME_T   FilterName;
} ESS_USERDBEX_T, *ESS_PUSERDBEX_T, **ESS_PPUSERDBEX_T,
 ESS_GROUPDBEX_T, *ESS_PGROUPDBEX_T, **ESS_PPGROUPDBEX_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	UserName	ユーザー名またはユーザー・グループ名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: `native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER`
ESS_USHORT_T	Type	構造体のタイプ。このフィールドには次の値が含まれます: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
ESS_APPNAME_T	AppName	アプリケーション名
ESS_DBNAME_T	DbName	データベース名

データ型	フィールド	説明
ESS_ACCESS_T	Access	ユーザーまたはグループに対して割り当てられたデータベースへのアクセス権限。アクセス権限は、管理サービス・インタフェースを使用して設定されます。 このフィールドの値は、次のビット値を任意に組み合せられます: ● ESS_PRIV_NONE ● ESS_PRIV_READ ● ESS_PRIV_WRITE ● ESS_PRIV_CALC ● ESS_PRIV_DBLOAD ● ESS_PRIV_DBDESIGN これらの値は 102 ページの「ビットマスク・データ型(C)」のサブセットです。
ESS_ACCESS_T	MaxAccess	すべてのソースのユーザーまたはグループに割り当てられた、データベースへの最大のアクセス権限。
ESS_FTRNAME_T	FilterName	割り当てられたデータベース・フィルタの名前(ある場合)。ない場合、最初のバイトは NULL です。

ESS_USERINFO_T, ESS_GROUPINFO_T

ユーザーまたはグループに関する情報を保管します。一部のフィールドはユーザーに特有であり、グループには使用できません。この構造体の「Access」、「Expiration」、および「PwdChgNow」フィールドのみ、API を使用して変更できます。フィールドは次のとおりです:

注： ロケール固有の拡張ユーザー情報構造体である207 ページの「ESS_USERINFOEX_T」も参照してください。

```
   typedef struct ESS_USERINFO_T
{
 /* The items below are 4.X and above */
 ESS_USERNAME_T  Name;
 ESS_APPNAME_T  AppName;
 ESS_DBNAME_T   DbName;
 ESS_BOOL_T    Login;
 ESS_USHORT_T   Type;
 ESS_ACCESS_T   Access;
 ESS_ACCESS_T   MaxAccess;
 ESS_DATE_T    Expiration;
 ESS_TIME_T    LastLogin;
 ESS_TIME_T    DbConnectTime;
 ESS_USHORT_T   FailCount;
 ESS_LOGINID_T  LoginId;

 /* The items below are 5.X and above */
 ESS_DESC_T    Description;
```

```
 ESS_EMAIL_T   EMailID;
 ESS_BOOL_T    LockedOut;
 ESS_BOOL_T    PwdChgNow;


} ESS_USERINFO_T, *ESS_PUSERINFO_T, **ESS_PPUSERINFO_T,
 ESS_GROUPINFO_T, *ESS_PGROUPINFO_T, **ESS_PPGROUPINFO_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ユーザーまたはグループ名
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)
ESS_DBNAME_T	DbName	現在接続しているデータベースの名前(該当する場合)
ESS_BOOL_T	Login	ログイン・ステータスを示すフラグ(ユーザーのみ)
ESS_USHORT_T	Type	構造体のタイプ(ユーザーまたはグループ)。このフィールドには次の値が含められます: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
ESS_ACCESS_T	Access	ユーザーまたはグループに割り当てられたデフォルトのアクセス権限。値: 次のビット値を任意に組み合せられます: ● ESS_ACCESS_SUPER /* スーパーバイザ、全ビットを設定 */ ● ESS_PRIV_APPCREATE /* アプリケーションの作成/削除権限 */ ● ESS_PRIV_USERCREATE /* ユーザーの作成/削除権限 */
ESS_ACCESS_T	MaxAccess	ユーザーの最大アクセス権(ユーザーのみ。グループのメンバーシップによる個別のアクセス権とアクセス・レベルを含む)。
ESS_DATE_T	Expiration	ユーザーのパスワードの失効日。
ESS_TIME_T	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付(ユーザーのみ)。
ESS_TIME_T	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。EssSetUser では設定できません。
ESS_USHORT_T	FailCount	最後に正常にログインしてからの、失敗したログインの回数(ユーザーのみ)。
ESS_LOGINID_T	LoginId	ユーザーのログイン識別タグ(ユーザーのみ)。
ESS_DESC_T	Description	ユーザー/グループの説明。
ESS_EMAIL_T	EMailID	ユーザー/グループの電子メール・アドレス。
ESS_BOOL_T	LockedOut	ユーザーがロック・アウトされていることを示すフラグ。
ESS_BOOL_T	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ。

ESS_USERINFOID_T, ESS_GROUPINFOID_T

ユーザーまたはグループに関する情報を保管します。この構造体は ESS_USERINFOEX_T に似ていますが、ProviderName および connparam フィールドが追加されています。

```
   typedef struct ESS_USERINFOID_T
{
 ESS_USERNAME_T   Name;
 ESS_USERNAME_T   ProviderName;
 ESS_PASSWORD_T   Password;
 ESS_APPNAME_T    AppName;
 ESS_DBNAME_T    DbName;
 ESS_BOOL_T     Login;
 ESS_USHORT_T    Type;
 ESS_ACCESS_T    Access;
 ESS_ACCESS_T    MaxAccess;
 ESS_DATE_T     Expiration;
 ESS_TIME_T     LastLogin;
 ESS_TIME_T     DbConnectTime;
 ESS_USHORT_T    FailCount;
 ESS_LOGINID_T    LoginId;
 ESS_DESC_T     Description;
 ESS_EMAIL_T     EMailID;
 ESS_BOOL_T     LockedOut;
 ESS_BOOL_T     PwdChgNow;
 ESS_PROTOCOL_T   protocol;
 ESS_CONNPARAM_T   connparam;

} ESS_USERINFOID_T, *ESS_PUSERINFOID_T, **ESS_PPUSERINFOID_T,
 ESS_GROUPINFOID_T, *ESS_PGROUPINFOID_T, **ESS_PPGROUPINFOID_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	ユーザー名
ESS_USERNAME_T	ProviderName	ユーザー・ディレクトリの名前。例: `@Native Directory`
ESS_PASSWORD_T	Password	外部認証済ユーザーのパスワード。Essbase の認証済メカニズムに外部認証済ユーザーを設定する場合にのみ、これを使用します。このパスワードは、サーバーから外部認証済ユーザーの情報を取得するなど、他の状況では無視されます。
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)
ESS_DBNAME_T	DbName	現在接続しているデータベースの名前(該当する場合)
ESS_BOOL_T	Login	ログイン・ステータスを示すフラグ
ESS_USHORT_T	Type	構造体のタイプ。このフィールドには次の値が含まれます: ● ESS_TYPE_USER

データ型	フィールド	説明
ESS_ACCESS_T	Access	ユーザーに割り当てられたデフォルトのアクセス権限。値には、次のビット値を任意に組み合せできます: ● ESS_ACCESS_SUPER /* スーパーバイザ、全ビットを設定 */ ● ESS_PRIV_APPCREATE /* アプリケーションの作成/削除権限 */ ● ESS_PRIV_USERCREATE /* ユーザーの作成/削除権限 */
ESS_ACCESS_T	MaxAccess	ユーザーの最大アクセス権(個別のアクセス権とグループ・メンバーシップによるアクセス・レベルを含む)
ESS_DATE_T	Expiration	ユーザーのパスワードの失効日
ESS_TIME_T	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付
ESS_TIME_T	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。EssSetUser では設定できません。
ESS_USHORT_T	FailCount	最後に正常にログインしてからの、失敗したログインの回数
ESS_LOGINID_T	LoginId	ユーザー・ログイン識別タグ
ESS_DESC_T	Description	ユーザーの説明
ESS_EMAIL_T	EMailID	ユーザーの電子メール・アドレス
ESS_BOOL_T	LockedOut	ユーザーがロック・アウトされていることを示すフラグ
ESS_BOOL_T	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ
ESS_PROTOCOL_T	protocol	外部認証プロトコル。
ESS_CONNPARAM_T	connparam	ディレクトリのユーザーまたはグループを識別する一意の ID 属性。例: native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER

ESS_USERINFOEX_T

ユーザーまたはグループに関する情報を保管します。一部のフィールドはユーザーに特有であり、グループには使用できません。この構造体で API を使用して変更できるフィールドは、Access、Expiration および PwdChgNow フィールドのみです。

この拡張ユーザー情報構造体は、EssGetUser が使用する標準の ESS_USERINFO_T 構造体とは多少異なります(204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」を参照してください)。この拡張構造体は、EssGetUserEx が使用します。

フィールドは次のとおりです:

```
   typedef struct ESS_USERINFOEX_T
{
ESS_USERNAME_T    Name;
```

```
ESS_PASSWORD_T   Password;
ESS_APPNAME_T    AppName;
ESS_DBNAME_T     DbName;
ESS_BOOL_T      Login;
ESS_USHORT_T     Type;
ESS_ACCESS_T     Access;
ESS_ACCESS_T     MaxAccess;
ESS_DATE_T      Expiration;
ESS_TIME_T      LastLogin;
ESS_TIME_T      DbConnectTime;
ESS_USHORT_T     FailCount;
ESS_LOGINID_T    LoginId;
ESS_DESC_T      Description;
ESS_EMAIL_T     EMailID;
ESS_BOOL_T      LockedOut;
ESS_BOOL_T      PwdChgNow;
ESS_PROTOCOL_T   protocol;
ESS_CONNPARAM_T  connparam;
} ESS_USERINFOEX_T, *ESS_PUSERINFOEX_T, **ESS_PPUSERINFOEX_T;
```

データ型	フィールド	説明
ESS_USERNAME_T	Name	外部認証済ユーザー名。
ESS_PASSWORD_T	Password	外部認証済ユーザーのパスワード。Essbase の認証済メカニズムに外部認証済ユーザーを設定する場合にのみ、これを使用します。このパスワードは、サーバーから外部認証済ユーザーの情報を取得するなど、他の状況では無視されます。
ESS_APPNAME_T	AppName	現在接続しているアプリケーションの名前(該当する場合)
ESS_DBNAME_T	DbName	現在接続しているデータベースの名前(該当する場合)
ESS_BOOL_T	Login	ログイン・ステータスを示すフラグ(ユーザーのみ)
ESS_USHORT_T	Type	構造体のタイプ(ユーザーまたはグループ)。このフィールドには次の値が含められます: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
ESS_ACCESS_T	Access	ユーザーまたはグループに割り当てられたデフォルトのアクセス権限。値: 次のビット値を任意に組み合せられます: ● ESS_ACCESS_SUPER /* Administrator, all bits set */ ● ESS_PRIV_APPCREATE /* アプリケーションの作成/削除権限 */ ● ESS_PRIV_USERCREATE /* ユーザーの作成/削除権限 */
ESS_ACCESS_T	MaxAccess	ユーザーの最大アクセス権(ユーザーのみ。グループのメンバーシップによる個別のアクセス権とアクセス・レベルを含む)。
ESS_DATE_T	Expiration	ユーザーのパスワードの失効日。
ESS_TIME_T	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付(ユーザーのみ)。

データ型	フィールド	説明
ESS_TIME_T	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。EssSetUser では設定できません。
ESS_USHORT_T	FailCount	最後に正常にログインしてからの、失敗したログインの回数(ユーザーのみ)。
ESS_LOGINID_T	LoginId	ユーザーのログイン識別タグ(ユーザーのみ)。
ESS_DESC_T	Description	ユーザー/グループの説明。
ESS_EMAIL_T	EMailID	ユーザー/グループの電子メール・アドレス。
ESS_BOOL_T	LockedOut	ユーザーがロック・アウトされていることを示すフラグ。
ESS_BOOL_T	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ。
ESS_PROTOCOL_T	protocol	外部認証プロトコル。
ESS_CONNPARAM_T	connparam	外部認証接続。

ESS_VARIABLE_T

ESS_VARIABLE_T はプライマリ代替変数データ型です。代替変数の値と名前、および変数が定義される Essbase データベース、アプリケーション、サーバーを識別します。

サーバー名はオプションですが、推奨します。サーバー名を指定しない場合、現在のサーバーがデフォルトになります。AppName はオプションです。DbName はオプションですが、存在する場合は、AppName メンバーが必要です。VarName は必須です。VarValue は必須です。

```
   typedef struct ESS_VARIABLE_T
{
 ESS_SVRNAME_T  Server;
 ESS_APPNAME_T  AppName;
 ESS_DBNAME_T   DbName;
 ESS_MBRNAME_T  VarName;
 ESS_CHAR_T    VarValue[ESS_VARVALUELEN];
} ESS_VARIABLE_T, *ESS_PVARIABLE_T, **ESS_PPVARIABLE_T;
```

データ型	フィールド	説明
ESS_SVRNAME_T	Server	変数が定義されているサーバーの名前(オプション)
ESS_APPNAME_T	AppName	変数を制限するアプリケーションの名前
ESS_DBNAME_T	DbName	変数を制限するデータベースの名前。使用する場合、アプリケーションの設定が必要です。
ESS_MBRNAME_T	VarName	代替変数の名前。
ESS_CHAR_T	VarValue[256]	代替変数の値。

8 CのメインAPI関数

この章の内容

C のメイン API 関数のカテゴリ

サブトピック

- C のメイン API 別名テーブル関数
- C のメイン API アプリケーション関数
- C のメイン API 属性関数
- C のメイン API データベース関数
- C のメイン API データベース・メンバー関数
- C のメイン API のドリルスルー関数
- C のメイン API ファイル関数
- C のメイン API グループ管理関数
- C のメイン API 初期化およびログイン関数
- C のメイン API LRO 関数
- C のメイン API のロケーション別名関数
- C のメイン API メモリー割当て関数
- C のメイン API のその他の関数
- C のメイン API オブジェクト関数
- C のメイン API パーティション関数
- C のメイン API パフォーマンス統計関数
- C のメイン API のレポート、更新、計算関数
- C のメイン API ランタイム代替変数の関数
- C のメイン API セキュリティ・フィルタ関数
- C のメイン API 代替変数の関数
- C のメイン API ユーザー管理関数
- C のメイン API のユーザー ID 関数およびグループ ID 関数
- C のメイン APIShared Services 関数
- C のメイン API の Unicode モードの関数

C のメイン API 別名テーブル関数

別名テーブルの関数は、データベースの別名テーブルを管理します。

関数	説明
EssListAliases	アクティブ・データベース内のすべての別名テーブルをリストします。
EssLoadAlias	構造化されたテキスト・ファイルから、アクティブ・データベースの別名テーブルをロードします
EssGetAlias	1 人のユーザーについて、アクティブなデータベースからアクティブな別名テーブル名を取得します。
EssSetAlias	1 人のユーザーについて、アクティブなデータベースにアクティブな別名テーブルを設定します。
EssDisplayAlias	アクティブなデータベースに別名テーブルのコンテンツをダンプします。
EssRemoveAlias	アクティブなデータベースから別名テーブルを削除します。
EssClearAliases	アクティブ・データベースのすべての別名テーブルを消去します。

C のメイン API アプリケーション関数

アプリケーション関数は、新規アプリケーションの作成および既存のアプリケーションの変更、コピー、情報の取得、管理を行います。

関数	説明
EssGetActive	呼出し元の現在のアクティブなアプリケーションとデータベースの名前を取得します。
EssSetActive	呼出し元のアクティブなアプリケーションとデータベースを設定します。
EssClearActive	ユーザーの現在のアクティブなアプリケーションおよびデータベースを消去します。
EssListApplications	呼出し元がアクセスできる、すべてのアプリケーションをリストします。
EssConvertApplicationtoUnicode	非 Unicode モードのアプリケーションを Unicode モードのアプリケーションに変換します。
EssCreateApplication	クライアントまたはサーバー上で、新規アプリケーションを作成します。
EssCreateApplicationEx	Unicode または非 Unicode モードを選択して、新規アプリケーション・タイプを作成します。
EssCreateStorageTypedApplicationEx	データ・ストレージ・モード(ブロックまたは集約)およびアプリケーション・タイプ(Unicode または非 Unicode)を選択して、新規アプリケーションを作成します。
EssDeleteApplication	クライアント上またはサーバー上で、既存のアプリケーションを削除します。
EssRenameApplication	クライアント上またはサーバー上で、既存のアプリケーションの名前を変更します。

関数	説明
EssCopyApplication	クライアント上またはサーバー上の既存のアプリケーションを、関連するすべてのデータベースとオブジェクトも含めて、新規アプリケーションにコピーします。
EssGetApplicationInfoEx	1 つ以上のアプリケーションから情報を取得します
EssGetApplicationState	ユーザーが構成可能なアプリケーションのパラメータが含まれている、アプリケーションの状態構造体を取得します。
EssSetApplicationState	アプリケーションの状態構造体を使用して、ユーザーが構成可能なアプリケーションのパラメータを設定します。
EssGetApplicationInfo	ユーザーが構成不可能なアプリケーションのパラメータが含まれている、アプリケーションの情報構造体を取得します。
EssLoadApplication	サーバー上のアプリケーションを開始します。
EssUnloadApplication	サーバー上のアプリケーションを停止します。

C のメイン API 属性関数

次の C メイン関数は、属性に関するものです。

関数	説明
EssCheckAttributes	指定した属性次元、基本次元、属性メンバーおよび基本メンバーに対する属性のタイプを戻します
EssFreeStructure	文字列タイプの属性情報用に動的に割り当てられたメモリーを解放します
EssGetAssociatedAttributesInfo	指定した基本メンバーに関連付けられている属性メンバーを戻します
EssGetAttributeInfo	指定した属性メンバーまたは属性次元に関する属性情報を戻します
EssGetAttributeSpecifications	アウトラインの属性指定を取得します

C のアウトライン API の782 ページの「C のアウトライン API 属性の関数」に関する説明を参照してください。

C のメイン API データベース関数

データベース関数は、データベース管理タスクを実行し、データベース情報構造体の取得と変更を行います。

関数	説明
EssBeginDataload	アクティブ・データベースに対して更新指定の送信を開始します。
EssBeginDataloadASO	集約ストレージ・データベース上でデータ・ロードを開始します。

関数	説明
EssClearDatabase	アクティブ・データベース内にロードされているすべてのデータを消去します。
EssCommitDatabase	アクティブなデータベースのデータ・ブロックをすべて強制的にディスクへ書き込みます。
EssCopyDatabase	クライアント上またはサーバー上の既存のデータベースを、関連するすべてのデータベースおよびオブジェクトも含めて、新規のデータベースにコピーします。
EssCreateDatabase	クライアントまたはサーバー上でアプリケーション内に新規データベースを作成します。
EssDeleteDatabase	クライアントまたはサーバー上で、アプリケーションから既存のデータベースを削除します。
EssEndDataload	アクティブなデータベースに送信される更新指定の終了をマークします。
EssGetCurrencyRateInfo	アクティブ・データベース・アウトライン内の、タグ付き通貨パーティション次元のすべてのメンバーについてのレート情報が含まれている構造体のリストを取得します。
EssGetDatabaseInfo	ユーザーが構成不可能なデータベースのパラメータが含まれている、データベースの情報構造体を取得します。
EssGetDatabaseInfoEx	1 つ以上のデータベースの情報を取得します。
EssGetDatabaseNote	データベースの最新情報に関するメッセージを取得します。
EssGetDatabaseState	ユーザーが構成可能なデータベースのパラメータが含まれている、データベースの状態構造体を取得します。
EssGetDatabaseStats	データベースに関する統計情報を含む、アクティブなデータベースの統計構造体を取得します。
EssListCurrencyDatabases	呼出し元がアクセスできる、特定のアプリケーション内の通貨データベースをすべてリストします。
EssListDatabases	呼出し元がアクセス可能な、特定のアプリケーション内またはサーバー全体の、すべてのデータベースをリストします。
EssListExistingLoadBuffers	集約ストレージ・データベースの既存のデータ・ロード・バッファを記述する構造体のリストを戻します。
EssLoadBufferInit	一時データ・ロード・バッファを作成します。
EssLoadBufferTerm	EssLoadBufferInit が割り当てた一時データ・ロード・メモリー・バッファを破棄します。
EssLoadDatabase	データベースを起動します。
EssMergeDatabaseData	2 つ以上のデータ・スライスを 1 つのデータ・スライスにマージします。
EssRenameDatabase	クライアントまたはサーバーのデータベースの名前を変更します。
EssSetDatabaseNote	データベースの最新情報に関するメッセージを設定します。

関数	説明
EssSetDatabaseState	データベースの状態構造体を使用して、ユーザーが構成可能なデータベースのパラメータを設定します。
EssUnloadDatabase	サーバー上でアプリケーション内のデータベースを停止します。
EssValidateDB	データの整合性のためにデータベースを検証します。

C のメイン API データベース・メンバー関数

これらの関数は、データベース・メンバーに関する情報を取得し、データベースの次元を構築します。

関数	説明
EssQueryDatabaseMembers	レポートスタイルのクエリーを実行して、選択したデータベース・メンバーの情報をリストします。
EssCheckMemberName	文字列がアクティブ・データベース・アウトライン内で有効なメンバー名であるかどうか確認します。
EssGetMemberInfo	アクティブ・データベース・アウトライン内の、特定のメンバーに関する情報が含まれている構造体を取得します。
EssGetMemberCalc	アクティブ・データベース・アウトライン内の、特定のメンバーの計算式を取得します。
EssGetDimensionInfo	次元に関する情報を取得します。
EssBuildDimension	データ・ファイルおよびルール・ファイルからのアクティブ・データベース内での次元の作成を可能にします。
EssBuildDimFile	この関数は、アクティブ・データベースのアウトラインからのメンバーの追加または削除に使用するデータ・ファイルを作成します。
EssBuildDimStart	この関数は、アクティブ・データベースのアウトラインからメンバーの追加または削除プロセスを開始します。

C のメイン API のドリルスルー関数

次に示すドリルスルー関数は、Oracle ERP および EPM アプリケーション上でホストされている情報にドリルスルーするためのドリルスルー URL を管理します。

- EssCreateDrillThruURL
- EssDeleteDrillThruURL
- EssGetCellDrillThruReports
- EssGetDrillThruURL
- EssListDrillThruURLs
- EssUpdateDrillThruURL

次に示すドリルスルー関数は、接続したリレーショナル・データベースからデータを取得します。

116 ページの「ドリルスルーの定数および構造体の定義」を参照してください。

グリッド API の次のドリルスルー関数を参照してください:

- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo

注： 今後のリリースでは、次の C のメイン API のドリルスルー関数は廃止され、対応するグリッド API 関数に置き換えられます。プログラムでは前述のグリッド API を使用してください。

- EssDTInit
- EssDTOpen
- EssDTClose
- EssDTExit
- EssDTGetData
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTListReports
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetData
- EssDTAPIGetColumns
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

C のメイン API ファイル関数

ファイル関数によって、アプリケーションは定義済のレポート・スクリプト、データ・ファイル、計算スクリプトをアクティブなデータベースに対して使用できます。テキスト・ファイルとバイナリ・ファイルの両方との間で、データのインポートとエクスポートも行います。

関数	説明
EssArchiveBegin	READ-ONLY ステータスに設定して、データベースをアーカイブ操作のために準備します。
EssArchiveEnd	アーカイブ操作の後、データベース・ステータスを READ-WRITE に戻します。
EssCalcFile	ファイルからアクティブなデータベースに対して計算スクリプトを実行します。
EssExport	現在のデータベースからテキスト・ファイルにデータをエクスポートします。
EssImport	テキスト・ファイルおよびその他のソースから現在のデータベースへデータをインポートします。
EssImportASO	異なるソースから集約ストレージ・データベースへデータをインポートします。
EssListDbFiles	指定したインデックスおよびデータ・ファイルに関する情報を取得します
EssReportFile	ファイルからアクティブなデータベースへレポート指定を送信します。
EssSetDefaultCalcFile	計算スクリプト・ファイルからアクティブ・データベースに対してデフォルト計算スクリプトを設定します。
EssUpdateFile	ファイルからアクティブ・データベースに対して更新指定を送信します。
EssUpdateFileEx	すべてのデータ・ロード・エラーを取得して、ファイルからアクティブなデータベースに更新指定を送信します。
EssUpdateFileASO	ファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。
EssUpdateFileASOEx	すべてのデータ・ロード・エラーを取得して、ファイルからアクティブな集約ストレージ・データベースに更新指定を送信します。
EssUpdateFileUTF8ASO	UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。
EssUpdateFileUTF8ASOEx	すべてのデータ・ロード・エラーを取得して、UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに更新指定を送信します。
EssUpdateFileUtf8Ex	すべてのデータ・ロード・エラーを取得して、UTF-8 でエンコードされたファイルからアクティブなデータベースに更新指定を送信します
EssDisplayTriggers	データベースに関連付けられたすべてのトリガーのリストを戻します。
EssMdxTrig	MDX 言語ファイルに含まれる操作に基づいてトリガーを操作します。
EssListSpoolFiles	データベースに関連付けられたすべてのスプール・ファイルのリストを戻します。

関数	説明
EssGetSpoolFile	データベースに関連付けられた特定のスプール・ファイルを戻します。
EssDeleteAllSplFiles	データベースに関連付けられたすべてのスプール・ファイルを削除します。
EssDeleteSplFile	特定のスプール・ファイルを削除します。

C のメイン API グループ管理関数

これらの関数は、グループの作成、グループ属性の設定と変更、および既存のグループに関する情報の取得を行います。

関数	説明
EssListGroups	特定の Essbase サーバーに対してアクセス権を所有しているすべてのグループをリストします。
EssCreateGroup	グループを新規作成します。
EssDeleteGroup	既存のグループを削除します。
EssRenameGroup	既存のグループの名前を変更します。
EssGetGroup	グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。
EssSetGroup	グループ情報構造体を設定します。
EssGetGroupList	グループのメンバーであるユーザーのリスト(またはユーザーが属するグループのリスト)を取得します。
EssSetGroupList	グループ・メンバーであるユーザーのリストを設定します。
EssAddToGroup	ユーザーをグループ・メンバー・リストへ追加します。
EssDeleteFromGroup	グループ・メンバーのリストからユーザーを削除します

C のメイン API 初期化およびログイン関数

この種の関数を使用して API の初期化、Essbase サーバーへのログインとログアウトを行います。バージョン情報の入手、アプリケーションによるローカル・コンテンツの作成と削除も行えます。

関数	説明
EssAutoLogin	ユーザーが Essbase サーバーにログインするためのダイアログ・ボックスを表示します。また、オプションでアクティブなアプリケーションとデータベースを選択します。
EssCreateLocalContext	ローカル API 操作で使用するローカル API コンテキストを作成します。
EssDeleteLocalContext	EssCreateLocalContext で作成されたローカル・コンテキストをリリースします。

関数	説明
EssGetAPIVersion	接続された API クライアント・モジュールの完全なバージョン番号を入手します。
EssGetVersion	接続されている Essbase サーバーの完全なバージョン番号を取得します。
EssInit	API およびメッセージ・データベースを初期化します。
EssLogin	ユーザーを Essbase サーバーにログインさせます。
EssLoginAs	別のユーザーとして Essbase サーバーにログインします。
EssLoginEx	認証トークンを使用して Essbase サーバーにログインします。
EssLoginExAs	認証トークンを使用して、別のユーザーとして Essbase サーバーにログインします。
EssLoginSetPassword	ユーザーをログインさせ、パスワードを変更します。
EssLogout	ユーザーを Essbase サーバーからログアウトさせます。
EssLogoutUser	スーパーバイザまたはアプリケーション・デザイナが他のユーザーを Essbase サーバーから切断できるようにします。
EssLogSize	Essbase サーバー・ログ・ファイル(essbase.log)のサイズ、またはアプリケーション・ログ・ファイル(appname.log)のサイズを戻します。
EssShutdownServer	スーパーバイザがリモートから Essbase サーバーを停止できるようにします。
EssTerm	API を終了し、API で使用されているすべてのシステム・リソースを解放します。
EssValidateHCtx	特定の API コンテキスト・ハンドル(hCtx)を検証します。
EssWriteToLogFile	Essbase サーバー・ログ・ファイル(essbase.log)またはアプリケーション・ログ・ファイル(appname.log)にメッセージを書き込みます。

C のメイン API LRO 関数

これらの関数は、LRO を作成、取得および削除し、LRO に関する情報を戻します。

関数	説明
EssLROAddObject	レポート・オブジェクトを Essbase データベースのデータ・セルにリンクします。
EssLRODeleteCellObjects	Essbase データベースの指定されたデータ・セルにリンクされているすべてのオブジェクトを削除します。
EssLRODeleteObject	Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを削除します。
EssLROGetCatalog	Essbase データベース内の指定したデータ・セルについて、LRO カタログ・エントリのリストを取得します。

関数	説明
EssLROGetCatalogBatch	Essbase データベース内の指定した複数のデータ・セルについて、LRO カタログ・エントリのリストを取得します。
EssLROGetObject	Essbase データベース内のデータ・セルにリンクされているオブジェクトを取得します。
EssLROListObjects	指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトのリストを取得します。
EssLROPurgeObjects	指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトを削除します。
EssLROUpdateObject	サーバーに LRO の更新済バージョンを保管します。

C のメイン API のロケーション別名関数

これらの関数は、ロケーション別名を作成、削除およびリストします。

関数	説明
EssCreateLocationAlias	別名を、ホスト名、アプリケーション名、データベース名、ユーザー・ログイン名およびユーザー・パスワードにマッピングします
EssDeleteLocationAlias	既存のロケーション別名を削除します
EssGetLocationAliasList	すべてのロケーション別名およびそのロケーション別名がマッピングされている名前を戻します

C のメイン API メモリー割当て関数

メモリー割当て関数はメモリー・ブロックの割当て、再配置および解放を行うことで、アプリケーションのメモリーを管理します。

関数	説明
EssAlloc	定義されたメモリー割当ての仕組みを使用して、メモリー・ブロックを割り当てます。
EssRealloc	以前に割り当てたメモリー・ブロックを再割当てします。
EssFree	定義されたメモリー割当ての仕組みを使用して、以前に割り当てたメモリー・ブロックを解放します。

C のメイン API のその他の関数

これらの関数は、非同期プロセスの管理、状態情報の取得、ログ・ファイルのプロセスおよびメッセージの取得を行います。

関数	説明
EssGetProcessState	計算やデータ・インポートなどの非同期プロセスの現在の状態を取得します。
EssCancelProcess	まだ完了していない非同期プロセスを取り消します。
EssGetLogFile	アプリケーション・ログ・ファイルの一部または全部を、サーバーからクライアントにコピーします。
EssDeleteLogFile	サーバー上のアプリケーション・ログ・ファイルを削除します。
EssGetGlobalState	システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を取得します。
EssSetGlobalState	システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を設定します。
EssSetPath	現在のプロセスに対して、ESSBASEPATH 環境変数を設定します。

C のメイン API オブジェクト関数

これらの関数は、オブジェクトを作成、削除、移動、コピーします。また、オブジェクト情報を取得して表示し、オブジェクトへのアクセスを制御します。

関数	説明
EssGetLocalPath	クライアント上のオブジェクト・ファイルの完全なローカル・ファイルを取得します。
EssListObjects	指定したタイプのすべてのオブジェクトをリストします。
EssGetObjectInfo	指定したオブジェクトに関する情報を取得します。
EssGetObject	オブジェクトをサーバーからローカル・ファイルにコピーし、オプションでオブジェクトをロックします。
EssPutObject	オブジェクトをローカル・ファイルからサーバーにコピーし、オプションでオブジェクトのロックを解除します。
EssLockObject	他のユーザーによる更新を防ぐため、サーバー上のオブジェクトをロックします。
EssUnlockObject	サーバー上のロックされたオブジェクトのロックを解除します。
EssCreateObject	オブジェクトを新規作成します。
EssDeleteObject	既存のオブジェクトを削除します。
EssRenameObject	既存のオブジェクトの名前が変更されます。
EssCopyObject	オブジェクトをコピーします。

C のメイン API パーティション関数

次の関数は、データベース上のパーティション操作を管理します。

関数	説明
EssPartitionApplyOtlChangeFile	メタデータの変更をファイルに適用するよう、サーバーに指示します。
EssPartitionApplyOtlChangeFileEx	メタデータの変更をファイルに適用するよう、サーバーに指示します。
EssPartitionApplyOtlChangeRecs	メタデータの変更をレコードに適用するよう、サーバーに指示します。
EssPartitionCloseDefFile	共有パーティションの定義ファイルを閉じます。
EssPartitionFreeDefCtx	共有パーティションのコンテキスト構造体の下に割り当てられたメモリーを動的に解放します。
EssPartitionFreeOtlChanges	ReadMetaChange ルーチンによって割り当てられたメモリーを解放します。
EssPartitionGetAreaCellCount	指定したスライス文字列内のセルの数を戻します。
EssPartitionGetList	現在選択されているデータベースが関与している、パーティション定義のリストを戻します。
EssPartitionGetOtlChanges	指定したソースからメタデータの変更を取り込みます。
EssPartitionGetReplCells	複製パーティションで識別されているすべてのデータ・セルを、ソース・データベースから選択したターゲット・データベースに複製します。
EssPartitionNewDefFile	設定された入力パラメータに基づいて、新しい共有パーティションの定義ファイルを作成して開きます。
EssPartitionOpenDefFile	既存の共有パーティションの定義ファイルを開きます。
EssPartitionPurgeOtlChangeFile	TimeStamp パラメータで指定した時刻より前に行われた、メタデータの変更を除去します。
EssPartitionPutReplCells	複製パーティションで識別されているすべてのデータ・セルを、選択したソース・データベースからターゲット・データベースに複製します。
EssPartitionReadDefFile	複製パーティションで識別されているすべてのデータ・セルを、選択したソース・データベースからターゲット・データベースに複製します。
EssPartitionReadOtlChangeFile	メタ変更をファイルからメモリーに読み取ります。
EssPartitionReplaceDefFile	新しい共有パーティション・ファイルが送信済で、このデータベースの既存のファイルが置換されることをサーバーに通知します。
EssPartitionResetOtlChangeTime	2 つのパーティション、ソースおよび宛先を取得します。ソース・パーティションの「前回のメタ変更」時刻を、宛先のパーティションの「前回のメタ変更」時刻に割り当てます。
EssPartitionValidateDefinition	指定されたパーティション定義の完全な検証を実行します。つまり、1 つのパーティション定義のソース部分とターゲット部分を検証します。新しいパーティション定義および既存のパーティション定義を変更する際に役に立ちます。

関数	説明
EssPartitionValidateLocal	指定されたサーバー上の全パーティション定義の部分的な検証を実行します。たとえば、データベースの再構築後など、メタデータ変更の後にパーティション定義の有効性を確認するのに役に立ちます。
EssPartitionWriteDefFile	共有パーティション定義ファイルの現在のメモリー・バージョンをディスクに書き込みます。

C のメイン API パフォーマンス統計関数

これらの関数は、スレッド、データベースおよびアプリケーションに関する入出力パフォーマンス統計を提供します。

関数	説明
EssDumpPerfStats	パフォーマンス統計テーブルを含む文字配列へのポインタを提供します
EssGetStatBufSize	パフォーマンス統計テーブルに必要なバッファ・サイズへのポインタを提供します
EssResetPerfStats	パフォーマンス統計テーブルの値をゼロにリセットします

C のメイン API のレポート、更新、計算関数

これらの関数は、アクティブ・データベースに対してレポート作成タスク(データの取得)、更新タスク(データのロード)および計算タスク(データの集約)を実行します。

関数	説明
EssReport	レポート指定を単一文字列としてアクティブなデータベースに送信します。
EssBeginReport	アクティブなデータベースへのレポート指定の送信を開始します。
EssEndReport	アクティブなデータベースに送信されるレポート指定の終わりをマークします。
EssUpdate	アクティブ・データベースに対して単一文字列として更新を送信します。
EssUpdateFileASO	ファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。
EssUpdateFileUTF8ASO	UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。
EssBeginUpdate	アクティブ・データベースに対して更新指定の送信を開始します。
EssEndUpdate	アクティブなデータベースに送信される更新指定の終了をマークします。
EssGetString	アクティブ・データベースから文字列データを取得します。
EssSendString	アクティブなデータベースにデータの文字列を送信します。

関数	説明
EssCalc	アクティブ・データベースに対して計算スクリプトを単一の文字列として送信し、オプションで計算スクリプトを実行します。
EssBeginCalc	計算スクリプトの送信を開始し、オプションでアクティブ・データベースに対して計算スクリプトを実行します。
EssEndCalc	アクティブなデータベースに送信される計算スクリプトの終わりをマークします。
EssDefaultCalc	アクティブ・データベースのデフォルト計算を実行します。
EssGetDefaultCalc	アクティブ・データベースのデフォルト計算スクリプトを取得します。
EssListCalcFunctions	使用可能な計算関数の一覧を表示します。
EssSetDefaultCalc	データベースのデフォルト計算スクリプトを設定します。

C のメイン API ランタイム代替変数の関数

これらの関数は、計算スクリプトで参照されたランタイム代替変数を渡し、それらに関する情報を戻します。

関数	説明
EssCalcFileWithRuntimeSubVars	指定したランタイム代替変数のアクティブ・データベースに対して、計算スクリプトを実行します。ランタイム代替変数は、拡張子が.rsv のテキスト・ファイル内で(このファイルはクライアント・コンピュータに存在する必要があります)、またはキー/値ペアの文字列として指定できます。
EssCalcWithRuntimeSubVars	指定したランタイム代替変数(キー/値ペアの文字列として指定)を使用して計算スクリプトを実行します。
EssGetRuntimeSubVars	この関数は、計算スクリプトが実行されるクライアントへのインタフェースとして実装されます。この関数は、指定された計算スクリプトの SET RUNTIMESUBVARS 計算コマンドにおけるランタイム代替変数の宣言で指定済のすべての情報(名前、値および説明)を取得します。

C のメイン API セキュリティ・フィルタ関数

セキュリティ・フィルタ関数は、フィルタの作成、フィルタ・コンテンツの設定、ユーザー・グループへのフィルタの割当て、データベースのフィルタ・リストの表示およびセキュリティ・フィルタに関するその他のデータの取得を行います。

関数	説明
EssListFilters	データベースのすべてのフィルタをリストします。
EssGetFilter	フィルタのコンテンツの取得を開始します。
EssGetFilterRow	フィルタの次の行を取得します。

関数	説明
EssCreateFilter	フィルタを作成します。フィルタのコンテンツの設定を開始します。
EssSetFilter	フィルタを作成または置換します。フィルタのコンテンツの設定を開始します。
EssSetFilterRow	フィルタの次の行を取得します。
EssGetFilterList	フィルタを割り当てられたユーザーのリストを取得します。
EssSetFilterList	フィルタを割り当てられたユーザーのリストを設定します。
EssDeleteFilter	既存のフィルタを削除します。
EssRenameFilter	既存のフィルタの名前を変更します。
EssCopyFilter	既存のフィルタをコピーします。
EssVerifyFilter	指定したデータベースに照らしあわせて、一連のフィルタ行の文字列の構文を確認します。
EssVerifyFilterRow	指定したデータベースに照らしあわせて、単一のフィルタ行の文字列の構文を確認します。

C のメイン API 代替変数の関数

これらの関数は、代替変数を作成、取得および削除し、代替変数に関する情報を戻します。

関数	説明
EssCreateVariable	この関数は代替変数を新規作成します。または同一のサーバー値、アプリケーション値およびデータベース値の変数名がすでに存在する場合は、既存の代替変数を変更します。
EssDeleteVariable	この関数は、代替変数を削除します。
EssGetVariable	この関数は、代替変数の値を取得します。
EssListVariables	この関数は、入力基準に適合する代替変数をすべてリストします。

C のメイン API ユーザー管理関数

ユーザー管理関数では、ユーザーの作成、パスワードの割当て、データベース、アプリケーション、計算スクリプトへのアクセス権の設定を行います。この関数は、ユーザー機能に関する情報を取得するためにも使用できます。

関数	説明
EssListUsers	特定の Essbase サーバーへのアクセス権があるすべてのユーザーをリストします。
EssCreateUser	新規ユーザーを作成します。

関数	説明
EssDeleteUser	既存のユーザーを削除します。
EssRenameUser	既存のユーザー名を変更します。
EssGetUser	ユーザーのセキュリティ情報を含むユーザー情報構造体を取得します。
EssSetUser	ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。
EssResetUser	ユーザーのセキュリティ構造体を最初の状態にリセットします。
EssSetPassword	既存のパスワードを消去して、ユーザーのパスワードを設定します。
EssGetApplicationAccess	アプリケーションへのユーザーのアクセス権情報が含まれているユーザー・アプリケーション・アクセス構造体のリストを取得します。
EssSetApplicationAccess	ユーザー・アプリケーション・アクセス構造体のリストを設定します。
EssGetDatabaseAccess	ユーザー・データベース・アクセス構造体のリストを取得します。
EssSetDatabaseAccess	ユーザー・データベース・アクセス構造体のリストを設定します。
EssGetCalcList	ユーザーがアクセス可能な計算スクリプト・オブジェクトのリストを取得します。
EssSetCalcList	ユーザーが使用可能な計算スクリプト・オブジェクトのリストを設定します。
EssListConnections	現在のアプリケーションおよびデータベースに接続されているすべてのユーザーをリストします。
EssListLogins	現在接続しているユーザーに関する情報をリストします。
EssListRequests	現在の Essbase ユーザー・セッションまたは要求に関する情報をリストします。
EssKillRequest	すべての、または特定の Essbase ユーザー・セッションまたは要求を終了します。
EssListLocks	特定のアプリケーションおよびデータベースに接続されているすべてのユーザーをリストします。
EssRemoveLocks	データベースに対してユーザーが保持しているすべてのデータ・ブロック・ロックを解除します。

C のメイン API のユーザー ID 関数およびグループ ID 関数

ユーザー ID 関数およびグループ ID 関数が機能拡張され、ユーザー・ディレクトリおよび一意の ID 属性を指定して、ディレクトリにホストされているユーザーおよびグループを識別できるようになりました。

関数	説明
EssAddToGroupEx	指定したグループにユーザーを追加します。EssAddToGroup に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssCreateExtGroup	外部ユーザー・ディレクトリにグループを作成します。
EssDeleteFromGroupEx	グループからユーザーを削除します。EssDeleteFromGroup に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssDeleteGroupEx	既存のグループを削除します。EssDeleteGroup に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssDeleteUserEx	ユーザーを削除します。EssDeleteUser に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssGetApplicationAccessEx	アプリケーションへのユーザーまたはグループのアクセス権情報が含まれているユーザーまたはグループのアプリケーション・アクセス構造体のリストを取得します。EssGetApplicationAccess に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssGetDatabaseAccessEx	データベースへのユーザーのアクセス権情報が含まれている、ユーザーのデータベース・アクセス構造体のリストを取得します。EssGetDatabaseAccess に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssGetGroupInfoEx	グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。EssGetGroup に似ていますが、ユーザー・ディレクトリの指定、または一意の ID 属性を受け入れることができます。
EssGetGroupListEx	グループのメンバーであるユーザーのリストまたはユーザーが属するグループのリストを取得します。EssGetGroupList に似ていますが、ユーザー・ディレクトリの指定または一意の ID 属性を受け入れることができます。
EssGetUserInfoEx	ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。EssGetUser に似ていますが、ユーザー・ディレクトリの指定またはの一意の ID 属性を受け入れることができます。
EssKillRequestEx	特定のユーザー・セッションまたは要求を終了します。EssKillRequest に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。
EssListConnectionsEx	現在ログインしているサーバーまたはアプリケーションに接続されているユーザーをすべてリストします。EssListConnections に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。
EssListGroupsInfoEx	ある特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセス権を持つすべてのグループをリストします。EssListGroups に似ていますが、グループ・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。
EssListLocksEx	ある特定のアプリケーションとデータベースに接続しているすべてのユーザーを、それらのユーザーが現在ロックしているデータ・ブロックのカウントとともにリストします。EssListLocks に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。

関数	説明
EssListLoginsEx	現在のセッションのログイン・インスタンスのリストを戻します。EssListLogins に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。
EssListRequestsEx	アクティブなセッションおよび要求に関する情報を戻します。EssListRequests に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。
EssListUsersInfoEx	特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。EssListUsers に似ていますが、ユーザー・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。
EssSetApplicationAccessEx	アプリケーションへのユーザーのアクセス権情報を含むユーザー・アプリケーション・アクセス構造体のリストを設定します。EssSetApplicationAccess に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。
EssSetCalcListEx	指定したユーザーまたはグループにアクセス可能な計算リストを設定します。EssSetCalcList に似ていますが、ユーザー・ディレクトリにホストされているユーザーおよびグループが含まれます。
EssSetDatabaseAccessEx	データベースへのユーザー・アクセスに関する情報を含む、ユーザーのデータベース・アクセス構造体のリストを設定します。EssSetDatabaseAccess に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。
EssSetFilterListEx	フィルタに割り当てられたグループまたはユーザーのリストを設定します。count パラメータはフィルタに割り当てられたグループまたはユーザーの数を制御します。count がゼロの場合、グループまたはユーザーすべてがリストから削除されます。
EssSetGroupListEx	グループのメンバーであるユーザーのリストを設定します。EssSetGroupList に似ていますが、ユーザー・ディレクトリの指定、または一意の ID 属性を受け入れることができます。

C のメイン APIShared Services 関数

Shared Services ユーザー管理により、異なる製品の様々なプロジェクトで作成された、ユーザー・アクセス権とアプリケーションへのアクセシビリティの一括管理が可能です。

次の関数は、Essbase を Shared Services モードへ移行するのに役立ちます。移行の後、Essbase のネイティブ・モードのかわりに Shared Services モードで、ユーザー、グループおよびアプリケーションを管理できます。

関数	説明
EssSetSSSecurityMode	Essbase サーバーと既存のユーザーおよびグループを Oracle Enterprise Performance Management System セキュリティ・モードに移行します。
EssSetUserToSS	1 人のユーザーを EPM System セキュリティ・モードに移行します。
EssSetGroupToSS	1 つのグループを EPM System セキュリティ・モードに移行します。

関数	説明
EssSetUsersToSS	すべてのユーザーを EPM System セキュリティ・モードに移行します。
EssSetGroupToSS	すべてのグループを EPM System セキュリティ・モードに移行します。
EssGetEssbaseSecurityMode	使用されるセキュリティのタイプを表示します。
EssListSSMigrFailedUsers	Shared Services へ正常に移行しなかったユーザーを表示します。
EssListSSMigrFailedGroups	Shared Services へ正常に移行しなかったグループを表示します。
EssReRegisterApplication	1 つまたはすべての Essbase アプリケーションを Shared Services アプリケーションとして再確立します。
EssSetEasLocation	Essbase 管理サーバーの場所を設定または変更します。これは、アプリケーションの作成または移行のとき Shared Services に登録されます。

C のメイン API の Unicode モードの関数

Essbase サーバーでは Unicode モードにあるときのみ、Unicode モード・アプリケーションの作成、または非 Unicode モード・アプリケーションの Unicode モードへの移行を実行できます。

次の関数は、Unicode モードで Essbase サーバーおよびアプリケーションを操作するときに役立ちます。

関数	説明
EssSetServerMode	Essbase サーバーのモードを Unicode または非 Unicode に設定します。
EssGetServerMode	Essbase サーバーが Unicode モードと非 Unicode モードのどちらであるかを示します。
EssCreateApplicationEx	Unicode モード・アプリケーションを作成します。
EssConvertApplicationtoUnicode	非 Unicode モードのアプリケーションを Unicode モードのアプリケーションに変換します。
EssGetApplicationInfo および EssGetApplicationInfoEx	ロケール情報を含むアプリケーション情報を戻します。
EssUpdateFileUTF8ASO	UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。
EssUpdateFileUTF8ASOEx	すべてのデータ・ロード・エラーを取得して、UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに更新指定を送信します。
EssUpdateFileUtf8Ex	すべてのデータ・ロード・エラーを取得して、UTF-8 でエンコードされたファイルからアクティブなデータベースに更新指定を送信します

C のメイン API 関数のリファレンス

「コンテンツ」ペインで C のメイン API 関数のアルファベット順のリストを参照してください。

EssAddToGroup

グループ・メンバーのリストにユーザーを追加します。

構文

```
ESS_FUNC_M EssAddToGroup (
hCtx, GroupName, UserName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	グループ名。
UserName	ESS_STR_T	グループ・リストへ追加するユーザー名。

備考

- この関数では、グループ・メンバー・リストにユーザーが追加されるとともに、ユーザー自身のグループ・リストにグループが追加されます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
ESS_FUNC_M
ESS_AddUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   GroupName;
 ESS_STR_T   UserName;
 GroupName = "PowerUsers";
 UserName = "Jim Smith";

 sts = EssAddToGroup (hCtx, GroupName, UserName);

 return (sts);
}
```

関連トピック

- EssAddToGroupEx
- EssDeleteFromGroup
- EssGetGroupList
- EssListGroups
- EssSetGroupList

EssAddToGroupEx

指定したグループにユーザーを追加します。EssAddToGroup に似ていますが、ユーザー・ディレクトリの指定、または GroupId や UserId の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssAddToGroupEx (
hCtx
,
GroupId
,
UserId
,
bUsingIdentity
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupId	ESS_STR_T	グループ名または ID (入力)。groupname@provider または一意の ID 属性として指定できます。
bIsGroupId	ESS_BOOL_T	入力。GroupId が名前か ID かを示します。TRUE の場合、GroupId は ID です。
UserId	ESS_STR_T	グループに追加するユーザー名(入力)。username@provider または一意の ID 属性として指定できます。
bUsingIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。

備考

- API は ID またはグループ名を受け入れることができます。グループ名は groupname@provider として指定できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
void DisplayUserList(ESS_USHORT_T count, ESS_PSTR_T UserList)
{
    ESS_USHORT_T i;

    for (i = 0; i < count; i++)
    {
        if (UserList [i])
            printf ("%s\n", UserList[i]);
    }
}

ESS_FUNC_M ESS_AddUser (ESS_HCTX_T  hCtx)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_STR_T groupId, userId;
 ESS_BOOL_T bGroupId, bUserId;
 ESS_BOOL_T bisIdentity;
 ESS_USHORT_T type;
 ESS_USHORT_T count;
 ESS_BOOL_T bUsingIdentity;
 ESS_PSTR_T pUserList;

 groupId = "IDRegularGroup@ldap";
 bGroupId = ESS_FALSE;
 userId = "IDUser6";
 bUserId = ESS_FALSE;
 sts = EssAddToGroupEx(hCtx, groupId, bGroupId, userId, bUserId);
 printf("EssAddToGroupEx sts: %ld\n", sts);
 if(!sts)
 {
  sts = EssGetGroupListEx(hCtx, groupId, bisIdentity, type, &count, &bUsingIdentity,
&pUserList);
   printf("EssGetGroupListEx sts: %ld\n", sts);
   if(!sts)
   {
    if(pUserList)
    {
     printf ("\n---User/Group list for %s:\n", groupId);
     DisplayUserList(count, pUserList);
    }
    else
     printf ("\tUser list is empty\n");
   }
 }

 return (sts);
}
```

関連トピック

- EssDeleteFromGroupEx
- EssGetGroupListEx

- EssListGroupsInfoEx

EssAlloc

定義されたメモリー割当ての仕組みを使用して、メモリー・ブロックを割り当てます。

構文

```
ESS_FUNC_M EssAlloc (
hInstance, Size, ppBlock
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。
Size	ESS_SIZE_T	割り当てるメモリー・ブロックのサイズ。
ppBlock	ESS_PPVOID_T	割り当てられたメモリー・ブロックを受け取るポインタのアドレス。

備考

- この関数は、EssInit 関数に渡されたユーザー指定のメモリー管理関数を使用してメモリーを割り当てます。このような関数が指定されていない場合、デフォルトのメモリー割当て関数(プラットフォーム依存)が使用されます。
- この関数を使用して割り当てられたメモリーの再割当てと解放は、それぞれ EssRealloc 関数と EssFree 関数を使用して行う必要があります。
- 一般的に、サイズが 0 のブロックの割当ての結果はプラットフォームやコンパイラに依存するため、このような割当てはお薦めしません。

戻り値

ppBlock に割り当てられたメモリー・ブロックへのポインタが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M ESS_GetAppActive (ESS_HCTX_T hCtx,
     ESS_HINST_T hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   pDbName;
 ESS_STR_T   pAppName;
 ESS_ACCESS_T  Access;

 if ((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pAppName)) == 0)
 {
  if ((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pDbName)) == 0)
  {
   if ((sts = EssGetActive (hCtx, &pAppName, &pDbName, &Access)) == 0)
```

```
    {
     if (pAppName)
     {
      if (*pAppName)
       printf ("Current active application is [%s]\r\n",pAppName);
      else
       printf ("No active Application is set\r\n");
      printf ("\r\n");
     }
    }
    EssFree (hInst, pDbName);
   }
   EssFree (hInst, pAppName);
  }
  return (sts);
}
```

関連トピック

- EssFree
- EssInit
- EssRealloc

EssArchive

使用されなくなりました。

この関数は、Essbase の以前のバージョンとの互換性のためにのみ保持されています。現行の Essbase アーカイブについては、EssArchiveBegin および EssArchiveEnd を参照してください。この関数は、エラー・メッセージ ESS_STS_OBSOLETE を戻します。

関連トピック

- EssRestore
- EssArchiveBegin
- EssArchiveEnd

EssArchiveBegin

サーバー・モードを読取り専用に変更して、サーバーでアーカイブの準備をします。

構文

```
ESS_FUNC_M EssArchiveBegin (
hCtx, AppName, DbName, FileName
);
```

パラメータ　データ型　　説明

hCtx　　ESS_HCTX_T API コンテキスト・ハンドル

パラメータ	データ型	説明
AppName	ESS_STR_T	アーカイブするアプリケーション名
DbName	ESS_STR_T	アーカイブするデータベース名
FileName	ESS_STR_T	アーカイブ情報を含むファイルの名前

備考

- この関数はサーバー・モードを読取り専用に変更します。このモードによって、データベース管理者はサーバー上のすべてのファイルをバックアップでき、バックアップ中にファイルに書き込まれないようにします。バックアップするデータベース・ファイルは、FileName パラメータで指定される app\db ディレクトリにリストされます。
- 指定したファイル内の既存の情報はすべて、アーカイブされたデータによって上書きされます。

戻り値

なし。

アクセス

呼出し元は、少なくともデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持ち、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択する必要があります。

例

```
    ESS_FUNC_M
ESS_ArchiveBegin(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;
 AppName  = "Sample";
 DbName   = "Basic";
 FileName = "Test.arc";

 /* Begin Archive */
 sts = EssArchiveBegin(hCtx, AppName, DbName,
    FileName);
 return (sts);
}
```

関連トピック

- EssArchiveEnd
- EssRestore

EssArchiveDatabase

指定されたバックアップ・ファイルにデータベースのアーカイブを作成します。

構文

```
     ESS_FUNC_M EssArchiveDatabase (hCtx, AppName, DbName, BackupFileName,
OptionsFileName, bOverWrite);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	ログイン・コンテキスト
AppName	ESS_STR_T	アプリケーション名 **注：** データベース・レベルでのみ動作します。AppName パラメータはアプリケーションを指定し、存在するデータベースにアクセスできるようにします。
DbName	ESS_STR_T	データベース名
BackupFileName	ESS_STR_T	データをアーカイブするバックアップ・ファイルへのフルパス。次の例のようにフル・パスを指定します: `c:\hyperion\Test.arc`
OptionsFileName	ESS_FILENAME_T	今後の使用のために予約済。 **注：** このリリースでは空の文字列を使用します。
bOverWrite	ESS_BOOL_T	ブール。値: • ESS_TRUE – 既存のバックアップ・ファイルを上書きします。 • ESS_FALSE – 上書きしません。既存のバックアップ・ファイルに追加します。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

アクセス

呼出し元にはデータベースへの Essbase 管理者アクセス権限が必要です。

例

```
    void RestoreDB()
{
    ESS_FUNC_M        sts = ESS_STS_NOERR;
    ESS_STR_T            AppName = "Backup";
    ESS_STR_T            DbName = "Basic";
    ESS_STR_T            BackupFileName =
                  "F:\\testArea\\ArchiveAndRestore\\TempBackup.arc";
```

```
    ESS_STR_T            optionsFileName = "";
    ESS_BOOL_T        bOverWrite;
    ESS_BOOL_T        bForceDiffName;
    ESS_USHORT_T        count;
    ESS_PDISKVOLUME_REPLACE_T    replaceVol;

    printf("\nArchive DB:\n");
    bOverWrite = ESS_TRUE;
    sts =
    EssArchiveDatabase(hCtx, AppName, DbName,
                                          BackupFileName, optionsFileName,
                                          bOverWrite);


    printf("EssArchiveDatabase sts: %ld\r\n",sts);

    sts = EssUnloadApplication(hCtx, AppName);
    printf("\nEssUnloadApplication sts: %ld\r\n",sts);

    printf("\nCase with no volume replacement:\n");
    bForceDiffName = ESS_FALSE;
    count = 0;
    replaceVol = ESS_NULL;
    sts = EssRestoreDatabase (hCtx, AppName, DbName,
                                             BackupFileName, bForceDiffName,
                                             count, replaceVol);
    printf("EssRestoreDatabase sts: %ld\r\n",sts);

    printf("\nCase with a replacement volume (index and page files to a different
volume):\n");
    bForceDiffName = ESS_FALSE;
    count = 1;
    if (count)
    {
        sts = EssAlloc(hInst, count * sizeof(ESS_DISKVOLUME_REPLACE_T),
                                    (ESS_PPVOID_T)&replaceVol);
        memset(replaceVol, 0, count * sizeof(ESS_DISKVOLUME_REPLACE_T));
    }
    strcpy(replaceVol->szPartition_Src, "C");
    strcpy(replaceVol->szPartition_Dest, "F");

    sts = EssUnloadApplication(hCtx, AppName);
    printf("\nEssUnloadApplication sts: %ld\r\n",sts);

    sts = EssRestoreDatabase (hCtx, AppName, DbName,
                                             BackupFileName, bForceDiffName,
                                             count, replaceVol);
    printf("EssRestoreDatabase sts: %ld\r\n",sts);

    if (replaceVol)
         EssFree(hInst, replaceVol);
}
```

関連トピック

- EssRestoreDatabase

EssArchiveEnd

アーカイブが完了した後、サーバーが読取り書込みモードに戻されます。

構文

```
ESS_FUNC_M EssArchiveEnd (
hCtx, AppName, DbName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アーカイブされたアプリケーション名。
DbName	ESS_STR_T	アーカイブされたデータベースの名前。

備考

- EssArchiveBegin を呼び出した後、EssArchiveEnd を呼び出して読取り書込みモードへ戻す必要があります。

戻り値

なし。

アクセス

呼出し元は、少なくともデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持ち、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択する必要があります。

例

```
    ESS_FUNC_M
ESS_ArchiveEnd(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 AppName  = "Sample";
 DbName   = "Basic";

 /* End Archive */
 sts = EssArchiveEnd(hCtx, AppName, DbName);
 return (sts);
}
```

関連トピック

- EssArchiveBegin
- EssRestore

EssAsyncBuildDim

非同期次元構築要求を発行します。

非同期データ・ロードおよび次元構築を使用する場合、このプロセス中に次の情報をクエリーできます:

- 次元構築/データ・ロード・プロセスの状態: 進行中、最終段階または完了済
- 次元構築/データ・ロード・プロセスの段階: データ・ソースを開いている最中、アウトラインの読取り中、次元の構築中、アウトラインの確認中またはアウトラインの書込み中
- 現在までに処理されたデータ・レコード数と拒否されたデータ・レコード数
- エラー・ファイルの名前および場所
- 現在までに処理されたデータ・レコードと拒否されたデータ・レコード

構文

```
ESS_FUNC_M EssAsyncBuildDim(
hCtx, RulesObj, DataObj, MbrUser, bOverwrite, usBuildOption, szTmpOtlFile
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RulesObj	ESS_POBJDEF_T	ルール・ファイル・オブジェクト定義構造体へのポインタ。
DataObj	ESS_POBJDEF_T	データ・ファイル・オブジェクト定義構造体へのポインタ。
MbrUser	ESS_PMBRUSER_T	SQL ユーザー構造体(データ・ソースが SQL データベースの場合)。 SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
bOverwrite	ESS_BOOL_T	ブール。値: ● ESS_TRUE - 既存のエラー・ファイルを上書きします。 ● ESS_FALSE - 上書きしません。既存のエラー・ファイルに追加します。
usBuildOption	ESS_USHORT_T	有効な値: ● ESS_INCDIMBUILD_BUILD メンバーの構築のみ行います。 ● ESS_INCDIMBUILD_VERIFY メンバーを構築してアウトラインを確認します。 ● ESS_INCDIMBUILD_SAVEOTL メンバーを構築して、アウトラインを一時アウトライン・ファイルに保存します。 ● ESS_INCDIMBUILD_ALL メンバーを構築し、アウトラインを確認し、再構築します。 ● ESS_INCDIMBUILD_ABORT 構築プロセスを中止します。

パラメータ	データ型	説明
szTmpOtlFile	ESS_STR_T	一時アウトライン・ファイルの名前。拡張子またはパスは必要ありません。この回の次元構築で得られたアウトラインにアウトライン確認エラーがある場合、Essbase は拡張子が .otb の一時アウトライン・ファイルを app/db ディレクトリに作成します。

備考

この関数は、データ・オブジェクトがクライアント上にある場合はエラーを戻します。エラーが戻された場合でも、クライアントとサーバー間のネットワーク接続はアクティブなままです。

接続を閉じるには、EssCloseAsyncProc を呼び出す必要があります。接続を閉じていない場合、サーバー要求ハンドラにより、同じログイン・セッションからの後続の要求がブロックされます。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
void ESS_AsyncBuildDim()
{
    ESS_STS_T sts = 0;
    ESS_OBJDEF_T Rules;
    ESS_OBJDEF_T Data;
    ESS_PMBRUSER_T pMbrUser;
    ESS_BOOL_T bOverwrite;
    ESS_USHORT_T usBuildOption;
    ESS_STR_T szTmpOtlFile;
    ESS_STR_T bldDimErrFile;
    ESS_STR_T asyncProcErrLog;
    ESS_BLDDL_STATE_T procState;
    ESS_BOOL_T errFileOverWrite;

    szAppName = "Sample";
    szDbName = "Basic";
    ESS_SetActive();

    AddMember("800");

    sts = EssBeginIncrementalBuildDim(hCtx);
    printf("EssBeginIncrementalBuildDim sts: %ld\n",sts);

    memset(&Rules,0,sizeof(ESS_OBJDEF_T));
    memset(&Data,0,sizeof(ESS_OBJDEF_T));
    Rules.hCtx   = hCtx;
    Rules.FileName = "apgeibl";
    Rules.AppName  = szAppName;
    Rules.DbName   = szDbName;
    Rules.ObjType  = ESS_OBJTYPE_RULES;
    Data.hCtx    = hCtx;
    Data.AppName   = szAppName;
    Data.DbName    = szDbName;
```

```
    Data.ObjType  = ESS_OBJTYPE_TEXT;
    Data.FileName  = "apgeibl1";

    pMbrUser = ESS_NULL;
    bOverwrite = ESS_TRUE;
    usBuildOption = ESS_INCDIMBUILD_BUILD;
    szTmpOtlFile = "asyncBldTmp";
    sts = EssAsyncBuildDim(hCtx, &Rules, &Data, pMbrUser, bOverwrite, usBuildOption,
szTmpOtlFile);
    printf("EssAsyncBuildDim sts: %ld\n",sts);

    sts = EssGetAsyncProcLog (hCtx, ".\\AsyncProc.log", ESS_TRUE);
    printf("EssGetAsyncProcLog sts: %ld\n",sts);

    sts = EssGetAsyncProcState(hCtx, &procState);
    printf("EssGetAsyncProcState sts: %ld\n",sts);
    if(!sts)
    {
        do
        {
            DisplyProcesStateInfo(procState);
            if(procState.ilProcessStatus)
            {
                sts = EssCancelAsyncProc(hCtx, asyncProcErrLog, errFileOverWrite);
                printf("EssCancelAsyncProc sts: %ld\n",sts);
            }
            else
            {
                sts = EssGetAsyncProcState(hCtx, &procState);
                printf("EssGetAsyncProcState sts: %ld\n",sts);
            }

        }while(procState.usProcessState != ESS_BLDDL_STATE_DONE);

        if(!procState.ilProcessStatus)
        {
            sts = EssCloseAsyncProc(hCtx, &procState);
            printf("EssCloseAsyncProc sts: %ld\n",sts);
        }
    }

    bldDimErrFile = "F:\\testArea\\mainapi\\BldDim.err";
    sts = EssEndIncrementalBuildDim(hCtx, ESS_DOR_ALLDATA, szTmpOtlFile,
bldDimErrFile, ESS_FALSE);
    printf("EssEndIncrementalBuildDim sts: %ld\n",sts);
}
```

関連トピック

- EssAsyncImport
- EssGetAsyncProcLog
- EssGetAsyncProcState
- EssCancelAsyncProc
- EssCloseAsyncProc

EssAsyncImport

非同期データ・ロード要求を発行します。

非同期データ・ロードおよび次元構築を使用する場合、このプロセス中に次の情報をクエリーできます:

- 次元構築/データ・ロード・プロセスの状態: 進行中、最終段階または完了済
- 次元構築/データ・ロード・プロセスの段階: データ・ソースを開いている最中、アウトラインの読取り中、次元の構築中、アウトラインの確認中またはアウトラインの書込み中
- 現在までに処理されたデータ・レコード数と拒否されたデータ・レコード数
- エラー・ファイルの名前および場所
- 現在までに処理されたデータ・レコードと拒否されたデータ・レコード

構文

```
ESS_FUNC_M EssAsyncImport (
hCtx, pRules, pData, pMbrUser, abortOnError
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRules	ESS_POBJDEF_T	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pData	ESS_POBJDEF_T	データ・ファイル・オブジェクト定義構造体へのポインタ。
pMbrUser	ESS_PMBRUSER_T	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
abortOnError	ESS_USHORT_T	TRUE の場合、最初のエラーでインポートが停止します。それ以外の場合は続行します。

備考

この関数は、データ・オブジェクトがクライアント上にある場合はエラーを戻します。エラーが戻された場合でも、クライアントとサーバー間のネットワーク接続はアクティブなままです。

接続を閉じるには、EssCloseAsyncProc を呼び出す必要があります。接続を閉じていない場合、サーバー要求ハンドラにより、同じログイン・セッションからの後続の要求がブロックされます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
ESS_AsyncImport()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_SHORT_T isAbortOnError;
    ESS_OBJDEF_T Rules;
    ESS_OBJDEF_T Data;
    ESS_PMBRUSER_T pUser;
    ESS_STR_T errorName;
    ESS_BLDDL_STATE_T procState;
    ESS_BOOL_T errFileOverWrite;

    szAppName = "Sample";
    szDbName = "Basic";
    ESS_SetActive();

    memset(&Rules,0,sizeof(ESS_OBJDEF_T));
    memset(&Data,0,sizeof(ESS_OBJDEF_T));
    Rules.hCtx   = hCtx;
    Rules.FileName = "Act1";
    Rules.AppName  = szAppName;
    Rules.DbName  = szDbName;
    Rules.ObjType  = ESS_OBJTYPE_RULES;
    Data.hCtx    = hCtx;
    Data.FileName = "Act1";
    Data.AppName  = szAppName;
    Data.DbName   = szDbName;
    Data.ObjType  = ESS_OBJTYPE_TEXT;

    errorName = ".\\asyncProcess.err";
    errFileOverWrite = ESS_TRUE;
    isAbortOnError = ESS_TRUE;
    pUser = ESS_NULL;  /* NULL equals a non-SQL data source */
    sts = EssAsyncImport (hCtx, &Rules, &Data, pUser, isAbortOnError);
    printf("EssAsyncImport sts: %ld\n",sts);

    sts = EssGetAsyncProcState(hCtx, &procState);
    printf("EssGetAsyncProcState sts: %ld\n",sts);
    if(!sts)
    {
        do
        {
            DisplyProcesStateInfo(procState);
            if(procState.ilProcessStatus)
            {
                sts = EssCancelAsyncProc(hCtx, errorName, errFileOverWrite);
                printf("EssCancelAsyncProc sts: %ld\n",sts);
            }
            else
            {
                sts = EssGetAsyncProcState(hCtx, &procState);
                printf("EssGetAsyncProcState sts: %ld\n",sts);
            }
```

```
        }while(procState.usProcessState != ESS_BLDDL_STATE_DONE);

        if(!procState.ilProcessStatus)
        {
            sts = EssCloseAsyncProc(hCtx, &procState);
            printf("EssCloseAsyncProc sts: %ld\n",sts);
        }
    }
}
```

関連トピック

- EssAsyncBuildDim
- EssGetAsyncProcLog
- EssGetAsyncProcState
- EssCancelAsyncProc
- EssCloseAsyncProc

EssAsyncImportASO

集約ストレージ・データベースに対する非同期データ・ロード要求を発行します。

非同期データ・ロードおよび次元構築を使用する場合、このプロセス中に次の情報をクエリーできます:

- 次元構築/データ・ロード・プロセスの状態: 進行中、最終段階または完了済
- 次元構築/データ・ロード・プロセスの段階: データ・ソースを開いている最中、アウトラインの読取り中、次元の構築中、アウトラインの確認中またはアウトラインの書込み中
- 現在までに処理されたデータ・レコード数と拒否されたデータ・レコード数
- エラー・ファイルの名前および場所
- 現在までに処理されたデータ・レコードと拒否されたデータ・レコード

構文

```
ESS_FUNC_M EssAsyncImportASO (
hCtx, pRules, pData, pUser, usAbortOnError, ulBufferId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRules	ESS_POBJDEF_T	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pData	ESS_POBJDEF_T	データ・ファイル・オブジェクト定義構造体へのポインタ。
pUser	ESS_PMBRUSER_T	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。

パラメータ	データ型	説明
usAbortOnError	ESS_USHORT_T	TRUE の場合、最初のエラーでインポートが停止します。それ以外の場合は続行します。
ulBufferID	ESS_ULONG_T	データ・ロード・バッファの ID（1 から 999,999 までの数）。データのロードが完了する前にバッファを廃棄するには、バッファを初期化する場合に使用したものと同じ ulBufferId 番号を使用する必要があります。

備考

この関数は、データ・オブジェクトがクライアント上にある場合はエラーを戻します。エラーが戻された場合でも、クライアントとサーバー間のネットワーク接続はアクティブなままです。

接続を閉じるには、EssCloseAsyncProc を呼び出す必要があります。接続を閉じていない場合、サーバー要求ハンドラにより、同じログイン・セッションからの後続の要求がブロックされます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void ESS_AsyncImportASO()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_SHORT_T isAbortOnError;
    ESS_OBJDEF_T Rules;
    ESS_OBJDEF_T Data;
    ESS_PMBRERR_T pMbrErr = NULL;
    ESS_PMBRUSER_T pMbrUser = NULL;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T ulDuplicateAggregationMethod;
    ESS_ULONG_T ulOptionsFlags;
    ESS_ULONG_T ulSize;
    ESS_ULONG_T ulBufferCnt;
    ESS_ULONG_T ulCommitType ;
    ESS_ULONG_T ulActionType;
    ESS_ULONG_T ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];
    ESS_STR_T errorName;
    ESS_BLDDL_STATE_T procState;
    ESS_BOOL_T errFileOverWrite;

    szAppName = "ASOSamp";
    szDbName = "Sample";
    ESS_SetActive();
```

```
    errorName = ".\\asyncProcess.err";
    errFileOverWrite = ESS_TRUE;
    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 1;
    ulBufferId = 100;
    sts = EssLoadBufferInit(hCtx, szAppName, szDbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);
    if(!sts)
    {
        /* Server object */
        Rules.hCtx   = hCtx;
        Rules.AppName  = szAppName;
        Rules.DbName  = szDbName;
        Rules.ObjType  = ESS_OBJTYPE_RULES;
        Rules.FileName = "Dataload";
        Data.hCtx    = hCtx;
        Data.AppName  = szAppName;
        Data.DbName   = szDbName;
        Data.ObjType  = ESS_OBJTYPE_TEXT;
        Data.FileName  = "Dataload";

        isAbortOnError = ESS_TRUE;
        sts = EssAsyncImportASO (hCtx, &Rules, &Data, pMbrUser, isAbortOnError,
ulBufferId);
        printf("EssAsyncImportASO sts: %ld\n",sts);
        if(!sts)
        {
            sts = EssGetAsyncProcState(hCtx, &procState);
            printf("EssGetAsyncProcState sts: %ld\n",sts);
            if(!sts)
            {
                do
                {
                    DisplyProcesStateInfo(procState);
                    if(procState.ilProcessStatus)
                    {
                        sts = EssCancelAsyncProc(hCtx, errorName, errFileOverWrite);
                        printf("EssCancelAsyncProc sts: %ld\n",sts);
                    }
                    else
                    {
                        sts = EssGetAsyncProcState(hCtx, &procState);
                        printf("EssGetAsyncProcState sts: %ld\n",sts);
                    }

                }while(procState.usProcessState != ESS_BLDDL_STATE_DONE);

                sts = EssCloseAsyncProc(hCtx, &procState);
                printf("EssCloseAsyncProc sts: %ld\n",sts);

                ulBufferCnt = 1;
                ulBufferIdAry[0] = ulBufferId;
                ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
                ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
```

```
                printf("\nIncrement to main slice:\n");
                ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
                sts = EssLoadBufferTerm(hCtx, szAppName, szDbName, ulBufferCnt,
ulBufferIdAry, ulCommitType, ulActionType, ulOptions);
                printf("EssLoadBufferTerm sts: %ld\n",sts);
            }
        }
    }
}
```

関連トピック

- EssAsyncBuildDim
- EssAsyncImport
- EssGetAsyncProcLog
- EssGetAsyncProcState
- EssCancelAsyncProc
- EssCloseAsyncProc

EssAutoLogin

ユーザーが Essbase サーバーにログインするためのダイアログ・ボックスを表示します。オプションでアクティブなアプリケーションとデータベースを選択できます。

構文

```
     ESS_FUNC_M EssAutoLogin
     (hInstance, Server, UserName,
Password, AppName, DbName, Options, pAccess, phCtx)
     ;
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル

パラメータ	データ型	説明
Server	ESS_SVRNAME_T	ネットワーク・サーバー名文字列 サーバー名は、APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します: http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1 保護モード(SSL)の場合、URL の構文は次のとおりです http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno たとえば、 https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes
UserName	ESS_USERNAME_T	ユーザー名文字列
Password	ESS_PASSWORD_T	パスワード文字列
AppName	ESS_ APPNAME _T	アプリケーション名
DbName	ESS_ DBNAME _T	データベース名
Options	ESS_USHORT_T	オプションのフラグ。値: ● AUTO_NODIALOG - (前述の引数の)デフォルト設定を使用して、ダイアログを表示しないでユーザーのログインを試みます。 ● AUTO_NOSELECT - ユーザーは、アプリケーションおよびデータベースを選択しないでもログインできます(ダイアログの下側の部分が表示されません)。 OR 演算子(|)を使用して AUTO_NODIALOG と AUTO_NOSELECT の両方を設定し、ダイアログ・ボックスの表示およびアプリケーションとデータベースの選択を省略して、ユーザー・ログインを実行します。 ● AUTO_NODIALOG|AUTO_NOSELECT: AUTO_DEFAULT - ダイアログ・ボックスが表示され、ユーザーは対話形式でログインし、アプリケーションとデータベースを選択できます。
pAccess	ESS_PACCESS_T	データベース・アクセス・レベルを受け取る変数のアドレス。
phCtx	ESS_PHCTX_T	Essbase コンテキスト・ハンドルを受け取る変数のアドレス。再ログインする既存の(有効な)コンテキスト・ハンドルを再使用する場合を除いて、ESS_INVALID_HCTX に設定します。

備考

- ダイアログ・ボックスは、関数により自動的に管理されます。ログイン・ダイアログでは、ユーザー・パスワードの変更や、データベース・ノート・メッセージの表示ができます。また、API を使用するすべてのアプリケーションで、標準化された強力なログイン画面が表示されます。

- Windows 環境でプログラミングをする場合は、EssLogin 関数のかわりにこの関数を使用してください。
- この関数は、EssInit が正しく実行された後で、かつコンテキスト・ハンドル引数を必要とするその他すべての API が呼び出される前に呼び出す必要があります。
- この関数は Windows 環境でのみサポートされます。UNIX 環境ではサポートされません。
- 文字列引数 Server、UserName、Password、AppName または DbName は、オプションで NULL でもかまいません。いずれかが NULL でない場合、それらがポイントするバッファは、ダイアログ・ボックスからユーザーによって選択された実績値を関数が戻すときに更新されます。渡された引数のいくつかが有効な文字列をポイントする場合、それらは、デフォルトでダイアログに表示される値として使用されます。これらの引数のバッファは、渡された値のみでなく、戻り値も含むことができる大きさである必要があります。
- ログインに成功すると、サーバー名とユーザー名が(ファイル ESSBASE.INI に)自動的に保管され、次にこの関数が呼び出される際にデフォルトとして使用されます(この引数が後続の呼出しに指定されていない場合)。正常に接続したすべてのサーバー名も保管され、表示されます。
- 「自動ログイン」ダイアログ・ボックスは、現在のアクティブなウィンドウ(フォーカスがあるウィンドウ)の子ウィンドウです。したがって、「自動ログイン」ダイアログが表示されている間は、アクティブなウィンドウを破棄したり、フォーカスを変更しないでください。
- ダイアログ・ボックスで「取消し」ボタンをクリックするか、[Esc]キーを押すと、この関数から値 ESS_STS_CANCEL が戻されます。
- Windows 環境では、エンド・ユーザーが「ヘルプ」ボタンをクリックすると、Essbase システムのログインに関するヘルプ・トピックが開きます。ESS_INIT_T 構造体の異なるヘルプ・ファイル名を指定すると、異なるヘルプ・ファイルをポイントするように「ヘルプ」ボタンを変更できます。

戻り値

成功の場合、phCtx の Essbase コンテキスト・ハンドルを戻します。それを、他の API 関数への後の呼出しで、引数として渡せます。また、pAccess で選択されたアプリケーションおよびデータベース(選択されている場合)へのユーザーのアクセス・レベルを戻します。

アクセス

この関数を呼び出す前に、EssInit 関数を呼び出し、API を初期化して、有効なインスタンス・ハンドルを取得してください。

関連トピック

- EssInit
- EssListDatabases
- EssLogin
- EssLogout
- EssSetActive

EssBeginCalc

計算スクリプトの送信を開始し、オプションでアクティブ・データベースに対して計算スクリプトを実行します。

構文

```
ESS_FUNC_M EssBeginCalc (
hCtx, Calculate
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Calculate	ESS_BOOL_T	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行されます。

備考

- この関数を呼び出した後は、EssSendString を呼び出して計算スクリプトを送信し、最後に EssEndCalc を呼び出す必要があります。
- 計算スクリプトの長さは、合計で 64KB 未満である必要があります。
- 計算を開始することも、または計算スクリプトの確認のみを行いエラーを戻すこともできます。
- 計算スクリプトの送信に成功し、計算が開始すると、この呼出しから戻された後も、サーバー上で非同期プロセスとして続行されます。EssEndCalc の呼出し後、呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- Calculate フラグが FALSE に設定されている場合、データベースは計算スクリプトの構文チェックのみを行います。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Calc  (ESS_HCTX_T     hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
```

```
 ESS_STR_T     Script;
 ESS_PROCSTATE_T  pState;
 Script = "CALC ALL;";

 sts = EssBeginCalc (hCtx,ESS_TRUE);
 if (!sts)
  sts = EssSendString (hCtx, Script);
 if (!sts)
  sts = EssEndCalc (hCtx);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while(!sts && (pState.State !=
     ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 return(sts);
}
```

関連トピック

- EssCalc
- EssCalcFile
- EssDefaultCalc
- EssEndCalc
- EssGetDefaultCalc
- EssGetProcessState
- EssSendString
- EssSetDefaultCalc

EssBeginDataload

アクティブ・データベースに更新指定の送信を開始し、更新用にロックされたデータ・ブロックのロックを解除できます。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。

構文

```
ESS_STS_T EssBeginDataload (
hCtx, Store, Unlock, abortOnError, pRules
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
Store;	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock;	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。

パラメータ	データ型	説明
abortOnError;	ESS_BOOL_T	TRUE の場合、最初のエラーでデータ・ロードが停止します。それ以外の場合は、データ・ロードを続行します。
pRules;	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。

備考

- この関数の後に、EssSendString を少なくとも 1 回呼び出して更新指定を送信し、次に EssEndDataload を呼び出す必要があります。
- EssBeginDataload の後に呼び出した EssSendString へ渡される各文字列の末尾は、改行復帰文字シーケンス("\r\n")である必要があります。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。
- 誤った入力行以降の入力行(レコード)を無視する EssBeginUpdate とは異なり、この関数は残りの入力行も処理し、必要に応じてコミットします。
- EssEndDataload は、156 ページの「ESS_MBRERR_T」にエラーのリンク・リストを戻します。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

なし。

アクセス

EssBeginDataload を使用するには、呼出し元がアクティブなデータベースに対する書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_STS_T   sts = ESS_STS_NOERR;
ESS_BOOL_T   Store;
ESS_BOOL_T   Unlock;
ESS_STR_T   Query1, Query2;
ESS_PMBRERR_T pMbrErr;

 Store  = ESS_TRUE;
 Unlock = ESS_FALSE;
 Query1 = "Year Market Scenario Measures Product 12345";
 Query2 = " Jan  East  Scenario Measures Coke  125";

 /* Begin Update */
 sts = EssBeginDataload (hCtx, Store, Unlock, ESS_FALSE, ESS_NULL);

 /* Send update specification */
 if(!sts)
```

```
  sts = EssSendString(hCtx, Query1);
  sts = EssSendString(hCtx, Query2);

/* End Update */
if(!sts)
  sts = EssEndDataload(hCtx, &pMbrErr);
```

関連トピック

- EssSendString
- EssEndDataload
- EssBeginUpdate
- EssEndUpdate
- EssUpdate
- EssImport

EssBeginDataloadASO

集約ストレージ・データベース上でデータ・ロードを開始します。

構文

```
ESS_FUNC_M EssBeginDataloadASO (
hCtx, Store, Unlock, abortOnError, pRules, ulBufferId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Store	ESS_BOOL_T	データの保管を制御します。ESS_TRUE の場合は、データがサーバーに保管され、ESS_FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	集約ストレージ・データベースではサポートされていません。このパラメータには必ず ESS_FALSE を渡す必要があります。
abortOnError	ESS_BOOL_T	ESS_TRUE は、プロセス中にエラーが発生した場合、データ・ロードが中止されることを示しています。
pRules	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID。データのロードが完了する前にバッファを廃棄するには、バッファの初期化に使用したのと同じ ulBufferId 番号を使用する必要があります。

備考

- この関数の後に、EssSendString を少なくとも 1 回呼び出して更新指定を送信し、次に EssEndDataload を呼び出す必要があります。
- EssBeginDataloadASO の後に呼び出した EssSendString へ渡される各文字列の末尾は、改行復帰文字シーケンス("\r\n")である必要があります。

- Store フラグが FALSE に設定されている場合、データベースは更新指定の構文チェックのみを行います。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    void TestBeginDataloadASO(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_BOOL_T    Store;
    ESS_BOOL_T    Unlock;
    ESS_BOOL_T    abortOnError;
    ESS_STR_T    loadString;
    ESS_OBJDEF_T rulesFile;
    ESS_PMBRERR_T pMbrErr;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T   ulDuplicateAggregationMethod;
    ESS_ULONG_T   ulOptionsFlags;
    ESS_ULONG_T   ulSize;
    ESS_ULONG_T   ulBufferCnt;
    ESS_ULONG_T   ulCommitType ;
    ESS_ULONG_T   ulActionType;
    ESS_ULONG_T   ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];

    /* EssLoadBufferInit */
    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
    ulBufferId = 201;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* EssBeginDataloadASO, EssSendString, EssEndDataload */
    Store  = ESS_TRUE;
    Unlock = ESS_FALSE;
    abortOnError = ESS_FALSE;
    loadString = "Mar Sale \"Curr Year\" \"Original Price\" \"017589\" \"13668\" Cash
\"No Promotion\" \"1 to 13 Years\" \"Under 20,000\" \"Digital Cameras\" 111";

    sts = EssBeginDataloadASO (hCtx, Store, Unlock, abortOnError, ESS_NULL,
```

```
ulBufferId);
    printf("EssBeginDataloadASO sts: %ld\n",sts);
    sts = EssSendString(hCtx, loadString);
    printf("EssSendString sts: %ld\n",sts);
    sts = EssEndDataload(hCtx, &pMbrErr);
    printf("EssEndDataload sts: %ld\n",sts);

    /* EssLoadBufferTerm */
    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\Commit data to main slice and destroy buffer:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);

}
```

関連トピック

- EssLoadBufferInit
- EssSendString
- EssEndDataload
- EssLoadBufferTerm
- EssImportASO
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssListExistingLoadBuffers
- EssMergeDatabaseData

EssBeginDataloadEx

アクティブ・データベースに更新指定の送信を開始し、更新用にロックされたデータ・ブロックのロックを解除できます。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。

構文

```
ESS_STS_T EssBeginDataloadEx (
hCtx, Store, Unlock, abortOnError, pRules, fullMbrNames
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
Store	ESS_BOOL_T	データのストレージを制御します。 ● ESS_TRUE データはサーバーに保管されます。 ● ESS_FALSE データは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
abortOnError;	ESS_BOOL_T	TRUE の場合、最初のエラーでデータ・ロードが停止します。それ以外の場合は、データ・ロードを続行します。
pRules	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
fullMbrNames	ESS_BOOL_T	TRUE の場合、エラー・ログはレコード全体の完全なメンバー名を印刷します。

備考

- この関数の後に、EssSendString を少なくとも 1 回呼び出して更新指定を送信し、次に EssEndDataload を呼び出す必要があります。
- EssBeginDataloadEx の後に呼び出した EssSendString へ渡される各文字列の末尾は、改行復帰文字シーケンス("\r\n")である必要があります。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。
- 誤った入力行以降の入力行(レコード)を無視する EssBeginUpdate とは異なり、この関数は残りの入力行も処理し、必要に応じてコミットします。
- EssEndDataload は、156 ページの「ESS_MBRERR_T」にエラーのリンク・リストを戻します。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

なし。

アクセス

EssBeginDataloadEx()を使用するには、呼出し元がアクティブなデータベースに対する書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_STS_T   sts = ESS_STS_NOERR;
ESS_BOOL_T   Store;
```

```
ESS_BOOL_T  Unlock;
ESS_STR_T   Query1, Query2;
ESS_PMBRERR_T pMbrErr;

 Store  = ESS_TRUE;
 Unlock = ESS_FALSE;
 Query1 = "Year Market Scenario Measures Product 12345";
 Query2 = " Jan  East  Scenario Measures Coke  125";

 /* Begin Update */
 sts = EssBeginDataloadEx(hCtx, Store, Unlock, ESS_FALSE, ESS_NULL, ESS_TRUE);

 /* Send update specification */
 if(!sts)
  sts = EssSendString(hCtx, Query1);
  sts = EssSendString(hCtx, Query2);

 /* End Update */
 if(!sts)
  sts = EssEndDataload(hCtx, &pMbrErr);
```

関連トピック

- EssSendString
- EssEndDataload
- EssBeginUpdate
- EssEndUpdate
- EssUpdate
- EssImport

EssBeginIncrementalBuildDim

アクティブ・データベースでメンバーを構築するプロセスを開始します。Essbase サーバーは、アクティブ・データベースのアウトラインを開き、次元構築の次の手順に備えて開いたままにします。

構文

```
ESS_FUNC_M EssBeginIncrementalBuildDim(
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_IncBuildDim( ESS_HCTX_T hCtx)
{
 ESS_STS_T   sts = 0;
 ESS_OBJDEF_T  RulesObj;
 ESS_OBJDEF_T  DataObj;
 ESS_STR_T   ErrorName;
 ESS_APPNAME_T   appname;
    ESS_DBNAME_T   dbname;

 memset(&RulesObj,0,sizeof(ESS_OBJDEF_T));
 memset(&DataObj,0,sizeof(ESS_OBJDEF_T));
 strcpy(appname, "sample");
 strcpy(dbname,"basic");

 RulesObj.hCtx   = hCtx;
 RulesObj.FileName = "genref";
 RulesObj.AppName  = appname;
 RulesObj.DbName  = dbname;
 RulesObj.ObjType  = ESS_OBJTYPE_RULES;

 DataObj.hCtx    = hCtx;
 DataObj.FileName  = "genref";
 DataObj.AppName  = appname;
 DataObj.DbName   = dbname;
 DataObj.ObjType  = ESS_OBJTYPE_TEXT;

 ErrorName     = "builddim.err";

 sts = EssBeginIncrementalBuildDim(hCtx);

 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataObj,NULL,ErrorName,true,ESS_INCDIMBUILD_BU
ILD,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_VER
IFY,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_SAV
EOTL,"tmpotl");


 sts = EssBeginStreamBuildDim(hCtx, &RulesObj,ESS_INCDIMBUILD_BUILD,"tmpotl");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-20\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-30\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-40    600-40-20\n");
 sts = EssEndStreamBuildDim(hCtx,ErrorName,false);
```

```
 sts = EssEndIncrementalBuildDim(hCtx,ESS_DOR_ALLDATA,"tmpotl",ErrorName,false);
 return sts;
}
```

関連トピック

- EssIncrementalBuildDim
- EssBeginIncrementalBuildDim
- EssBeginStreamBuildDim
- EssEndIncrementalBuildDim
- EssEndStreamBuildDim

EssBeginReport

アクティブなデータベースへのレポート指定の送信を開始します。この呼出しの後、EssSendString を続けて呼び出してレポート指定を送信し、最後に EssEndReport を呼び出す必要があります。レポート・データを出力することも、確認のみ行って、エラーがあれば戻させることもできます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。

構文

```
ESS_FUNC_M EssBeginReport (
hCtx, Output, Lock
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Output	ESS_BOOL_T	データの出力を制御します。TRUE の場合は、指定したレポートに従ってサーバーから出力されます。FALSE の場合は、データは出力されません。
Lock	ESS_BOOL_T	ブロックのロックを制御します。TRUE の場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSE の場合は、ブロックのロックは行われません。

備考

- この関数に続いて、EssSendString を少なくとも 1 回呼び出し、その後、EssEndReport を呼び出す必要があります。

  .

- この関数によってデータが出力される場合(Output フラグが TRUE)、EssGetString を呼び出して戻されたデータを読み取ることができます。
- この関数によってブロックがロックされる場合(Lock フラグが TRUE)、呼出し元はロックされたブロックのロック解除を行う必要があります(たとえば、Unlock フラグを TRUE に設定して EssUpdate を呼び出します)。

- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
     ESS_HINST_T  hInst
     )
{
 ESS_FUNC_M    sts   = ESS_STS_NOERR;
 ESS_STR_T   rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");
 if (!sts)
  sts = EssEndReport (hCtx);
 /**************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

関連トピック

- EssBeginUpdate
- EssEndReport
- EssGetString
- EssReport

- EssReportFile
- EssSendString

EssBeginStreamBuildDim

次元構築プロセスを開始します。

この関数は EssEndStreamBuildDim の前に呼び出す必要があります。この関数を呼び出した後、EssSendString を呼び出してソース・レコードを Essbase サーバーに送信します。

構文

```
ESS_FUNC_M EssBeginStreamBuildDim (
hCtx, RulesObj, usBuildOption, szTmpOtlFilename
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RulesObj	ESS_POBJDEF_T	ルール・ファイル・オブジェクト定義構造体へのポインタ。
usBuildOption	ESS_USHORT_T	有効な値: • ESS_INCDIMBUILD_BUILD メンバーの構築のみ行います。 • ESS_INCDIMBUILD_VERIFY メンバーを構築してアウトラインを確認します。 • ESS_INCDIMBUILD_SAVEOTL メンバーを構築して、アウトラインを一時アウトライン・ファイルに保存します。 • ESS_INCDIMBUILD_ALL メンバーを構築し、アウトラインを確認し、再構築します。
szTmpOtlFilename	ESS_STR_T	一時アウトライン・ファイル名。この回の次元構築で得られたアウトラインにアウトライン確認エラーがある場合、Essbase は拡張子が otb の一時アウトライン・ファイルを作成します。

備考

Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_IncBuildDim( ESS_HCTX_T hCtx)
{
    ESS_STS_T   sts = 0;
    ESS_OBJDEF_T  RulesObj;
    ESS_OBJDEF_T  DataObj;
    ESS_STR_T   ErrorName;
    ESS_APPNAME_T appname;
    ESS_DBNAME_T  dbname;

 memset(&RulesObj,0,sizeof(ESS_OBJDEF_T));
 memset(&DataObj,0,sizeof(ESS_OBJDEF_T));
 strcpy(appname, "sample");
 strcpy(dbname,"basic");

 RulesObj.hCtx   = hCtx;
 RulesObj.FileName = "genref";
 RulesObj.AppName  = appname;
 RulesObj.DbName   = dbname;
 RulesObj.ObjType  = ESS_OBJTYPE_RULES;

 DataObj.hCtx    = hCtx;
 DataObj.FileName  = "genref";
 DataObj.AppName   = appname;
 DataObj.DbName    = dbname;
 DataObj.ObjType   = ESS_OBJTYPE_TEXT;

 ErrorName     = "builddim.err";

 sts = EssBeginIncrementalBuildDim(hCtx);

 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataObj,NULL,ErrorName,true,ESS_INCDIMBUILD_BU
ILD,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_VER
IFY,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_SAV
EOTL,"tmpotl");


 sts = EssBeginStreamBuildDim(hCtx, &RulesObj,ESS_INCDIMBUILD_BUILD,"tmpotl");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-20\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-30\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-40    600-40-20\n");
 sts = EssEndStreamBuildDim(hCtx,ErrorName,false);
```

```
 sts = EssEndIncrementalBuildDim(hCtx,ESS_DOR_ALLDATA,"tmpotl",ErrorName,false);
 return sts;
}
```

関連トピック

- EssIncrementalBuildDim
- EssBeginIncrementalBuildDim
- EssBeginStreamBuildDim
- EssEndIncrementalBuildDim
- EssEndStreamBuildDim

EssBeginUpdate

アクティブ・データベースに対して更新指定の送信を開始します。この呼出しの後、EssSendString を呼び出して更新指定を送信し、最後に EssEndUpdate を呼び出す必要があります。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

構文

```
ESS_FUNC_M EssBeginUpdate (
hCtx, Store, Unlock
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。

備考

- この関数を呼び出した場合は、その後で EssSendString を 1 回以上呼び出し、最後に EssEndUpdate を実行する必要があります。
- この関数の後に呼び出した EssSendString へ渡される各文字列の末尾は、改行復帰文字シーケンス("\r\n")である必要があります。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。
- Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数の呼出し直後に、テキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必

要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_VOID_T
ESS_BeginUpdate(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_BOOL_T  Store;
 ESS_BOOL_T  Unlock;
 ESS_STR_T   Query;

 Store = ESS_TRUE;
 Unlock = ESS_FALSE;
 Query = "Year Market Scenario Measures Product 12345";

 /* Begin Update */
 sts = EssBeginUpdate (hCtx, Store, Unlock);

 /* Send update specification */
 if(!sts)
  sts = EssSendString(hCtx, Query);

 /* End Update */
 if(!sts)
  sts = EssEndUpdate(hCtx);
}
```

関連トピック

- EssBeginReport
- EssEndUpdate
- EssSendString
- EssUpdate
- EssUpdateFile

EssBuildDimension

データ・ファイルまたはルール・ファイルに対して、アクティブなデータベースのアウトラインのメンバーに関する追加または削除を実行します。

構文

```
      ESS_FUNC_M EssBuildDimension (
      hCtx, rulesObj, dataObj,
mbrUser, ErrorName
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRulesObj	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pDataObj	159 ページの「ESS_OBJDEF_T」	データ・ファイル・オブジェクト定義構造体へのポインタ。
pMbrUser	157 ページの「ESS_MBRUSER_T」	SQL ユーザー構造体(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
ErrorName	ESS_STR_T	クライアントでのエラー出力ファイルの名前。

備考

- MbrUser が NULL 以外の場合、SQL データ・ソースとみなされます。
- データ・ソースのインポートについては、EssImport を参照してください。
- データベースは、アクティブ・データベースである必要があります。EssSetActive を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_BuildDim(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_OBJDEF_T  RulesObj;
 ESS_OBJDEF_T  DataObj;
 ESS_MBRUSER_T User;
 ESS_STR_T    ErrorName;

 RulesObj.hCtx    = hCtx;
 RulesObj.FileName = "Prodmap";
 RulesObj.ObjType  = ESS_OBJTYPE_RULES;

 DataObj.hCtx    = hCtx;
 DataObj.FileName  = "Prodtabl";
 DataObj.ObjType   = ESS_OBJTYPE_TEXT;
```

```
 ErrorName     = "builddim.err";

 sts        = EssBuildDimension (hCtx, &RulesObj, &DataObj,
          NULL, ErrorName);
 return (sts);
 /*****************************************************************/
 /*                                */
 /* When a SQL data source is defined in the rules file, define   */
 /* the variables in the ESS_OBJDEF_T DataObj structure as follows: */
 /*   DataObj.hCtx   = hCtx;                  */
 /*   DataObj.AppName  = NULL;                  */
 /*   DataObj.DbName   = NULL;                  */
 /*   DataObj.ObjType  = ESS_OBJTYPE_NONE;            */
 /*   DataObj.FileName = NULL;                  */
 /*                                */
 /* Also, provide strings for the variables in the ESS_MBRUSER_T   */
 /* User structure; for example:                  */
 /*   User.User    = "Dbusernm";                */
 /*   User.Password = "Dbpasswd";                 */
 /*                                */
 /* Use a blank string for User and Password, if the SQL source   */
 /* does not require user and password information; for example:   */
 /*   User.User    = "";                     */
 /*   User.Password = "";                      */
 /*                                */
 /* Also, define sts as follows:                  */
 /*   sts =  EssBuildDimension (hCtx, &RulesObj, &DataObj,     */
 /*      &User, ErrorName);                 */
 /*                                */
 /*****************************************************************/
}
```

関連トピック

- EssImport
- EssBuildDimFile
- EssBuildDimStart
- EssOtlRestructure

EssBuildDimFile

アクティブなデータベース・アウトラインに対するメンバーの追加または削除に使用する、データ・ファイルを作成します。詳細は EssBuildDimension を参照してください。

構文

```
      ESS_FUNC_M EssBuildDimFile (
      hCtx, RulesObj, DataObj, MbrUser,
ErrorName, fOverwriteErrorFile
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RulesObj	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
DataObj	159 ページの「ESS_OBJDEF_T」	データ・ファイル・オブジェクト定義構造体へのポインタ。
MbrUser	157 ページの「ESS_MBRUSER_T」	SQL ユーザー構造体(データ・ソースが SQL データベースの場合)。構造体が NULL の場合は、SQL 以外のデータ・ソースです。
ErrorName	ESS_STR_T	クライアントに出力されたエラー名。
fOverwriteErrorFile	ESS_BOOL_T	この関数により既存のファイル名 ErrorFile を上書きするかどうかを指定するブール値。

備考

- MbrUser が NULL 以外の場合、SQL データ・ソースとみなされます。
- データ・ソースのインポートについては、EssImport の説明を参照してください。
- データベースは、アクティブ・データベースである必要があります。EssSetActive の説明を参照してください。
- EssBuildDimStart は、この関数を使用する前に呼び出しておく必要があります。
- 再構築を行う前に繰り返しこの関数を呼び出し、複数のルール・ファイルまたはデータ・ファイル(あるいはその両方)を使用してアウトラインにメンバーを追加できます。
- この関数の呼出しが完了した後、データベースを再構築する必要があります。
- 再構築後、アウトラインのロックを解除する必要があります。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、指定したデータベースに対するデータベース・デザイン権限 ESS_PRIV_DBDESIGN を持っている必要があります。

例

```
    ESS_FUNC_M EssBuildDimFile (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M sts = ESS_STS_NOERR;
 ESS_OBJDEF_T RulesObj;
 ESS_OBJDEF_T DataObj;
 ESS_STR_T ErrorName;

 RulesObj.hCtx = hCtx;
 RulesObj.FileName = "Prodmap";
 RulesObj.ObjType = ESS_OBJTYPE_RULES;
```

```
 DataObj.hCtx = hCtx;
 DataObj.FileName = "Prodtabl";
 DataObj.ObjType = ESS_OBJTYPE_TEXT;
 ErrorName = "builddim.err";

 sts = EssBuildDimFile (hCtx, &RulesObj,
    &DataObj, NULL, ErrorName);
 return (sts);
}
```

関連トピック

- EssImport
- EssBuildDimension
- EssBuildDimStart
- EssOtlRestructure
- EssUnlockObject

EssBuildDimStart

アクティブ・データベース・アウトラインにメンバーを追加または削除するプロセスを開始します。

構文

```
ESS_FUNC_M  EssBuildDimStart (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- データ・ソースのインポートについては、EssImport の説明を参照してください。
- データベースは、アクティブ・データベースである必要があります。EssSetActive の説明を参照してください。
- アウトライン・オブジェクトは **EssBuildDimStart** を呼び出す前にロックする必要があります。EssLockObject の説明を参照してください。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M Ess_BuildDimStart (ESS_HCTX_T hCtx)
{

 sts = EssBuildDimStart (hCtx);
 return (sts);
}
```

関連トピック

- EssImport
- EssBuildDimension
- EssBuildDimFile
- EssLockObject
- EssOtlRestructure

EssCalc

単一の文字列を送信します。この関数は、EssBeginCalc を呼出し、次に EssSendString を呼び出して、最後に EssEndCalc を呼び出すのと同じです。計算を開始することも、計算スクリプトの確認のみを行い、エラーがあれば戻させることもできます。

構文

```
ESS_FUNC_M EssCalc (
hCtx, Calculate, CalcScript
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Calculate	ESS_BOOL_T	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行され、呼出しは非同期になります。
CalcScript	ESS_STR_T	単一の文字列としての計算スクリプト(64KB 未満)。

備考

- 計算スクリプトの文字列の長さは、64KB 未満にする必要があります。
- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- この API 呼出しは Calculate パラメータが TRUE の場合にのみ非同期になります。それ以外の場合は、簡単な同期要求です。

  非同期要求中は、要求の完了前に、制御がプログラムにすぐに戻されます。非同期要求の実行中は、現在の API コンテキスト・ハンドルに有効な要求の

セットは限定されます。その間に無効な要求を渡すと、エラーが戻されます。非同期操作中の API コンテキストに有効な API 呼出しは次のとおりです: EssGetProcessState、EssCancelProcess。

- Calculate フラグが FALSE に設定されている場合、データベースは計算スクリプトの構文チェックのみを行い、呼び出しは同期です。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CalcLine  (ESS_HCTX_T      hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    Script;
 ESS_PROCSTATE_T  pState;

 Script = "CALC ALL;";
 sts = EssCalc(hCtx, ESS_TRUE, Script);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
      ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
 }
 return(sts);
}
```

関連トピック

- EssBeginCalc
- EssCalcFile
- EssDefaultCalc
- EssEndCalc
- EssGetDefaultCalc
- EssGetProcessState
- EssSendString
- EssSetDefaultCalc

EssCalcFile

ファイルからアクティブなデータベースに対して計算スクリプトを実行します。

構文

```
      ESS_FUNC_M
      EssCalcFile
       (
      hDestCtx, hSrcCtx, AppName, DbName,
FileName, Calculate
      );
```

パラメータ　データ型　説明

hDestCtx	ESS_HCTX_T	Essbase サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	計算スクリプト・ファイルの場所の API コンテキスト・ハンドル。計算スクリプト・ファイルは、クライアント・コンピュータ上またはターゲット・データベースと同じ Essbase サーバー・コンピュータ上に存在することができます。
AppName	ESS_STR_T	計算スクリプト・ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	計算スクリプト・ファイルの場所のデータベース名。
FileName	ESS_STR_T	計算スクリプト・ファイルの名前。
Calculate	ESS_BOOL_T	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行され、呼出しは非同期になります。

備考

- 計算スクリプトのサイズは 64KB を超えることはできません。
- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- この API 呼出しは Calculate パラメータが TRUE の場合にのみ非同期になります。それ以外の場合は、簡単な同期要求です。

  非同期要求中は、要求の完了前に、制御がプログラムにすぐに戻されます。非同期要求の実行中は、現在の API コンテキスト・ハンドルに有効な要求のセットは限定されます。その間に無効な要求を渡すと、エラーが戻されます。非同期操作中の API コンテキストに有効な API 呼出しは次のとおりです:
  EssGetProcessState、EssCancelProcess。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
```

```
ESS_CalcFile (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_SHORT_T   isResponse;
 ESS_HCTX_T  hSrcCtx;
 ESS_BOOL_T  isObject = ESS_FALSE;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;
 ESS_PROCSTATE_T pState;

 hSrcCtx  = hCtx;
 AppName  = "Sample";
 DbName   = "Basic";
 FileName = "Test";

 sts = EssCalcFile (hCtx, hSrcCtx, AppName,
    DbName, FileName, ESS_TRUE);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
      ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 return(sts);
}
```

関連トピック

- EssBeginCalc
- EssCalc
- EssDefaultCalc
- EssSetDefaultCalcFile

EssCalcFileWithRuntimeSubVars

指定したランタイム代替変数のアクティブ・データベースに対して、計算スクリプトを実行します。ランタイム代替変数は、拡張子が.rsv のテキスト・ファイル内で(このファイルはクライアント・コンピュータに存在する必要があります)、またはキー/値ペアの文字列として指定できます。

構文

```
ESS_FUNC_M EssCalcFileWithRuntimeSubVars (
hDestCtx
,
hSrcCtx
,
AppName
,
DbName
,
FileName
```

```
,
RtSV
,
bRtSVFile
,
Calculate
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	Essbase サーバー上のターゲット・データベースの API コンテキスト・ハンドル
hSrcCtx	ESS_HCTX_T	計算スクリプト・ファイルの場所の API コンテキスト・ハンドル。計算スクリプト・ファイルは、クライアント・コンピュータ上またはターゲット・データベースと同じ Essbase サーバー上に存在することができます。
AppName	ESS_STR_T	計算スクリプト・ファイルに関連付けられているアプリケーションの名前
DbName	ESS_STR_T	計算スクリプト・ファイルに関連付けられているデータベースの名前
FileName	ESS_STR_T	計算スクリプト・ファイルの名前。計算スクリプト・ファイルは、ターゲット・データベースの場所と同じ Essbase サーバー上、またはクライアント・コンピュータ上に存在することができます。

RtSV	ESS_STR_T	次のオプションのいずれか:
		● ランタイム代替変数ファイルの名前とフル・パス(たとえば、C:\myRTSVfile.rsv)。これはクライアント・コンピュータ上に存在する必要があります。Essbase は、Essbase サーバー・コンピュータ上にあるランタイム代替変数ファイルをサポートしていません。
		ランタイム代替変数ファイルは、拡張子が.rsv のテキスト・ファイルとして作成する必要があります。ファイルの各行では、1 つのランタイム代替変数をキー/値ペアとして定義し、セミコロンで終える必要があります。この.rsv ファイルの例では、4 つのランタイム代替変数の名前と値が指定されています(たとえば、"a"という名前のランタイム代替変数の値は 100 です):
		`a=100;`
		`b=200;`
		`c=@CHILDREN("100");`
		`d=@TODATE("DD/MM/YY","10/11/12");`
		● ランタイム代替変数のキー/値ペアの文字列。文字列は一重引用符で囲み、キー/値ペアはセミコロンで区切る必要があります。このランタイム代替変数文字列の例では、4 つのランタイム代替変数の名前と値が指定されています(たとえば、"a"という名前のランタイム代替変数の値は 100 です):
		`'a=100;b=@CHILDREN("100");c="Actual"->"Final";d="New York";'`
		この引数は、bRtSVFile 引数とともに使用されます。

bRtSVFile	ESS_BOOL_T	RtSV 引数がランタイム代替変数ファイルの名前とフル・パスを参照するか(TRUE)、ランタイム代替変数のキー/値ペアの文字列を参照するか(FALSE)を示すフラグ。

パラメータ	データ型	説明
Calculate	ESS_BOOL_T	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行され、呼出しは非同期になります。

戻り値

なし。

アクセス

この関数を呼び出すには、アクティブ・データベースに対する計算権限(ESS_PRIV_CALC)が必要です。

例

```
    void Ess_CalcFileWithRuntimeSubVars(ESS_HINST_T hInst, ESS_HCTX_T hCtx)
{
 ESS_STS_T       sts;
 ESS_STR_T AppName   = "Sample";
 ESS_STR_T DbName    = "Basic";
 ESS_STR_T FileName  = "testrt"; \\ Server side calc script file. PLease provide
this when using server side calc script file
 //ESS_STR_T FileName = "D:\\temp\\testrt.csc"; \\Client side calc script file.
 //ESS_STR_T Param = "D:\\temp\\temp1.rsv"; \\ Client side param file.
 ESS_STR_T Param    = "mySales=700"; \\ Client side param string.
 ESS_BOOL_T Calculate = TRUE;
 ESS_ACCESS_T   Access;

 ESS_PROCSTATE_T pState;

 ESS_HCTX_T    hLocalCtx = ESS_INVALID_HCTX;
 sts = EssCreateLocalContext (hInst, ESS_NULL, ESS_NULL, &hLocalCtx);

 //hLocalCtx = hCtx;
 //sts = EssSetActive (hCtx, AppName, DbName, &Access);

 //For calc file on server With Param String
 sts = EssCalcFileWithRuntimeSubVars(hCtx, hCtx, AppName, DbName, FileName, Param,
FALSE, Calculate);

 //For Calc file on client With Param String
 sts = EssCalcFileWithRuntimeSubVars(hCtx, hLocalCtx, NULL, NULL, FileName, Param,
FALSE, Calculate);

 //For calc file on server With Param File
 sts = EssCalcFileWithRuntimeSubVars(hCtx, hCtx, AppName, DbName, FileName, Param,
TRUE, Calculate);

 //For calc file on client With Param File
 sts = EssCalcFileWithRuntimeSubVars(hCtx, hLocalCtx, NULL, NULL, FileName, Param,
TRUE, Calculate);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while(!sts && (pState.State !=
```

```
      ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
  }

  if(sts)
   printf("API could not be executed.");
}
```

関連トピック

- EssGetRuntimeSubVars
- EssCalcWithRuntimeSubVars
- EssGetProcessState
- EssCancelProcess
- EssBeginCalc
- EssCalc
- EssDefaultCalc
- EssSetDefaultCalc

EssCalcWithRuntimeSubVars

指定したランタイム代替変数(キー/値ペアの文字列として指定)を使用して計算スクリプトを実行します。計算を開始することも、計算スクリプトの確認のみを行い、エラーがあれば戻すこともできます。

構文

```
ESS_FUNC_M EssCalcWithRuntimeSubVars (
hCtx
,
CalcScript
,
RtSV
,
Calculate
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
CalcScript	ESS_STR_T	単一文字列としての計算スクリプト。 計算スクリプトの文字列は 64KB を超えることはできません。
RtSV	ESS_STR_T	ランタイム代替変数のキー/値ペアを指定する文字列。文字列は一重引用符で囲み、キー/値ペアはセミコロンで区切る必要があります。このランタイム代替変数文字列の例では、4 つのランタイム代替変数の名前と値が指定されています(たとえば、"a"という名前のランタイム代替変数の値は 100 です): `'a=100;b=@CHILDREN("100");c="Actual"->"Final";d="New York";'`

パラメータ	データ型	説明
Calculate	ESS_BOOL_T	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行され、呼出しは非同期になります。

備考

- この関数が正しく実行され、計算が開始されると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行されます。呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- この API 呼出しは Calculate パラメータが TRUE の場合にのみ非同期になります。それ以外の場合、呼出しは単純な同期要求です。

  非同期要求中は、要求の完了前に、制御がプログラムにすぐに戻されます。非同期要求の実行中は、現在の API コンテキスト・ハンドルに有効な要求のセットは限定されます。その間に無効な要求が行われると、エラーが戻されます。非同期操作中の API コンテキストに対して有効な API 呼出し: EssGetProcessState、EssCancelProcess。
- Calculate パラメータが FALSE に設定されている場合、データベースは計算スクリプトの構文チェックのみを行い、呼出しは同期です。

戻り値

なし。

アクセス

この関数を呼び出すには、アクティブ・データベースに対する計算権限 (ESS_PRIV_CALC)が必要です。

例

```
    ESS_FUNC_M
ESS_CalcWithRuntimeSubVars(ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    Script;
 ESS_STR_T    ParamString = "mySales=700;myCOGS=100;";
 ESS_PROCSTATE_T pState;

 Script = "SET RUNTIMESUBVARS {salesNum =400; mySales=300;
myRTVar=@CHILDREN(\"100\"); kmdsdmclms=@TODATE(\"DD/MM/YY\",\"10/11/12\");
myCOGS=50;};FIX (@INTERSECT(&myRTVar, \"100-10\")) Sales = &mySales; COGS=&myCOGS;
ENDFIX;";

 sts = EssCalcWithRuntimeSubVars(hCtx, Script, ParamString, ESS_TRUE);

 if (!sts)
  printf ("\r\nAPI EssCalcWithParam executed successfully...\r\n\r\n");
}
```

関連トピック

- EssCalcFileWithRuntimeSubVars

- EssGetRuntimeSubVars
- EssGetProcessState
- EssCancelProcess
- EssBeginCalc
- EssCalc
- EssDefaultCalc
- EssSetDefaultCalc

EssCancelAsyncProc

非同期データ・ロードまたは次元構築プロセスを取り消します。

構文

```
ESS_FUNC_M EssCancelAsyncProc (
hCtx, ErrorFileName, ErFileOverWrite
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ErrorFileName	ESS_STR_T	エラー・ファイル名。
ErFileOverWrite	ESS_BOOL_T	TRUE の場合、エラー・ファイルを上書きします。

備考

EssAsyncImport または EssAsyncBuildDim を使用して非同期プロセスが開始された後に、この関数を呼び出します。

戻り値

正常終了の場合はネットワーク接続を閉じてエラー・ログが戻されます。それ以外の場合、エラー・コードが戻されます。

例

EssAsyncBuildDim の例を参照してください。

関連トピック

- EssAsyncBuildDim
- EssAsyncImport
- EssAsyncImportASO
- EssGetAsyncProcLog
- EssGetAsyncProcState
- EssCloseAsyncProc

EssCancelProcess

まだ完了していない非同期プロセスを取り消します

構文

```
ESS_FUNC_M EssCancelProcess (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数を使用してプロセスを取り消した場合、一部のデータのみ再計算され、一貫性が失われた状態のままデータベースが残ることがあります。
- 非同期データベース操作(たとえば計算)が正しく開始された後以外にこの関数を呼び出すと、エラーが発生します。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_VOID_T
ESS_CancelProcess(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 ESS_STR_T     Script;
 ESS_PROCSTATE_T  pState;
 ESS_USHORT_T    Count;

 Script = "CALC ALL;";

 sts = EssBeginCalc (hCtx,ESS_TRUE);

 if (!sts)
  sts = EssSendString (hCtx, Script);
 if (!sts)
  sts = EssEndCalc (hCtx);
 /************************************
  Check process state and cancel it
  if it takes too long
 ************************************/

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while(!sts && (pState.State !=
     ESS_STATE_DONE))
  {
   Count = Count + 1;
   if (Count == 1000)
```

```
      sts = EssCancelProcess(hCtx);

    sts = EssGetProcessState (hCtx, &pState);
   }
  }
}
```

関連トピック

- EssBeginCalc
- EssCalc
- EssGetProcessState
- EssImport

EssCheckAttributes

指定したメンバーごとに属性情報を戻します。

構文

```
ESS_FUNC_M EssCheckAttributes (
hCtx, Count, pMemberNameArray, ppAttributeTypeArray
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
Count;	ESS_USHORT_T	指定した次元およびメンバーの数。
pMemberNameArray;	ESS_PMBRNAME_T	指定した次元およびメンバーの名前の配列。
ppAttributeTypeArray;	ESS_PPUSHORT_T	属性のタイプの配列に対する、次の定数識別子のいずれか (表 6 を参照): ● ESS_ATTRIBUTE_DIMENSION ● ESS_ATTRIBUTE_MEMBER ● ESS_STANDARD_DIMENSION ● ESS_STANDARD_MEMBER ● ESS_BASE_DIMENSION ● ESS_BASE_MEMBER ● ESS_ATTRIBUTED_MEMBER ● ESS_INVALID_MEMBER

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    void  ESS_CheckAttributes()
{
 ESS_STS_T    sts=-1,sts1=-1;
```

```
 int      counter,i,j;
 ESS_PMBRNAME_T  pMbrNames=ESS_NULL;
 ESS_PUSHORT_T  pMbrAttrTypes=ESS_NULL;
 ESS_CHAR_T   buf[80]="";

 /* counter = 4; */
 printf("Please enter the number of member names that follow: ");
 gets(buf);
 counter=atoi(buf);

 if (counter)
 {
   sts1 = EssAlloc(hInst, (counter * sizeof(ESS_MBRNAME_T)),
(ESS_PPVOID_T)&pMbrNames);
  if (!sts1)
  {
   memset(pMbrNames, 0, (counter * sizeof(ESS_MBRNAME_T)));

   for (i = 0; i < counter; i++)
   {
    printf("Enter member name: ");
    gets(buf);
    strcpy(pMbrNames[i],buf);
   }

   sts = EssCheckAttributes(hCtx,counter,pMbrNames,&pMbrAttrTypes);
   if (sts)
    fprintf(stderr, "sts = %ld \n",sts);
   else if (pMbrAttrTypes)
   {
    for (j = 0; j < counter; j++)
    {
     switch(pMbrAttrTypes[j])
     {
      case ESS_STANDARD_MEMBER:
       strcpy(buf,"ESS_STANDARD_MEMBER");
       break;

      case ESS_STANDARD_DIMENSION:
       strcpy(buf,"ESS_STANDARD_DIMENSION");
       break;

      case ESS_BASE_MEMBER:
       strcpy(buf,"ESS_BASE_MEMBER");
       break;

      case ESS_BASE_DIMENSION:
       strcpy(buf,"ESS_BASE_DIMENSION");
       break;

      case ESS_ATTRIBUTE_MEMBER:
       strcpy(buf,"ESS_ATTRIBUTE_MEMBER");
       break;

      case ESS_ATTRIBUTE_DIMENSION:
       strcpy(buf,"ESS_ATTRIBUTE_DIMENSION");
       break;
```

```
          case ESS_ATTRIBUTED_MEMBER:
           strcpy(buf,"ESS_ATTRIBUTED_MEMBER");
           break;

          default:
           strcpy(buf,"Unknown attribute type");
         }
        printf("%s is of type %s\n",pMbrNames[j],buf);
       }
       printf("\n");
     }
    }
  }
}
```

関連トピック

- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssCheckMemberName

文字列がアクティブ・データベース・アウトライン内で有効なメンバー名であるかどうか確認します。

構文

```
ESS_FUNC_M EssCheckMemberName (
hCtx, MbrName, pValid
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
MbrName	ESS_STR_T	確認されるメンバー名。

パラメータ	データ型	説明
pValid	ESS_PBOOL_T	有効なメンバー・フラグを受け取る変数のアドレス。メンバーが有効な場合は TRUE に設定します。

備考

この関数は、リレーショナル・スパンのブール式が設定されており、指定されたメンバー名のリレーショナル・ストアでの有効性を判断できるかどうかを確認します。

戻り値

正常終了の場合、名前の文字列 MbrName がアクティブ・データベース・アウトラインの中で有効なメンバー名であるかどうかを示すフラグ pValid が戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_CheckMemberName(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts;
 ESS_STR_T    MbrName;
 ESS_BOOL_T   pValid;

 MbrName = "Profit";
 sts = EssCheckMemberName(hCtx, MbrName, &pValid);

 if(pValid)
  printf("\"%s\" is a valid member name\n",
  MbrName);

 return (sts);
}
```

関連トピック

- EssGetMemberInfo
- EssQueryDatabaseMembers
- EssSetActive

EssClearActive

ユーザーの現在のアクティブなアプリケーションおよびデータベースを消去します。

構文

```
ESS_FUNC_M EssClearActive (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_UnloadDb (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";
 /*
  * IF the current active is the same as the
  * unload db, ClearActive first
  */
   sts = EssClearActive(hCtx);
 /*
  * ELSE
  *
  */
   sts = EssUnloadDatabase(hCtx, AppName,
      DbName);
 return (sts);
}
```

関連トピック

- EssGetActive
- EssSetActive

EssClearAliases

アクティブなデータベースのすべての別名テーブルを完全に削除します。

構文

```
ESS_FUNC_M EssClearAliases (
```

```
    hCtx
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数は、アクティブな別名テーブルもデフォルトの別名テーブルも削除できません。
- この API 関数を使用する前に、EssSetAlias を使用してアクティブな別名を"default"に設定します。
- EssListConnections を呼び出して、別名テーブルを消去しようとしているデータベースが他のユーザーによって使用されていないことを確認します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_ClearAliases (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 sts = EssClearAliases(hCtx);
 if(!sts)
  printf("All alias tables are removed.\r\n");
 return (sts);
}
```

関連トピック

- EssListAliases
- EssRemoveAlias
- EssSetActive

EssClearDatabase

アクティブ・データベース内にロードされているすべてのデータを消去します。

注意 この関数を使用して削除されたデータは復元できません。注意して使用してください!

構文

```
ESS_FUNC_M EssClearDatabase (
hCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が少なくともデータベースへの書込み権限(ESS_PRIV_WRITE)を持ち、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_ClearDb (ESS_HCTX_T   hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 sts = EssClearDatabase(hCtx);
 return (sts);
}
```

関連トピック

- EssDeleteDatabase
- EssUnloadDatabase
- EssSetActive

EssCloseAsyncProc

終了または取り消した非同期の次元構築またはデータ・ロードの接続を閉じて、プロセスの現在の状態が戻されます。

構文

```
ESS_FUNC_M EssCloseAsyncProc (
hCtx, ProcState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ProcState	ESS_PBLDDL_STATE_T	割り当てられたプロセス状態構造体を受け取るポインタのアドレス。

備考

EssAsyncImport または EssAsyncBuildDim を使用して非同期プロセスが開始された後に、この関数を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

例

EssAsyncBuildDim の例を参照してください。

関連トピック

- EssAsyncBuildDim
- EssAsyncImport
- EssGetAsyncProcLog
- EssGetAsyncProcState
- EssCancelAsyncProc

EssClrSpanRelationalSource

Essbase に関係するデータが接続したリレーショナル・ストアに存在することを通知する、ブール bSpanRelPart フィールドを消去します。
EssQueryDatabaseMembers などのその他の API 関数の一部は、bSpanRelPart を読み込み、bSpanRelPart が設定されている場合はリレーショナル・ストアにアクセスします。

構文

```
ESS_FUNC_M EssClrSpanRelationalSource (
hCtx
);
```

パラメータ　データ型　　説明

hCtx　　ESS_HCTX_T API コンテキスト・ハンドル。

備考

一部の API 関数は、リレーショナル・ストアから情報を取得するように拡張されています。

- EssQueryDatabaseMembers - リレーショナル・ストアからメンバー名を戻します。
- EssGetMemberInfo - リレーショナル・ストア内のメンバーに関する情報を戻します。
- EssCheckMemberName - リレーショナル・ストアで有効なメンバー名を確認します。

- EssGetMemberCalc - 入力として渡されたリレーショナル・メンバーを認識し、すべてのリレーショナル・メンバーに対して NULL 文字列を戻します。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
      ESS_HINST_T  hInst
     )
{
 ESS_FUNC_M    sts    = ESS_STS_NOERR;
 ESS_STR_T   rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");
 if (!sts)
  sts = EssClrSpanRelationalSource (hCtx);
 /**************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

関連トピック

- EssSetSpanRelationalPartition

EssCommitDatabase

使用されなくなりました。コミットは Essbase サーバーにより自動的に処理されます。この関数は、エラー・メッセージ ESS_STS_OBSOLETE を戻します。データのコミットの詳細は、『Oracle Essbase データベース管理者ガイド』を参照してください。

EssCompactOutline

サーバー側でコンパクト化が必要なアウトライン・ファイルをコンパクト化します。

構文

```
ESS_FUNC_M EssCompactOutline (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	ログイン時に取得した API コンテキスト。

備考

- この関数を使用する場合は、ユーザーがアクティブに設定されている必要があります。

戻り値

成功の場合、0 が戻されます。このアクションを行っているユーザーが存在しないことを確認後、アウトラインのみの再構築が行われます。

例

```
    #include <windows.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#pragma pack(push, api, 1)
#include <essapi.h>
#include <essotl.h>
#pragma pack(pop, api)

/* default names */
ESS_SVRNAME_T  srvrName    =    "localhost";
ESS_USERNAME_T  userName    =    "essexer";
ESS_PASSWORD_T  pswd      =    "password";
ESS_APPNAME_T  app       =    "ASOSamp";
ESS_DBNAME_T   db       =    "Sample";

int main(int argc, char *argv[ ])
{

    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_HINST_T hInst = NULL;
    ESS_HOUTLINE_T hOutlineQuery = NULL, hOutline = NULL;
    ESS_HCTX_T hCtx = NULL;
    ESS_USHORT_T Items;
    ESS_PAPPDB_T pAppsDbs = NULL;
    ESS_ACCESS_T Access;

    ESS_INIT_T InitStruct =    /* Define init */
```

```
                /* structure */
    {
        ESS_API_VERSION,    /* Version of API */
            (ESS_PVOID_T)0,   /* user-defined message context */
            0,     /* max handles   */
            0L,     /* max buffer size */
            NULL, //(ESS_STR_T)"C:\\Hyperion\\products\\Essbase\\EssbaseServer", /*
local path */
            /* The following parameters use defaults  */
            NULL,    /* message db path */
            NULL,    /* allocation function pointer */
            NULL,    /* reallocation function pointer */
            NULL,    /* free function pointer */
            NULL, //(ESS_PFUNC_T)MessageFunc,  /* error handling function pointer */
            NULL,    /* path name of user-defined */
            /* Application help file */
            0L    /* Reserved for internal use. */
            /* Set to NULL */
#ifdef AD_UTF8
            , ESS_API_UTF8
#endif
    };

    /* get appname and dbname from the argument list */
    if (argc < 6) {
        puts(" Usage: EssCompactOtl ServerName Userid Password AppName DbName\n");
        exit (0);
  }

    strcpy(srvrName, argv[1]);
  strcpy(userName, argv[2]);
    strcpy(pswd, argv[3]);
    strcpy(app, argv[4]);
    strcpy(db, argv[5]);

    /* Initialize the Essbase API */
    if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
    {
     printf("EssInit failure: %ld\n", sts);
     exit ((int) sts);
    }

    /* Login to Essbase */
    if ((sts = EssLogin (hInst, srvrName, userName, pswd, &Items, &pAppsDbs, &hCtx)) !
= ESS_STS_NOERR)
    {
     printf("EssLogin failure: %ld\n", sts);
     exit ((int) sts);
    }

    if(pAppsDbs)
        EssFree(hInst, pAppsDbs);

    /* Select the application */
    if ((sts = EssSetActive(hCtx, app, db, &Access)) != ESS_STS_NOERR)
    {
     printf("EssSetActive failure: %ld\n", sts);
```

```
     exit ((int) sts);
    }

    /* compact the outline and restructure */
    if ((sts = EssCompactOutline(hCtx)) != ESS_STS_NOERR)
    {
     printf("EssCompactOutline failure: %ld\n", sts);
     exit ((int) sts);
    }

    /* done, logout and terminate the api */
    if ((sts = EssLogout (hCtx)) != ESS_STS_NOERR)
    {
     printf("EssLogout failure: %ld\n", sts);
     exit ((int) sts);
    }

    if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
    {
        /* error terminating API */
        exit((int) sts);
    }

    return(0);
}
```

EssConvertApplicationtoUnicode

Unicode モードのアプリケーションを作成します。Unicode モードとして定義されている場合、Essbase サーバーでは、非 Unicode モードのアプリケーションを Unicode モードに移行できます。

構文

```
ESS_FUNC_M EssConvertApplicationToUnicode(
      hCtx
      ,
      AppName
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	移行するアプリケーションの名前。指定されたアプリケーションが存在し、Unicode モードになっていないことが必要です。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

呼出し元が、アプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

関連トピック

- EssCreateApplicationEx

EssCopyApplication

クライアント上またはサーバー上の既存のアプリケーションを、関連するすべてのデータベースとオブジェクトも含めて、新規アプリケーションにコピーします。

構文

```
ESS_FUNC_M EssCopyApplication (
hCtx, hSrcCtx, SrcApp, DestApp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	使用されていません - hCtx と同じになります。
SrcApp	ESS_STR_T	コピーする既存のアプリケーションの名前。
DestApp	ESS_STR_T	新規アプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

- クライアント・アプリケーションをコピーすると、ローカル・アプリケーションのディレクトリとコンテンツもコピーされます。
- この関数は、クライアント・アプリケーションをクライアント上の新規アプリケーションにコピーする場合またはサーバー・アプリケーションを同じサーバー上の新規アプリケーションにコピーする場合にのみ使用できます。異なるサーバーにアプリケーションをコピーする場合は、EssCopyObject を使用します。
- 新規アプリケーションは起動されません。EssLoadApplication を呼び出して、新たにコピーされたアプリケーションを起動します。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)およびコピー元のソース・アプリケーションに対するアプリケーション・デザイナ権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CopyApp(ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_STR_T   SrcApp;
 ESS_STR_T   DestApp;

 hSrcCtx = hCtx;
 SrcApp = "Sample";
 DestApp = "NewSamp";

 sts = EssCopyApplication(hCtx, hSrcCtx, SrcApp,
    DestApp);
 return (sts);
}
```

関連トピック

- EssCopyDatabase
- EssCopyObject
- EssLoadApplication

EssCopyDatabase

クライアント上またはサーバー上の既存のデータベースを、関連するすべてのデータベースおよびオブジェクトも含めて、新規のデータベースにコピーします。サーバーにデータベースがコピーされた場合、新しいデータベースが起動されます。

構文

```
ESS_FUNC_M EssCopyDatabase (
hCtx, hSrcCtx, SrcApp, DestApp, SrcDb, DestDb
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	使用されていません – hCtx と同じになります。
SrcApp	ESS_STR_T	ソース・アプリケーション名。
DestApp;	ESS_STR_T	宛先アプリケーションの名前。
SrcDb;	ESS_STR_T	コピーする既存のデータベース名。
DestDb	ESS_STR_T	新規データベースの名前。1903 ページの「データベース名の制限」を参照してください。

備考

- クライアントのデータベースをコピーすると、ローカル・データベースのディレクトリとコンテンツもコピーされます。
- この関数は、クライアントのデータベースをクライアント上の別のデータベースにコピーする場合、またはサーバーのデータベースを同じサーバー上の別のデータベースにコピーする場合にのみ使用できます。異なるサーバー間でデータベースをコピーする場合は、EssCopyObject を使用します。

戻り値

なし。

アクセス

サーバー・データベースの場合、呼出し元がデータベースの作成/削除/編集権限(ESS_PRIV_DBCREATE)およびコピー元のソース・データベースに対するデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CopyDatabase(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_HCTX_T   hSrcCtx;
 ESS_STR_T    SrcApp;
 ESS_STR_T    DestApp;
 ESS_STR_T    SrcDb;
 ESS_STR_T    DestDb;

 hSrcCtx = hCtx;
 SrcApp  = "Sample";
 DestApp = "NewSamp";
 SrcDb   = "Basic";
 DestDb  = "NewBasic";

 sts = EssCopyDatabase(hCtx, hSrcCtx, SrcApp,
    DestApp, SrcDb, DestDb);

 return(sts);
}
```

関連トピック

- EssCopyApplication
- EssCopyObject

EssCopyFilter

既存のフィルタをコピーします。

構文

```
ESS_FUNC_M EssCopyFilter (
hCtx, hSrcCtx, SrcApp, DestApp, SrcDb, DestDb, SrcName, DestName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	使用されていません - hCtx と同じになります。
SrcApp	ESS_STR_T	コピー元アプリケーション名。
DestApp	ESS_STR_T	コピー先アプリケーション名。
SrcDb	ESS_STR_T	コピー元データベース名。
DestDb	ESS_STR_T	宛先データベース名。
SrcName	ESS_STR_T	コピーする既存のフィルタのソース名。
DestName	ESS_STR_T	コピーされるフィルタのコピー先の名前。1904 ページの「フィルタ名の制限」を参照してください。

備考

- ソース・フィルタが存在している必要があります。
- 既存のフィルタを誤って上書きするのを防ぐため、呼出し元はコピー先フィルタが存在しているかどうかを確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CopyFilter (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_HCTX_T     hSrcCtx;
 ESS_STR_T     SrcApp;
 ESS_STR_T     DestApp;
 ESS_STR_T     SrcDb;
 ESS_STR_T     DestDb;
 ESS_STR_T     SrcName;
 ESS_STR_T     DestName;

 hSrcCtx  = hCtx;
 SrcApp   = "Sample";
 SrcDb    = "Basic";
 SrcName  = "OldFilter";
```

```
 DestApp  = "Sample";
 DestDb   = "Basic";
 DestName  = "NewFilter";

 sts = EssCopyFilter(hCtx, hSrcCtx, SrcApp,
    DestApp, SrcDb, DestDb, SrcName, DestName);
 if(!sts)
  printf("The Filter is copied.\r\n");

 return (sts);
}
```

関連トピック

- EssDeleteFilter
- EssListFilters
- EssRenameFilter

EssCopyObject

サーバーまたはクライアントのオブジェクト・システムにオブジェクトをコピーします。

構文

```
      ESS_FUNC_M EssCopyObject (
      hSrcCtx, hDestCtx, ObjType,
SrcApp, DestApp, SrcDb, DestDb, SrcObj, DestObj
      );
```

パラメータ	データ型	説明
hSrcCtx	ESS_HCTX_T	コピー元オブジェクトの API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
hDestCtx	ESS_HCTX_T	コピー先オブジェクトの API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用できる値は、102 ページの「ビットマスク・データ型(C)」を参照してください。
SrcApp	ESS_STR_T	コピー元アプリケーション名。
DestApp	ESS_STR_T	コピー先アプリケーション名。
SrcDb	ESS_STR_T	コピー元データベース名。NULL の場合は、コピー元のアプリケーションのサブディレクトリが使用されます。
DestDb	ESS_STR_T	宛先データベース名。NULL の場合は、コピー先のアプリケーションのサブディレクトリが使用されます。
SrcObj	ESS_STR_T	コピー元のソース・オブジェクトの名前。

パラメータ	データ型	説明
DestObj	ESS_STR_T	コピー先のオブジェクトの名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- オブジェクトはクライアントからサーバーへ、サーバーからクライアントへ、または同一サーバー上でコピーできます。いずれの場合も、コピー先のオブジェクトは既存のもの以外か、呼出し元がロックしたものである必要があります。
- アウトライン・オブジェクトはコピーできません。関連付けられているアウトラインを含めてデータベースをコピーするには、EssCopyDatabase を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がオブジェクトを含む指定されたソース・アプリケーション、またはデータベース(オブジェクト・タイプに依存します)、あるいはその両方に対する適切なレベルのアクセス権と、指定された宛先アプリケーションまたはデータベースに対するアプリケーションまたはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CopyObject(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_HCTX_T     hDestCtx;
 ESS_STR_T      SrcApp;
 ESS_STR_T      DestApp;
 ESS_STR_T      SrcDb;
 ESS_STR_T      DestDb;
 ESS_STR_T      SrcObj;
 ESS_STR_T      DestObj;
 ESS_OBJTYPE_T   ObjType;

 hDestCtx  = hCtx;
 SrcApp   = "Sample";
 SrcDb    = "Basic";
 SrcObj   = "Test";
 DestApp  = "Sample";
 DestDb   = "Basic";
 DestObj  = "NewTest";
 ObjType  = ESS_OBJTYPE_TEXT;

 sts = EssCopyObject(hCtx,hDestCtx,ObjType,SrcApp,
    DestApp,SrcDb,DestDb,SrcObj,DestObj);
```

```
 if(!sts)
  printf("The Object is copied.\r\n");

 return (sts);
}
```

関連トピック

- EssCreateObject
- EssDeleteObject
- EssListObjects
- EssRenameObject

EssCreateApplication

クライアントまたはサーバー上で、新規アプリケーションを作成します。アプリケーションがサーバーで作成された場合は、起動も行われます。

構文

```
ESS_FUNC_M EssCreateApplication (
hCtx, AppName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。EssCreateDatabase または EssCreateApplication を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。
- Unicode モードのアプリケーションを作成するには、EssCreateApplicationEx を使用します。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateApp (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 AppName = "Sample";
 sts = EssCreateApplication (hCtx, AppName);
 return(sts);
}
```

関連トピック

- EssCreateStorageTypedApplication
- EssCreateDatabase
- EssCreateObject
- EssCreateApplicationEx

EssCreateApplicationEx

クライアントまたはサーバー上で、新規アプリケーションを作成します。アプリケーションがサーバーで作成された場合は、起動も行われます。この関数は、Unicode モードのアプリケーションを作成できます。

構文

```
ESS_FUNC_M EssCreateApplicationEx(
      hCtx
      ,
      AppName
      ,
      usAppType
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。
usAppType		新規アプリケーションのアプリケーション・タイプ(Unicode または非 Unicode)。 有効な値は次のとおりです: ● ESS_APP_UNICODE - 0x0003 - Unicode アプリケーションを作成します。サーバーが Unicode モードでない場合、関数は失敗します。 ● ESS_APP_NONUNICODE - 0x0002 - 非 Unicode アプリケーションを作成します。

備考

- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。EssCreateDatabaseEx または EssCreateApplicationEx を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

関連トピック

- EssCreateStorageTypedApplicationEx
- EssCreateDatabaseEx
- EssCreateObject
- EssConvertApplicationtoUnicode

EssCreateDatabase

クライアントまたはサーバー上で、アプリケーション内に新規データベースを作成します。データベースがサーバー上で作成された場合、起動も行われます。

構文

```
ESS_FUNC_M EssCreateDatabase (
hCtx, AppName, DbName, DbType
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	データベースを含むアプリケーションの名前。
DbName	ESS_STR_T	作成するデータベースの名前。1903 ページの「データベース名の制限」を参照してください。
DbType	ESS_USHORT_T	作成するデータベースのタイプ。ESS_DBTYPE_NORMAL または ESS_DBTYPE_CURRENCY

備考

- クライアント・データベースを作成すると、ローカル・データベース・ファイルを含むディレクトリが作成されます。

- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。EssCreateDatabase または EssCreateApplication を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。

戻り値

なし。

アクセス

サーバー・データベースの場合は、呼出し元がデータベースの作成/削除/編集権限(ESS_PRIV_DBCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateDb (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;

 AppName = "Sample";
 DbName  = "Basic";

 sts = EssCreateDatabase(hCtx, AppName, DbName, ESS_DBTYPE_NORMAL);
 return (sts);
}
```

関連トピック

- EssCreateApplication
- EssCreateObject

EssCreateDatabaseEx

クライアントまたはサーバー上で、アプリケーション内に新規データベースを作成します。データベースがサーバー上で作成された場合、起動も行われます。この関数は、重複しているメンバー名をサポートしているデータベースの作成に使用されます。

構文

```
ESS_FUNC_M EssCreateDatabaseEx (
hCtx, AppName, DbName, DbType, bNonUniqueName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
AppName	ESS_STR_T	データベースを含むアプリケーションの名前。
DbName	ESS_STR_T	作成するデータベースの名前。1903 ページの「データベース名の制限」を参照してください。
DbType	ESS_USHORT_T	作成するデータベースのタイプ。ESS_DBTYPE_NORMAL または ESS_DBTYPE_CURRENCY
bNonUniqueName	ESS_BOOL_T	TRUE に設定すると、この関数は重複したメンバー名に対応したアウトラインを含むデータベースを作成します。FALSE に設定すると、EssCreateDatabase と同様に機能します。

備考

- クライアント・データベースを作成すると、ローカル・データベース・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。EssCreateDatabase、EssCreateDatabaseEx または EssCreateApplication を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

サーバー・データベースの場合は、呼出し元がデータベースの作成/削除/編集権限(ESS_PRIV_DBCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_CreateDb()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T AppName;
 ESS_STR_T DbName;

 AppName = "Sample";
 DbName = "Basic";
 sts = EssCreateDatabaseEx(hCtx, AppName, DbName, ESS_DBTYPE_NORMAL, TRUE);

 return (sts);

}
```

関連トピック

- EssCreateDatabase
- EssCreateApplication
- EssCreateObject

EssCreateDrillThruURL

アクティブなデータベース・アウトライン内に、指定されたリンクと名前を使用してドリルスルー URL を作成します。

1904 ページの「ドリルスルー URL の制限」を参照してください。

構文

```
ESS_FUNC_M EssCreateDrillThruURL (
hCtx, pUrl
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pUrl	ESS_PDURLINFO_T	URL 定義。

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内にドリルスルー URL が作成されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESS_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EssSetActive()を使用して、指定したデータベースをアクティブなデータベースとして選択しておく必要があります。

例

```
    /* Sample Code for EssCreateDrillThruURL */

ESS_STS_T sts = ESS_STS_NOERR;
ESS_DURLINFO_T url;
ESS_USHORT_T usCountOfURLs, i;
ESS_PDURLINFO_T listOfURLs;
ESS_STR_T urlName = "";
ESS_PDURLINFO_T urlInfo;
ESS_STR_T fileName = "";
ESS_CHAR_T xmlString[XML_CHAR_MAX];

/* Valid case */

memset(&url, '\0', sizeof(ESS_DURLINFO_T));
fileName = "F:\\testarea\\mainapi\\sample1.xml";
GetFileContent(fileName, xmlString);

printf("\nValid case:\n");
 url.bIsLevel0 = ESS_TRUE;
 url.cpURLName = "Drill Through to EPMI";
 url.cpURLXml = xmlString;
```

```
 url.iURLXmlSize = (ESS_SHORT_T) strlen(xmlString)+1;
 url.iCountOfDrillRegions = 2;
sts = EssAlloc (hInst, sizeof(ESS_STR_T) * url.iCountOfDrillRegions,
&(url.cppDrillRegions));
 url.cppDrillRegions[0] = "@idesc(\"Qtr1\")";
 url.cppDrillRegions[1] = "@idesc(\"Qtr2\")";
sts = EssCreateDrillThruURL(hCtx, &url);
printf("EssCreateDrillThruURL sts: %ld\n",sts);
```

EssCreateExtGroup

外部ユーザー・ディレクトリにグループを作成します。

構文

```
ESS_FUNC_M EssCreateExtGroup (hCtx,
GroupName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupName	ESS_STR_T	作成するグループの名前(入力)。1904 ページの「グループ名の制限」を参照してください。

備考

指定したグループは、Shared Services 内に存在する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_CreateGroup (ESS_HCTX_T  hCtx)
{
  ESS_FUNC_M   sts = ESS_STS_NOERR;
  ESS_STR_T    GroupName;

  GroupName = "Group 1";

  sts = EssCreateExtGroup (hCtx, GroupName);

  return (sts);
}
```

関連トピック

- EssDeleteUser
- EssListUsers
- EssRenameUser
- EssSetPassword
- EssSetUser
- EssGetUserEx
- 207 ページの「ESS_USERINFOEX_T」

EssCreateExtUser

新規の外部認証ユーザーを作成します。

構文

```
ESS_FUNC_M EssCreateExtUser (hCtx, UserName, Password, SecurityProvider,
ProviderConnectionParameters);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	作成するユーザーの名前。1904 ページの「ユーザー名の制限」を参照してください。
Password	ESS_STR_T	新規ユーザーのセキュリティ・パスワード。1904 ページの「パスワードの制限」を参照してください。 **注：** Password パラメータは、Shared Services を変更することで重複可能です。このパラメータには空の文字列を使用できます。
SecurityProvider	ESS_PROTOCOL_T	外部認証メカニズムの名前。
ProviderConnectionParameters	ESS_CONNPARAM_T	外部認証メカニズムで使用されるパラメータ(ある場合)。

備考

- 指定したユーザーが存在していないことが必要です。
- EssSetUser 関数でユーザーのアクセス・レベルとその他のパラメータを設定できます。
- この関数を呼び出す前に、パスワードが正しく入力されたことをプログラムで確認する必要があります(たとえばパスワードを 2 度入力させるなど)。一度入力したパスワードは取得できません。ただし、EssSetPassword 関数を使用すればパスワードを変更できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_CreateUser (ESS_HCTX_T  hCtx)
{
  ESS_FUNC_M   sts = ESS_STS_NOERR;
  ESS_STR_T    UserName;
  ESS_STR_T    Password;
  ESS_PROTOCOL_T Protocol;
  ESS_CONN_PARAM ConnParam;

  UserName = "Jim Smith";
  Password = "Password";
  Protocol = CSS;
  ConnParam = NULL;

  sts = EssCreateExtUser (hCtx, UserName, Password, Protocol, ConnParam);

  return (sts);
}
```

関連トピック

- EssDeleteUser
- EssListUsers
- EssRenameUser
- EssSetPassword
- EssSetUser
- EssGetUserEx
- 207 ページの「ESS_USERINFOEX_T」

EssCreateFilter

新規フィルタを作成し、そのコンテンツの設定を開始します。

構文

```
ESS_FUNC_M  EssCreateFilter (
hCtx, AppName, DbName, FilterName, Active, Access
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。

パラメータ	データ型	説明
FilterName	ESS_STR_T	フィルタ名。1904 ページの「フィルタ名の制限」を参照してください。
Active	ESS_BOOL_T	フィルタのアクティブ・フラグ。TRUE の場合はフィルタがアクティブに設定され、TRUE でない場合は非アクティブに設定されます。
Access	ESS_ACCESS_T	デフォルトのフィルタ・アクセス・レベル。

備考

- フィルタが存在しない場合は、この呼出しによってフィルタが作成されます。
- フィルタがすでに存在する場合は、エラー・メッセージが戻されます。
- この呼出しの後に EssSetFilterRow を続けて呼び出して、フィルタのすべての行を設定する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateFilter (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FilterName;
 ESS_BOOL_T    Active;
 ESS_ACCESS_T   Access, AccessAry[3];
 ESS_STR_T    RowString[3];
 ESS_USHORT_T   ind;

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "NewFilter";
 Active    = ESS_TRUE;

  /***** Create Filter *****/
  sts = EssCreateFilter(hCtx, AppName, DbName,
     FilterName, Active, Access);
 if(!sts)
 {
  RowString[0] = "@IDESCENDANTS(Scenario)";
  RowString[1] = "@IDESCENDANTS(Product)";
  RowString[2] = "Qtr1, @IDESCENDANTS(\"Colas\")";

  AccessAry[0] = ESS_ACCESS_READ;
  AccessAry[1] = ESS_ACCESS_NONE;
  AccessAry[2] = ESS_ACCESS_WRITE;
  /***** Set Filter Rows *****/
```

```
   for(ind = 0; ind < 3; ind++)
   {
    sts = EssSetFilterRow(hCtx, RowString[ind],
         AccessAry[ind]);
    if(sts)
     printf("Cannot set Filter row %s\r\n",
         RowString[ind]);
   }
    sts = EssSetFilterRow(hCtx,
       "",ESS_ACCESS_NONE);
 }
 return (sts);
}
```

関連トピック

- EssGetFilter
- EssListFilters
- EssSetFilterRow
- EssSetFilter

EssCreateGroup

グループを新規作成します。

構文

```
ESS_FUNC_M EssCreateGroup (
hCtx, GroupName
);
```

パラメータ　データ型　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	作成するグループの名前。1904 ページの「グループ名の制限」を参照してください。

備考

- 指定したグループが存在していないことが必要です。
- EssSetGroup 関数でグループのアクセス・レベルを設定できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateGroup (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   GroupName;
 GroupName = "PowerUsers";

 sts = EssCreateGroup (hCtx, GroupName);

 return (sts);
}
```

関連トピック

- EssCreateExtGroup
- EssDeleteGroup
- EssListGroups
- EssRenameGroup
- EssSetGroup

EssCreateLocalContext

ローカル API 操作で使用するローカル API コンテキストを作成します。

構文

```
ESS_FUNC_M EssCreateLocalContext (
hInstance, UserName, Password, phLocalCtx
);
```

パラメータ	データ型	説明
hInstance;	ESS_HINST_T	API インスタンス・ハンドル。
UserName;	ESS_STR_T	現在使用されていません。NULL に設定する必要があります。
Password;	ESS_STR_T	現在使用されていません。NULL に設定する必要があります。
phLocalCtx;	ESS_PHCTX_T	Essbase ローカル・コンテキスト・ハンドルを受け取る変数のアドレス。

備考

- この関数はローカル API 操作(ローカル・ファイル/オブジェクト関数など)へのアクセスを必要とする場合に呼び出す必要があります。EssInit を呼び出した後に呼び出します。
- 関数はクライアント・アプリケーションにつき 1 回呼び出すのみです。コンテキスト・ハンドルは、ローカルの API 操作すべてに使用できます。

- アプリケーションがローカル・オブジェクトへのアクセスを終了した場合は、EssDeleteLocalContext を呼び出す必要があります。

戻り値

正常終了の場合は、有効なローカル・コンテキスト・ハンドルが phLocalCtx に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

EssGetLocalPath の例を参照してください。

関連トピック

- EssDeleteLocalContext
- EssInit

EssCreateLocationAlias

新規にロケーション別名を作成します。つまり、別名の文字列を、次の 5 個の文字列が順に並んだ形式へマッピングします。5 個の文字列とは: ホスト名、アプリケーション名、データベース名、ユーザー・ログイン名、ユーザー・パスワードです。

構文

```
ESS_FUNC_M EssCreateLocationAlias (
hCtx
,
pAlias
,
pHost
,
pApp
,
pDb
,
pName
,
pPassword
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
pAlias;	ESS_STR_T	ロケーション別名。
pHost;	ESS_STR_T	ターゲット・ホスト。
pApp;	ESS_STR_T	ターゲット・アプリケーション。

パラメータ	データ型	説明
pDb;	ESS_STR_T	ターゲット・データベース。
pName;	ESS_STR_T	ユーザー・ログイン名。
pPassword;	ESS_STR_T	ユーザー・パスワード。

戻り値

pAlias と同じ名前のロケーション別名がすでに存在する場合は、エラーが戻されます。

関連トピック

- EssDeleteLocationAlias
- EssGetLocationAliasList

EssCreateObject

サーバーまたはクライアントのオブジェクト・システムで、オブジェクトを新規作成します。

構文

```
ESS_FUNC_M EssCreateObject (
hCtx, ObjType, AppName, DbName, ObjName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	作成するオブジェクトの名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- 作成するオブジェクトが存在していないことが必要です。
- サーバー上で新規作成されたオブジェクトにはデータが含まれておらず、単なるプレースホルダとして機能し、他のユーザーによるオブジェクトの作成を防止します。作成されたオブジェクトを更新する場合、EssLockObject を使用してロックし、EssPutObject を使用して保存する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がオブジェクトを含むために指定されたアプリケーションまたはデータベースに対するアプリケーション・デザイン権限、またはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateObject (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    ObjName;
 ESS_OBJTYPE_T  ObjType;

 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Test";
 ObjType = ESS_OBJTYPE_OUTLINE;

 sts = EssCreateObject(hCtx, ObjType, AppName,
    DbName, ObjName);

 if(!sts)
  printf("The Object is created.\r\n");
 return (sts);
}
```

関連トピック

- EssCopyObject
- EssDeleteObject
- EssListObjects
- EssLockObject
- EssPutObject
- EssRenameObject

EssCreateStorageTypedApplication

ブロック(多次元)または集約のいずれかのデータ・ストレージ・モード・オプションで、新規アプリケーションを作成します。

構文

```
ESS_FUNC_M EssCreateStorageTypedApplication (
hCtx, AppName
```

```
    , StorageType);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。
StorageType	ESS_DATA_STORAGE_T	新しいアプリケーションのデータ・ストレージ・タイプ。 StorageType に対して有効な値は、次のとおりです: ● ESS_DEFAULT_DATA_STORAGE – ESS_MULTIDIM_DATA_STORAGE と同じです ● ESS_MULTIDIM_DATA_STORAGE – ブロック・ストレージ(多次元)で、デフォルトのストレージ・タイプです ● ESS_ASO_DATA_STORAGE – 集約ストレージ

備考

- 新しいアプリケーションは、非 Unicode モードで作成されます。
- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。アプリケーションまたはデータベースを作成する関数を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateASOApp (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 AppName = "Sample";
 sts = EssCreateStorageTypedApplication (hCtx, AppName, ESS_ASO_DATA_STORAGE);
 return(sts);
}
```

関連トピック

- EssCreateStorageTypedApplicationEx
- EssCreateApplication
- EssCreateDatabase

- EssCreateObject

EssCreateStorageTypedApplicationEx

データ・ストレージ・モードのオプション(ブロックまたは集約)およびアプリケーション・モード(Unicode または非 Unicode)を指定して、新規アプリケーションを作成します。

構文

```
ESS_FUNC_M EssCreateStorageTypedApplicationEx (
hCtx, AppName
, storageType, usAppType);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。
StorageType	ESS_DATA_STORAGE_T	新しいアプリケーションのデータ・ストレージ・タイプ。 StorageType に対して有効な値は、次のとおりです: ● ESS_DEFAULT_DATA_STORAGE - ESS_MULTIDIM_DATA_STORAGE と同じです ● ESS_MULTIDIM_DATA_STORAGE - ブロック・ストレージ(多次元)で、デフォルトのストレージ・タイプです ● ESS_ASO_DATA_STORAGE - 集約ストレージ
usAppType	ESS_USHORT_T	新規アプリケーションのアプリケーション・タイプ(Unicode または非 Unicode)。 usAppType の有効な値は次のとおりです: ● ESS_APP_UNICODE - 0x0003- Unicode アプリケーション。サーバーが Unicode モードでない場合、関数は失敗します。 ● ESS_APP_NONUNICODE - 0x0002- 非 Unicode アプリケーション。

備考

- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。アプリケーションまたはデータベースを作成する関数を呼び出した後で EssSetActive を呼び出し、EssRestructure などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションまたはデータベース)に対して実行されないようにします。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateASOApp (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 ESS_STR_T      AppName;
 AppName = "Sample";
 sts = EssCreateStorageTypedApplicationEx (hCtx, AppName,ESS_ASO_DATA_STORAGE,
ESS_APP_UNICODE);
 return(sts);
}
```

関連トピック

- EssCreateStorageTypedApplication
- EssCreateApplication
- EssCreateDatabase
- EssCreateObject

EssCreateUser

新規ユーザーを作成します。

構文

```
ESS_FUNC_M EssCreateUser (
hCtx, UserName, Password
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	作成するユーザーの名前。1904 ページの「ユーザー名の制限」を参照してください。
Password	ESS_STR_T	新規ユーザーのセキュリティ・パスワード。1904 ページの「パスワードの制限」を参照してください。

備考

- 指定したユーザーが存在していないことが必要です。
- EssSetUser 関数でユーザーのアクセス・レベルとその他のパラメータを設定できます。
- この関数を呼び出す前に、パスワードが正しく入力されたことをプログラムで確認する必要があります(パスワードを 2 回入力するなど)。一度入力したパ

スワードは取得できません。ただし、EssSetPassword 関数を使用すればパスワードを変更できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CreateUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   UserName;
 ESS_STR_T   Password;

 UserName = "Jim Smith";
 Password = "Password";

 sts = EssCreateUser (hCtx, UserName, Password);

 return (sts);
}
```

関連トピック

- EssDeleteUser
- EssListUsers
- EssRenameUser
- EssSetPassword
- EssSetUser

EssCreateVariable

代替変数を新規作成、または同一のサーバー値、アプリケーション値およびデータベース値を持つ変数名がすでに存在している場合には既存の代替変数を変更します。

構文

```
ESS_FUNC_M EssCreateVariable (
hCtx, pVariable
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pVariable	ESS_PVARIABLE_T	作成された代替変数の説明を含む構造体を指すポインタ。

備考

- 変数範囲はサーバー、アプリケーション、またはデータベースに適用できます。範囲は ESS_VARIABLE_T 構造体で制御されます。サーバー、アプリケーション、データベースがすべて指定されている場合は、代替変数は指定されたデータベースにのみ適用されます。サーバーとアプリケーションのみが指定されている場合、代替変数は指定されたアプリケーション内のすべてのデータベースに適用されます。サーバーのみが指定されている場合、代替変数は指定されたサーバーのすべてのアプリケーションとデータベースに適用されます。
- 新規変数が既存の変数と同じ名前および適用範囲で作成された場合、Essbase からエラー・メッセージは戻されずに新しい値で古い値が置換されます。
- 指定した 1 台のサーバー上では、同じ名前で適用範囲(アプリケーションとデータベース)の異なる複数の代替変数を作成できます。

戻り値

成功の場合、ゼロが戻されます。

例

```
    /*
** ESS_CreateVariable() creates a substitution variable using
** the API EssCreateVariable, and sets its value.
*/
ESS_FUNC_M
ESS_CreateVariable (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_VARIABLE_T Variable;
 printf("\n ****************************************");
 printf("\n **** An example of using EssCreateVariable");
 printf("\n ****************************************");
 /* Create Variable 'QuarterName' at the level of the server/App/Db */
 strcpy(Variable.VarName,  "QuarterName");
 strcpy(Variable.Server,  "Local");
 strcpy(Variable.AppName,  "Sample");
 strcpy(Variable.DbName,  "Basic");
 strcpy(Variable.VarValue, "Qtr1");
 sts = EssCreateVariable(hCtx, &Variable);
 if (sts == ESS_STS_NOERR)
  printf("\n Variable 'QuarterName' is created at the Server/App/Db level
  with value 'Qtr1'");

 /* Change Value of 'QuarterName' from Qtr1 to Qtr2 */
 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName,  "QuarterName");
  strcpy(Variable.Server,  "Local");
```

```
  strcpy(Variable.AppName,  "Sample");
  strcpy(Variable.DbName,  "Basic");
  strcpy(Variable.VarValue, "Qtr2");
  sts = EssCreateVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
   printf("\n Variable 'QuarterName' at the Server/App/Db level is updated
   to value 'Qtr2'");
 }
 /* Create Variable 'MarketName' at the level of the Server/App */
 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName,  "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName,  "Sample");
  strcpy(Variable.DbName,  "");
  strcpy(Variable.VarValue, "East");
  sts = EssCreateVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
   printf("\n Variable 'MarketName' is created at the Server/App level");
 }
 /* Create Variable "MarketName' at the level of the Server */
 /* This shows that you can have the same variable name at different levels*/

 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName,  "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName,  "");
  strcpy(Variable.DbName,  "");
  strcpy(Variable.VarValue, "Market");
  sts = EssCreateVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
   printf("\n Variable 'MarketName' is created at the Server level");
 }
 if (sts == ESS_STS_NOERR)
  printf("\n --> No Errors in EssCreateVariable\n\n\n");
 else
  printf("\n --> Error in EssCreateVariable number: %d\n\n\n", sts);

 return (sts);
} /* end ESS_CreateVariable */
```

出力

```
    ****************************************
**** An example of using EssCreateVariable
****************************************
Variable 'QuarterName' is created at the Server/App/Db level with value 'Qtr1'
Variable 'QuarterName' at the Server/App/Db level is updated to value 'Qtr2'
Variable 'MarketName' is created at the Server/App level
Variable 'MarketName' is created at the Server level
--> No Errors in EssCreateVariable
```

関連トピック

- 209 ページの「ESS_VARIABLE_T」
- EssDeleteVariable
- EssGetVariable
- EssListVariables

EssDefaultCalc

アクティブ・データベースのデフォルト計算を実行します。

構文

```
ESS_FUNC_M EssDefaultCalc (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数が正常終了して計算が開始すると、この呼出しから戻った後に非同期プロセスとしてサーバー上で続行されます。呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- デフォルト計算スクリプトを取得および設定するには、関数 EssGetDefaultCalc、EssSetDefaultCalc および EssSetDefaultCalcFile を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_CalcDefault  (ESS_HCTX_T     hCtx)
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 ESS_PROCSTATE_T  pState;

 sts = EssDefaultCalc(hCtx);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
      ESS_STATE_DONE))
```

```
      sts = EssGetProcessState (hCtx,
         &pState);
  }

 return (sts);
}
```

関連トピック

- EssBeginCalc
- EssCalc
- EssGetDefaultCalc
- EssSetDefaultCalc
- EssSetDefaultCalcFile

EssDeleteAllSplFiles

データベースのすべてのトリガー・ログ・ファイルを削除します。

構文

```
ESS_FUNC_M EssDeleteAllSplFiles (
hCtx, AppName, DbName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssDisplayTriggers
- EssListSpoolFiles
- EssGetSpoolFile
- EssDeleteSplFile
- EssMdxTrig

EssDeleteApplication

クライアント上またはサーバー上で、既存のアプリケーションを削除します。アプリケーションがサーバー上で実行されている場合、最初に停止されます。

構文

```
ESS_FUNC_M EssDeleteApplication (
hCtx, AppName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	削除するアプリケーションの名前。

備考

クライアント・アプリケーションを削除すると、ローカルのアプリケーションのディレクトリとコンテンツが削除されます。アプリケーションで保管されているすべてのオブジェクトも、すべてのデータベースも含めて削除されます。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteApp (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 AppName =  "Sample";

 sts = EssDeleteApplication (hCtx, AppName);

 return(sts);
}
```

関連トピック

- EssDeleteDatabase
- EssDeleteObject

EssDeleteDatabase

クライアントまたはサーバー上で、アプリケーションから既存のデータベースを削除します。データベースがサーバー上で実行されている場合、最初に停止されます。

構文

```
ESS_FUNC_M EssDeleteDatabase (
hCtx, AppName, DbName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	データベースを含むアプリケーション名。
DbName	ESS_STR_T	削除するデータベースの名前。

備考

- クライアント・データベースを削除すると、ローカルのデータベースのディレクトリとコンテンツも削除されます。
- サーバー・データベースを削除すると、そのデータベースに関連付けられているすべてのオブジェクトも削除されます。

戻り値

なし。

アクセス

サーバー・データベースの場合は、呼出し元がデータベースの作成/削除/編集権限(ESS_PRIV_DBCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteDb (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 AppName = "Sample";
 DbName  = "Basic";

 /* IF the current active is the same as the
  * unload db, ClearActive first */
   sts = EssClearActive(hCtx);

 /* ELSE */
   sts = EssDeleteDatabase(hCtx, AppName,
      DbName);
 return (sts);
}
```

関連トピック

- EssDeleteApplication
- EssDeleteObject

EssDeleteDrillThruURL

アクティブなデータベース・アウトライン内で、指定された URL 名のドリルスルー URL を削除します。

構文

```
ESS_FUNC_M EssDeleteDrillThruURL (
hCtx, URLName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
URLName	ESS_STR_T	ドリルスルー URL 名。

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内の指定されたドリルスルー URL が削除されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESS_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EssSetActive を使用して、指定したデータベースをアクティブ・データベースとして選択しておく必要があります。

例

```
    ESS_STS_T sts = ESS_STS_NOERR;

sts = EssDeleteDrillThruURL(hCtx, "Drill Through to EPMI");
printf("EssDeleteDrillThruURL sts: %ld\n",sts);
```

EssDeleteFilter

既存のフィルタを削除します。

構文

```
ESS_FUNC_M EssDeleteFilter (
hCtx, AppName, DbName, FilterName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
FilterName	ESS_STR_T	フィルタ名。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteFilter (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FilterName;

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "Test";

 sts = EssDeleteFilter(hCtx, AppName, DbName,
    FilterName);
 return (sts);
}
```

関連トピック

- EssCopyFilter
- EssListFilters
- EssRenameFilter
- EssSetFilter

EssDeleteFromGroup

グループ・メンバーのリストからユーザーを削除します。

構文

```
ESS_FUNC_M EssDeleteFromGroup (
hCtx, GroupName, UserName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	グループ名。
UserName	ESS_STR_T	グループ・リストから削除するユーザー名。

備考

指定したグループのメンバー・リストから指定したユーザーを削除するのみでなく、この関数は削除するユーザーの関連グループのリストから指定したグループも削除します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RemoveUser (ESS_HCTX_T   hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   GroupName;
 ESS_STR_T   UserName;
 GroupName = "PowerUsers";
 UserName  = "Jim Smith";

 sts = EssDeleteFromGroup (hCtx, GroupName, UserName);

 return (sts);
}
```

関連トピック

- EssDeleteFromGroupEx
- EssAddToGroup
- EssGetGroupList
- EssListGroups
- EssSetGroupList

EssDeleteFromGroupEx

グループからユーザーを削除します。EssDeleteFromGroup に似ていますが、ユーザー・ディレクトリの指定、または GroupId や UserId の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssDeleteFromGroupEx (
hCtx
,
GroupId
,
bIsGroupId
,
UserId
,
bUsingIdentity
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupId	ESS_STR_T	グループ名または ID（入力)。groupname@provider または一意の ID 属性として指定できます。
bIsGroupId	ESS_BOOL_T	入力。GroupId が名前か ID かを示します。TRUE の場合、GroupId は ID です。
UserId	ESS_STR_T	グループから削除するユーザー名(入力)。username@provider または一意の ID 属性として指定できます。
bUsingIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。

備考

指定したグループのメンバー・リストから指定したユーザーを削除するのみでなく、この関数は削除するユーザーの関連グループのリストから指定したグループも削除します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
void DisplayUserList(ESS_USHORT_T count, ESS_PSTR_T UserList)
{
    ESS_USHORT_T i;

    for (i = 0; i < count; i++)
    {
        if (UserList [i])
            printf ("%s\n", UserList[i]);
    }
```

```
}

ESS_FUNC_M ESS_RemoveUser (ESS_HCTX_T  hCtx)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_STR_T groupId, userId;
 ESS_BOOL_T bGroupId, bUserId;
 ESS_BOOL_T bisIdentity;
 ESS_USHORT_T type;
 ESS_USHORT_T count;
 ESS_BOOL_T bUsingIdentity;
 ESS_PSTR_T pUserList;

 groupId = "IDRegularGroup@ldap";
 userId = "IDUser8@ldap";
 bGroupId = ESS_FALSE;
 bUserId = ESS_TRUE;
 sts = EssDeleteFromGroupEx (hCtx, groupId, bGroupId, userId, bUserId);
 printf("EssDeleteFromGroupEx sts: %ld\n", sts);
 if(!sts)
 {
  sts = EssGetGroupListEx(hCtx, groupId, bisIdentity, type, &count, &bUsingIdentity,
&pUserList);
   printf("EssGetGroupListEx sts: %ld\n", sts);
   if(!sts)
   {
    if(pUserList)
    {
     printf ("\n---User/Group list for %s:\n", groupId);
     DisplayUserList(count, pUserList);
    }
    else
     printf ("\tUser list is empty\n");
   }
 }


 return (sts);
}
```

関連トピック

- EssAddToGroupEx
- EssGetGroupListEx
- EssListGroupsInfoEx

EssDeleteGroup

既存のグループを削除します。

構文

```
      ESS_FUNC_M EssDeleteGroup (
      hCtx, GroupName
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	削除するグループの名前。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteGroup (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   GroupName;
 GroupName = "PowerUsers";

 sts = EssDeleteGroup (hCtx, GroupName);

 return (sts);
}
```

関連トピック

- EssDeleteGroupEx
- EssCreateGroup
- EssListGroups
- EssRenameGroup

EssDeleteGroupEx

既存のグループを削除します。EssDeleteGroup に似ていますが、ユーザー・ディレクトリの指定、または GroupID の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssDeleteGroupEx (
hCtx
,
GroupId
,
bisIdentity
 );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupId	ESS_STR_T	削除するグループ名(入力)。groupname@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。GroupId が名前か ID かを示します。TRUE の場合、GroupId は ID です。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
ESS_FUNC_M ESS_DeleteGroupEx (ESS_HCTX_T hCtx)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T groupId;
 ESS_BOOL_T bIsIdentity;

 groupId = "IDTempGroup1@ldap";
 bIsIdentity = ESS_FALSE;
 sts = EssDeleteGroupEx(hCtx, groupId, bIsIdentity);
 printf("EssDeleteGroupEx sts: %ld\n", sts);

 return (sts);
}
```

関連トピック

- EssDeleteUserEx

EssDeleteLocalContext

以前に EssCreateLocalContext で作成されたローカル・コンテキストをリリースします。

構文

```
ESS_FUNC_M EssDeleteLocalContext (
hLocalCtx
);
```

パラメータ　データ型　　説明

hLocalCtx　ESS_HCTX_T API ローカル・コンテキスト・ハンドル。

備考

この関数はローカル・コンテキストに対してのみ使用してください。ログイン・コンテキストには、EssLogout 関数を使用します。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

EssGetLocalPath の例を参照してください。

関連トピック

- EssCreateLocalContext
- EssLogout
- EssTerm

EssDeleteLocationAlias

既存のロケーション別名を削除します。

構文

```
ESS_FUNC_M EssDeleteLocationAlias (
hCtx
,
pAlias
);
```

パラメータ　データ型　　説明

hCtx;　　ESS_HCTX_T API コンテキスト・ハンドル。

pAlias;　ESS_STR_T　ロケーション別名。

戻り値

pAlias という名前がついたロケーション別名が見つからない場合は、エラーが戻されます。

関連トピック

- EssCreateLocationAlias
- EssGetLocationAliasList

EssDeleteLogFile

サーバー上のアプリケーション・ログ・ファイルまたは Essbase サーバー・ログ・ファイル(essbase.log)を削除します。

構文

```
ESS_FUNC_M EssDeleteLogFile (
hCtx, AppName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名または NULL。NULL の場合は、この関数によって Essbase サーバー・ログ・ファイル(essbase.log)が削除されます。

備考

- メッセージ・ログを表示するには、EssGetLogFile を使用します。
- essbase.log の場所については、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元は指定されたアプリケーションに対するアプリケーション・デザイナ権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteLogFile (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 AppName = "Sample";

 sts = EssDeleteLogFile (hCtx, AppName);
 return(sts);
}
EssDeleteLogFile ("")  //Deletes Agent log file.
```

関連トピック

- EssGetLogFile
- EssLogSize
- EssWriteToLogFile

EssDeleteObject

既存のオブジェクトを削除します。

構文

```
ESS_FUNC_M EssDeleteObject (
hCtx, ObjType, AppName, DbName, ObjName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	削除するオブジェクトの名前。

備考

- オブジェクトを削除するには、そのオブジェクトがロックされていないことが必要です。
- アウトライン・オブジェクトは削除できません。関連付けられているアウトラインを含めてデータベースを削除するには、EssDeleteDatabase 関数を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、オブジェクトが含まれる指定されたアプリケーションまたはデータベースに対してアプリケーション・デザイン権限またはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteObject (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     ObjName;
 ESS_OBJTYPE_T   ObjType;
 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Test";
 ObjType = ESS_OBJTYPE_TEXT;
```

```
 sts = EssDeleteObject(hCtx, ObjType, AppName,
    DbName, ObjName);
 return (sts);
}
```

関連トピック

- EssCreateObject
- EssListObjects

EssDeleteSplFile

データベースについての特定のトリガー・ログ・ファイルを削除します。

構文

```
ESS_FUNC_M EssDeleteSplFile (
hCtx, AppName, DbName, SplName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
SplName	ESS_STR_T	削除するスプール・ファイルの名前。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssDisplayTriggers
- EssListSpoolFiles
- EssGetSpoolFile
- EssDeleteAllSplFiles
- EssMdxTrig

EssDeleteUser

ユーザーを削除します。

構文

```
ESS_FUNC_M EssDeleteUser (
hCtx, UserName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	削除するユーザー名。

備考

呼出し元は、呼び出したユーザーと管理者のいずれも削除できません。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_DeleteUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   UserName;
 UserName = "Jim Smith";

 sts = EssDeleteUser (hCtx, UserName);

 return (sts);
}
```

関連トピック

- EssDeleteUserEx
- EssCreateExtUser
- EssCreateUser
- EssListUsers
- EssRenameUser

EssDeleteUserEx

ユーザーを削除します。EssDeleteUser に似ていますが、ユーザー・ディレクトリの指定、または一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssDeleteUserEx (
hCtx
,
UserId
,
bIsIdentity
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	コンテキスト・ハンドル(入力)。
UserId	ESS_STR_T	削除するユーザー名(入力)。username@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。

備考

呼出し元は、呼び出したユーザーと管理者のいずれも削除できません。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
ESS_FUNC_M ESS_DeleteUserEx (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M sts = ESS_STS_NOERR;
 ESS_STR_T userId;
 ESS_BOOL_T bIsIdentity;

 userId = "IDUser3@ldap";
 bIsIdentity = ESS_FALSE;
 sts = EssDeleteUserEx(hCtx, userId, bIsIdentity);
 printf("EssDeleteUserEx sts: %ld\n", sts);

 return (sts);
}
```

関連トピック

- EssDeleteGroupEx

EssDeleteVariable

代替変数を削除します。

構文

```
ESS_FUNC_M EssDeleteVariable (
hCtx, pVariable
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API へのコンテキスト・ハンドル。
pVariable	209 ページの「ESS_VARIABLE_T」	削除される代替変数の説明を含む構造体を指すポインタ。

戻り値

成功の場合、ゼロが戻されます。

例

```
    /*
** ESS_DeleteVariable() deletes a substitution variable using
** the API EssDeleteVariable.
*/
ESS_FUNC_M
ESS_DeleteVariable (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_VARIABLE_T  Variable;
 ESS_PVARIABLE_T pVariables;
 ESS_ULONG_T   ulCount, i;
 printf("\n ****************************************");
 printf("\n **** An example of using EssDeleteVariable");
 printf("\n ****************************************");

 strcpy(Variable.VarName, "QuarterName");
 strcpy(Variable.Server,  "Local");
 strcpy(Variable.AppName, "Sample");
 strcpy(Variable.DbName,  "Basic");
 sts = EssDeleteVariable(hCtx, &Variable);
 if (sts == ESS_STS_NOERR)
  printf("\n Variable 'QuarterName' at the Server/App/Db level is deleted");
 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName, "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName, "Sample");
  strcpy(Variable.DbName,  "");
  sts = EssDeleteVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
    printf("\n Variable 'MarketName' at the Server/App level is deleted");
 }

 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName, "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName, "");
  strcpy(Variable.DbName,  "");
  sts = EssDeleteVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
    printf("\n Variable 'MarketName' at the Server level is deleted");
 }
```

```
/*********************************************************/
/* List the variables at the level of the Server/App/Db- */
/* We should not have any               */
/*********************************************************/
if (sts == ESS_STS_NOERR)
{
 strcpy(Variable.Server,  "local");
 strcpy(Variable.AppName, "Sample");
 strcpy(Variable.DbName,  "Basic");
 sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
 if (sts == ESS_STS_NOERR)
 {
  printf("\n--- Number of Substitution Variables at the Server, App and Db
   level is: %ld\n", ulCount);
  for (i = 0; i < ulCount; i++)
  {
   printf("Variable name   : %s\n",  pVariables[i].VarName);
   printf("Server name    : %s\n",  pVariables[i].Server);
   printf("Application name : %s\n",  pVariables[i].AppName);
   printf("Database name   : %s\n",  pVariables[i].DbName);
   printf("Variable value  : %s\n\n", pVariables[i].VarValue);
  }
 }
}
/*********************************************************/
/* List the variables - at the level of the App     */
/*********************************************************/

if (sts == ESS_STS_NOERR)
{
 strcpy(Variable.Server,  "local");
 strcpy(Variable.AppName, "Sample");
 strcpy(Variable.DbName,  "");
 sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
 if (sts == ESS_STS_NOERR)
 {
  printf("\n--- Number of Substitution Variables at the Server and App
   level is: %ld\n", ulCount);
  for (i = 0; i < ulCount; i++)
  {
   printf("Variable name   : %s\n",  pVariables[i].VarName);
   printf("Server name    : %s\n",  pVariables[i].Server);
   printf("Application name : %s\n",  pVariables[i].AppName);
   printf("Database name   : %s\n",  pVariables[i].DbName);
   printf("Variable value  : %s\n\n", pVariables[i].VarValue);
  }
 }
}
/*********************************************************/
/* List variables at the server level          */
/*********************************************************/
if (sts == ESS_STS_NOERR)
{
 strcpy(Variable.Server,  "local");
 strcpy(Variable.AppName, "");
 strcpy(Variable.DbName,  "");
 sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
```

```
  if (sts == ESS_STS_NOERR)
  {
   printf("\n--- Number of Substitution Variables at the Server level is:
    %ld\n", ulCount);
   for (i = 0; i < ulCount; i++)
   {
    printf("Variable name   : %s\n",  pVariables[i].VarName);
    printf("Server name    : %s\n",  pVariables[i].Server);
    printf("Application name : %s\n",  pVariables[i].AppName);
    printf("Database name   : %s\n",  pVariables[i].DbName);
    printf("Variable value  : %s\n\n", pVariables[i].VarValue);
   }
  }
 }
 if (sts == ESS_STS_NOERR)
  printf("\n --> No Errors in EssDeleteVariable\n\n\n");
 else
  printf("\n --> Error in EssDeleteVariable number: %d\n\n\n", sts);
 return (sts);
} /* end ESS_DeleteVariable */
```

出力

```
    ****************************************
**** An example of using EssDeleteVariable
****************************************
Variable 'QuarterName' at the Server/App/Db level is deleted
Variable 'MarketName' at the Server/App level is deleted
Variable 'MarketName' at the Server level is deleted
--- Number of Substitution Variables at the Server, App and Db level is: 0
--- Number of Substitution Variables at the Server and App level is: 0
--- Number of Substitution Variables at the Server level is: 0
 --> No Errors in EssDeleteVariable
```

関連トピック

- 209 ページの「ESS_VARIABLE_T」
- EssCreateVariable
- EssGetVariable
- EssListVariables

EssDisplayAlias

アクティブ・データベース内の別名テーブルのコンテンツをダンプします。

構文

```
ESS_FUNC_M EssDisplayAlias (
hCtx, AliasName, pCount, ppAliases
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AliasName	ESS_STR_T	別名テーブル名。
pCount	ESS_PUSHORT_T	別名のカウントを受け取る変数のアドレス。
ppAliases	156 ページの「ESS_MBRALT_T」	メンバーの別名テーブルを受け取るポインタのアドレス。

備考

ppAliases に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_DisplayAlias (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_USHORT_T  Count;
 ESS_USHORT_T  ind;
 ESS_PMBRALT_T Altlist;
 ESS_STR_T   AltName;

 AltName = "TestAlias";

 sts = EssDisplayAlias (hCtx, AltName, &Count, &Altlist);
 if (Count)
 {
  printf ("\r\n-----Alias Contents-----\r\n\r\n");

  for (ind = 0; ind < Count; ind++)
  {
    printf ("%s==>%s\r\n",
        Altlist [ind].MbrName, Altlist [ind].AltName);
  }
  printf ("\r\n");
 }

 return (sts);
}
```

関連トピック

- EssListAliases

EssDisplayTriggers

データベースのすべてのトリガーをリストします。

構文

```
ESS_FUNC_M EssDisplayTriggers (
hCtx, AppName, DbName, pszTrg, pCount, ppTriggerList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名
pszTrg	ESS_STR_T	情報を戻す特定のトリガー名。pszTrg が""(空の文字列)の場合、指定されたデータベース内のすべてのトリガーが戻されます。
pCount	ESS_PUSHORT_T	情報を戻すトリガーの数を受け取る変数のアドレス。
ppTriggerList	ESS_PPTRIGGERINFO_T	トリガー情報構造体の割り当てられた配列を受け取るポインタのアドレス。トリガー情報構造体には、各トリガー名、トリガー定義、トリガーが使用可能かどうかを示すブール値のフィールドが含まれます。

備考

ppTriggerList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、データベース内のトリガーのカウントが pCount に、トリガー名の配列が ppTriggerList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssListSpoolFiles
- EssGetSpoolFile
- EssDeleteAllSplFiles
- EssDeleteSplFile
- EssMdxTrig

EssDTAPIClose

ドリルスルー・セッションを終了します。

構文

```
ESS_FUNC_M EssDTAPIClose (
pDTAPIInst
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル

備考

- この関数はドリルスルー・セッションを閉じますが、メモリーは解放しません。
- EssDTAPIExit はドリルスルー・セッションを閉じ、メモリーを解放します。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIConnect

指定したドリルスルー・インスタンス・ハンドルで Essbase Studio への接続を確立します。

構文

```
ESS_FUNC_M EssDTAPIConnect (
pDTAPIInst
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_DTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIExecuteReport

レポート定義構造体の配列に対するインデックスにより識別される、ドリルスルー・レポートを実行します。

構文

```
ESS_FUNC_M EssDTAPIExecuteReport (
pDTAPIInst
,
index
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル
index;	ESS_ULONG_T	実行するレポートのインデックス

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection

- EssDTAPISetInfo

EssDTAPIExit

ドリルスルー・セッションを終了し、指定したドリルスルー・インスタンス・ハンドルのメモリーを解放します。

構文

```
ESS_FUNC_M EssDTAPIExit (
pDTAPIInst
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。

備考

- この関数はドリルスルー・セッションを閉じ、メモリーを解放します。
- EssDTAPIClose はドリルスルー・セッションを閉じますが、メモリーは解放しません。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIGetColumns

指定したドリルスルー・インスタンス・ハンドルに対するレポート・ヘッダー情報構造体の配列を取得します。

構文

```
ESS_FUNC_M EssDTAPIGetColumns (
pDTAPIInst
,
```

```
ppCol
,
pulCount
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_DTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。
ppCol;	142 ページの「ESS_DTAPIHEADER_T」	指定した列のレポート・ヘッダー構造体の配列。
pulCount;	ESS_PULONG_T	ppCol レポート・ヘッダー情報配列に含まれるデータ・ブロックの数。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIGetData

指定したドリルスルー・インスタンス・ハンドルに対するレポート・データ構造体の配列を取得します。

構文

```
ESS_FUNC_M EssDTAPIGetData (
pDTAPIInst
,
ppData
,
pulRowCount
,
pulColCount
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_DTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。
ppData;	141 ページの「ESS_DTAPIDATA_T」	指定したデータ・セルのレポート・データ構造体の配列。
pulRowCount;	ESS_PULONG_T	ppData レポート・データ配列に含まれるデータ・ブロックの行数。
pulColCount;	ESS_PULONG_T	ppData レポート・データ配列に含まれるデータ・ブロックの列数。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIGetError

エラー・ステータスとメッセージを取得します。

構文

```
ESS_FUNC_M EssDTAPIGetError (
pDTAPIInst
,
ppData
,
pMsg
,
ulCount
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_DTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。
ppData;	ESS_STS_T	エラー・ステータス。
pMsg;	ESS_PSTR_T	エラー・メッセージ。

パラメータ	データ型	説明
ulCount;	ESS_ULONG_T	エラー・メッセージ・バッファのサイズ。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIGetInfo

指定されたドリルスルー・ハンドルのドリルスルー接続情報を取得します。

構文

```
ESS_FUNC_M EssDTAPIGetInfo (
pDTAPIInst
,
pDTInfo
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル
pDTInfo;	142 ページの「ESS_DTAPIINFO_T」	指定したデータ・セル範囲の接続情報構造体へのポインタ

備考

- この関数を呼び出す前に、ESS_DTAPIINFO_T のメモリーを割り当てます。
- sPassword は、pDTInfo に戻されません。つまり、ESS_DTAPIINFO_T の sPassword フィールドは戻されません。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect

- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIGetReports

指定したドリルスルー・インスタンス・ハンドルのレポート・リストを戻します。

構文

```
ESS_FUNC_M EssDTAPIGetReports (
pDTAPIInst
,
ppReports
,
pulCount
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_DTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。
ppReports;	143 ページの「ESS_DTAPIREPORT_T」	レポート定義構造体の配列へのポインタ。
pulCount;	ESS_PULONG_T	ppReports のデータ・ブロックの数。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIInit
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPIInit

ドリルスルー・セッションを開始し、ドリルスルー・インスタンス・ハンドルを戻します。

構文

```
ESS_FUNC_M EssDTAPIInit (
pDTAPIInit
,
pDTAPIInst
);
```

パラメータ	データ型	説明
pDTAPIInit;	ESS_PDTAPIINIT_T	現在使用されていません。NULL に設定されます。
ppDTAPIInst;	ESS_PDTAPIHINST_T	ドリルスルー初期化構造体へのポインタ。

備考

- この関数は、ppDTAPIHInst を初期化します。
- 入力用の pDTAPIInit は、現在使用されていないため、NULL に設定されます。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPISetConnection
- EssDTAPISetInfo

EssDTAPISetConnection

接続情報文字列と拡張メンバー・コメント文字列を使用して、ドリルスルー・ハンドルを初期化し、ドリルスルー・ウィザードを起動します。

構文

```
ESS_FUNC_M EssDTAPISetConnection (
pDTAPIInst
,
pEMC
,
```

```
ulCount
,
pDTInfo
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	ドリルスルー初期化構造体へのポインタ。
pEMC;	ESS_PSTR_T	拡張メンバー・コメント。
ulCount;	ESS_ULONG_T	拡張メンバー・コメントのブロック数。
pDTInfo;	ESS_PSTR_T	接続情報。

備考

ドリルスルー・インスタンス・ハンドルを初期化するには、EssGDTRequestDrillThrough を使用します。理由:

- 現在、EssDTAPISetConnection では pDTAPIInst を初期化できません。
- セキュリティ上の問題があるため、Essbase サーバーからは pConnection(接続情報文字列)と pEMC(拡張メンバー・コメント文字列)を取得できません。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetInfo

EssDTAPISetInfo

指定したドリルスルー・ハンドルのドリルスルー接続情報を設定します。

構文

```
ESS_FUNC_M EssDTAPISetInfo (
pDTAPIInst
,
pDTInfo
);
```

パラメータ	データ型	説明
pDTAPIInst;	ESS_PDTAPIHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。
pDTInfo;	142 ページの「ESS_DTAPIINFO_T」	指定したデータ・セル範囲の接続情報構造体へのポインタ。

備考

ESS_DTAPIINFO_T の uInputOption フィールドは無視されます。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTAPIClose
- EssDTAPIConnect
- EssDTAPIExecuteReport
- EssDTAPIExit
- EssDTAPIGetColumns
- EssDTAPIGetData
- EssDTAPIGetError
- EssDTAPIGetInfo
- EssDTAPIGetReports
- EssDTAPIInit
- EssDTAPISetConnection

EssDTClose

指定したドリルスルー・インスタンス・ハンドルのドリルスルー・セッションを終了します。

構文

```
ESS_FUNC_M EssDTClose (
pDTInst
);
```

パラメータ	データ型	説明
pDTInst;	ESS_PDTHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。

備考

- この関数はドリルスルー・セッションを閉じますが、メモリーは解放しません。
- EssDTExit はドリルスルー・セッションを閉じ、メモリーを解放します。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTExit
- EssDTGetData
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTInit
- EssDTListReports
- EssDTOpen

EssDTExit

ドリルスルー・セッションを終了し、指定したドリルスルー・インスタンス・ハンドルのメモリーを解放します。

構文

```
ESS_FUNC_M EssDTExit (
pDTInst
);
```

パラメータ	データ型	説明
pDTInst;	ESS_PDTHINST_T	指定したデータ・セル範囲の初期化済ドリルスルー・インスタンス・ハンドル。

備考

- この関数はドリルスルー・セッションを閉じ、メモリーを解放します。
- EssDTClose はドリルスルー・セッションを閉じますが、メモリーは解放しません。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTGetData
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTInit
- EssDTListReports
- EssDTOpen

EssDTGetData

指定したドリルスルー・インスタンス・ハンドルに対するレポート・データの配列を取得します。

構文

```
ESS_FUNC_M EssDTGetData (
pDTInst
,
pData
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst;	ESS_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
pData;	143 ページの「ESS_DTDATA_T」	指定したデータ・セルに対するレポート・データ構造体の配列。
pulCount;	ESS_PULONG_T	pData ヘッダー情報の配列に含まれるデータ・ブロックの数。

備考

pulCount が 0(ゼロ)になるまで、この関数を呼び出します。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTExit
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTInit
- EssDTListReports
- EssDTOpen

EssDTGetHeader

指定したドリルスルー・インスタンス・ハンドルに対するレポート・ヘッダー構造体の配列を取得します。

構文

```
ESS_FUNC_M EssDTGetHeader (
pDTInst
,
pBuffer
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst;	ESS_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
pBuffer;	143 ページの「ESS_DTBUFFER_T」	指定した列のレポート・ヘッダー構造体の配列。
pulCount;	ESS_PULONG_T	pBuffer レポート・ヘッダー情報の配列に含まれるデータ・ブロックの数。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTExit
- EssDTGetData
- EssDTGetHeaderInfo
- EssDTInit
- EssDTListReports
- EssDTOpen

EssDTGetHeaderInfo

指定したドリルスルー・インスタンス・ハンドルのレポート・データ・ヘッダー情報を取得します。

構文

```
ESS_FUNC_M EssDTGetHeaderInfo (
pDTInst
,
ppHeader
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst	ESS_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
ppHeader	144 ページの「ESS_DTHEADER_T」	指定したドリルスルー・インスタンス・ハンドルのヘッダー情報構造体の配列。
pulCount	ESS_PULONG_T	ppHeader ヘッダー情報配列のブロック数。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTExit
- EssDTGetData

- EssDTGetHeader
- EssDTInit
- EssDTListReports
- EssDTOpen
- EssDTOpen

EssDTInit

ドリルスルー・セッションを開始し、ドリルスルー・インスタンス・ハンドルを戻します。

構文

```
ESS_FUNC_M EssDTInit (
pInit
,
pDTInst
);
```

パラメータ	データ型	説明
pDTInit;	ESS_PDTINIT_T	（現在使用されていません。NULL に設定されます。）
ppDTInst;	ESS_PDTHINST_T	ドリルスルー初期化構造体へのポインタ。

備考

- この関数は、ppDTHInst を初期化します。
- 入力用の pDTInit は、現在使用されていないため、NULL に設定されます。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTExit
- EssDTGetData
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTListReports
- EssDTOpen

EssDTListReports

指定したドリルスルー・インスタンス・ハンドルのレポート名のリストを戻します。

構文

```
ESS_FUNC_M EssDTListReports (
```

```
pDTInst, pBuffer, pulCount
);
```

パラメータ	データ型	説明
pDTInst	ESS_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
pBuffer	ESS_PSTR_T	指定したドリルスルー・インスタンス・ハンドルのレポート名の配列。
pulCount	ESS_PULONG_T	pBuffer ヘッダー情報の配列に含まれるブロックのカウント。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTGetData
- EssDTGetHeader
- EssDTInit
- EssDTOpen

EssDTOpen

接続情報文字列と拡張メンバー・コメント文字列を使用して、ドリルスルー・ハンドルを初期化し、ドリルスルー・ウィザードを起動します。

構文

```
ESS_FUNC_M EssDTOpen (
pDTInst
,
pEMC
,
ulCount
,
pConnection
);
```

パラメータ	データ型	説明
pDTInst;	ESS_PDTHINST_T	ドリルスルー初期化構造体へのポインタ。
pEMC;	ESS_PSTR_T	拡張メンバー・コメント。
ulCount;	ESS_ULONG_T	拡張メンバー・コメントのブロック数。
pConnection;	ESS_PSTR_T	接続情報。

備考

- この関数は、pDTInst を初期化します。

- アウトラインを指定してデータ・セルを選択すると、Essbase から pConnection(接続情報文字列)と pEMC(拡張メンバー・コメント文字列)を取得できます。

関連トピック

- 第 6 章「C のメイン API の使用」
- 120 ページの「C のメイン API の構造体」
- EssDTClose
- EssDTExit
- EssDTGetData
- EssDTGetHeader
- EssDTGetHeaderInfo
- EssDTInit
- EssDTListReports

EssDumpPerfStats

パフォーマンス統計テーブルを文字配列としてダンプします。

構文

```
ESS_FUNC_M EssDumpPerfStats (hCtx, pStatBuf, [thdSN])
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
pStatBuf;	ESS_STR_T	パフォーマンス統計テーブルのダンプ先アドレスへのポインタ(入力)。
thdSN;	ESS_INT_T	オプション。統計をダンプするスレッド・シリアル番号(入力)。デフォルトは 0 です(すべてのスレッドがダンプされます)。

備考

この関数を呼び出す前に EssGetStatBufSize を呼び出して、pStatBuf で指定されたアドレスにパフォーマンス統計テーブル用に割り当てるためのメモリー量を確認します。

戻り値

- 成功の場合、この関数は
  - 0 を戻します。
  - パフォーマンス統計テーブルが、pStatBuf で指定されたアドレスから始まる文字配列としてダンプされます。
- この関数の呼出し元は、pStatBuf で指定されたアドレスのメモリーの割当てと解放を行います。
- パフォーマンス統計テーブルの詳細は、『Oracle Essbase テクニカル・リファレンス』を参照してください。

アクセス

この関数を使用するには、スーパーバイザ・アクセス権が必要です。

例

```
    /* This function gets the array of performance stats */

ESS_STS_T ESSGetPerfStats(ESS_HCTX_T *context)
{
 ESS_STS_T  sts;
 ESS_ULONG_T bufsize;
 ESS_PUCHAR_T poutarray;  /* Pointer to the stats staging area */

 /* Get the size of the output buffer */
 if(sts = EssGetStatBufSize(context, &bufsize))
  return(sts);

 if(bufsize)
 {
  /* Allocate a staging area */
  (ESS_PVOID_T)(poutarray) = malloc (bufsize);

  /* Fill the staging area */
  sts = EssDumpPerfStats(context, poutarray);
  if(sts)
   return(sts);

  /* Do something useful with the stats here */
  /* ..................................... */

  /* Free the staging area */
  sts = EssFree(context, poutarray);
  if(sts)
   return(sts);
 }
 else
 {
  printf("Performance Statistics not enabled, call ResetPerfStats()\n");
 }
 return(ESS_STS_NOERR);
}
```

関連トピック

- EssGetStatBufSize
- EssResetPerfStats

EssEndCalc

アクティブなデータベースに送信される計算スクリプトの終わりをマークします。この関数は、EssSendString を使用して計算スクリプトを送信した後に呼び出す必要があります。

構文

```
ESS_FUNC_M EssEndCalc (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数より前に EssBeginCalc を呼び出して、少なくとも 1 回は EssSendString を呼び出している必要があります。
- EssBeginCalc、EssSendString および EssEndCalc の呼出しが正常に行われた場合、呼出し元は ESS_STATE_DONE が戻されるまで EssGetProcessState を呼び出して、プロセスが完了したことを定期的に確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Calc  (ESS_HCTX_T     hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     Script;
 ESS_PROCSTATE_T  pState;
 Script = "CALC ALL;";

 sts = EssBeginCalc (hCtx,ESS_TRUE);
 if (!sts)
  sts = EssSendString (hCtx, Script);
 if (!sts)
  sts = EssEndCalc (hCtx);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
     ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
 }
 return(sts);
}
```

関連トピック

- EssBeginCalc

- EssCalc
- EssSendString

EssEndDataload

アクティブなデータベースに送信される更新指定の終わりをマークします。この関数は、EssSendString を使用して更新指定を送信した後に呼び出す必要があります。

構文

```
ESS_STS_T EssEndDataload (
hCtx, ppMbrError
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
ppMbrError;	156 ページの「ESS_MBRERR_T」	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラー(およびエラー文字列)は次のとおりです: ● ESS_MBRERR_UNKNOWN（未定義メンバー[membername]がデータロード中です。[number]レコードが戻されました。） ● ESS_MBRERR_DBACCESS（アクセス権限が不適切なため、このデータベースではロックを実行できません。） ● ESS_MBRERR_BADDATA（データ列に無効なメンバー[membername]が存在します。） ● ESS_MBRERR_DUPLICATE（データ・レコードの同じ次元からのメンバーが重複しています。[number]レコードが完了しました。） ● AD_MSGDL_ERRORLOAD（アイテム/レコード[number]でのデータロードができません。）

備考

- この関数より前に EssBeginDataload を呼び出して、少なくとも 1 回は EssSendString を呼び出している必要があります。
- ppMbrErr に対して割り当てられたメモリーは、EssFreeMbrErr を使用して解放する必要があります。

戻り値

成功の場合、0 が戻されます。それ以外の場合は、次のエラー・コードが戻されます:

- abortOnError が TRUE の場合:
  - 最初のエラー条件のエラー・コードが戻されます。
  - エラー・リストは NULL です。
- abortOnError が FALSE の場合:
  - サーバーがデータを処理して続行できる場合は、エラー・リストが戻されます。

- それ以外の場合は、例外状況でサーバーが続行できない理由を説明する次のようなエラー・コードが戻されます。例:

  AD_MSGDL_COLS (レコードに含まれるデータ値が多すぎる)

  AD_MSGDL_MISDIM (すべての次元を選択する前にデータ値を検出した)

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_STS_T   sts = ESS_STS_NOERR;
ESS_BOOL_T   Store;
ESS_BOOL_T   Unlock;
ESS_STR_T   Query1, Query2;
ESS_PMBRERR_T pMbrErr;

 Store  = ESS_TRUE;
 Unlock = ESS_FALSE;
 Query1 = "Year Market Scenario Measures Product 12345";
 Query2 = " Jan  East  Scenario Measures Coke  125";

 /* Begin Update */
 sts = EssBeginDataload (hCtx, Store, Unlock, ESS_FALSE, ESS_NULL);

 /* Send update specification */
 if(!sts)
  sts = EssSendString(hCtx, Query1);
  sts = EssSendString(hCtx, Query2);

 /* End Update */
 if(!sts)
  sts = EssEndDataload(hCtx, &pMbrErr);
```

関連トピック

- EssBeginDataload
- EssSendString
- EssBeginUpdate
- EssEndUpdate
- EssUpdate

EssEndIncrementalBuildDim

構築次元の四捨五入を完了します: アウトラインを確認し、エラーがなければ、アウトラインを書いて閉じ、再構築します。アウトラインにエラーがある場合、"szTmpOtlFile"で指定されたアウトライン・ファイルにアウトラインを書いて、アウトラインを閉じます。EAS アウトライン・エディタなどのアウトライン編集ツールを使用して、アウトラインに問題がないか確認できます。

構文

```
ESS_FUNC_M EssEndIncrementalBuildDim (
hCtx, restructOption, szTmpOtlFile, ErrorName, bOverwrite
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
restructOption	ESS_SHORT_T	再構築オプション。有効な値: ● ESS_DOR_ALLDATA すべてのデータを保持します ● ESS_DOR_NODATA すべてのデータを破棄します ● ESS_DOR_LOWDATA レベル 0 のすべてのデータを保持します ● ESS_DOR_INDATA すべての入力データを保持します
szTmpOtlFile	ESS_STR_T	一時アウトライン・ファイル名。
ErrorName	ESS_STR_T	クライアントでのエラー出力ファイルの名前。
bOverwrite	ESS_BOOL_T	ブール。値: ● ESS_TRUE – 既存のエラー・ファイルを上書きします。 ● ESS_FALSE – 上書きしません。既存のエラー・ファイルに追加します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_IncBuildDim( ESS_HCTX_T hCtx)
{
    ESS_STS_T   sts = 0;
    ESS_OBJDEF_T  RulesObj;
    ESS_OBJDEF_T  DataObj;
    ESS_STR_T   ErrorName;
    ESS_APPNAME_T appname;
    ESS_DBNAME_T  dbname;

 memset(&RulesObj,0,sizeof(ESS_OBJDEF_T));
 memset(&DataObj,0,sizeof(ESS_OBJDEF_T));
 strcpy(appname, "sample");
 strcpy(dbname,"basic");

 RulesObj.hCtx   = hCtx;
 RulesObj.FileName = "genref";
```

```
 RulesObj.AppName  = appname;
 RulesObj.DbName   = dbname;
 RulesObj.ObjType  = ESS_OBJTYPE_RULES;

 DataObj.hCtx    = hCtx;
 DataObj.FileName  = "genref";
 DataObj.AppName   = appname;
 DataObj.DbName    = dbname;
 DataObj.ObjType   = ESS_OBJTYPE_TEXT;

 ErrorName     = "builddim.err";

 sts = EssBeginIncrementalBuildDim(hCtx);

 if (!sts)
  sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataObj,NULL,ErrorName,true,ESS_INCDIMBUILD_BU
ILD,NULL);
 if (!sts)
  sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_VER
IFY,NULL);
 if (!sts)
  sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_SAV
EOTL,"tmpotl");


 sts = EssBeginStreamBuildDim(hCtx, &RulesObj,ESS_INCDIMBUILD_BUILD,"tmpotl");
 if (!sts)
  sts = EssSendString(hCtx, "600    600-20    600-20-20\n");
 if (!sts)
  sts = EssSendString(hCtx, "600    600-20    600-20-30\n");
 if (!sts)
  sts = EssSendString(hCtx, "600    600-40    600-40-20\n");
 sts = EssEndStreamBuildDim(hCtx,ErrorName,false);


 sts = EssEndIncrementalBuildDim(hCtx,ESS_DOR_ALLDATA,"tmpotl",ErrorName,false);
 return sts;
}
```

関連トピック

- EssIncrementalBuildDim
- EssBeginIncrementalBuildDim
- EssBeginStreamBuildDim
- EssEndIncrementalBuildDim
- EssEndStreamBuildDim

EssEndReport

アクティブなデータベースに送信されるレポート指定の終わりをマークします。この関数は、(EssSendString を使用して)レポート指定を送信してから(EssGetString を使用して)戻されたデータを読み取る前に呼び出す必要があります。

構文

```
ESS_FUNC_M EssEndReport (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数より前に EssBeginReport を呼び出し、少なくとも 1 回は EssSendString を呼び出している必要があります。
- レポート・シーケンスを開始する EssBeginReport の呼出しで出力フラグが TRUE の場合は、EssEndReport への呼出しの後に NULL 文字列が戻されるまで、EssGetString への呼出しを繰り返す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
     ESS_HINST_T  hInst
    )
{
 ESS_FUNC_M    sts    = ESS_STS_NOERR;
 ESS_STR_T   rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");
 if (!sts)
  sts = EssEndReport (hCtx);
 /**************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
```

```
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

関連トピック

- EssBeginReport
- EssGetString
- EssSendString

EssEndStreamBuildDim

次元構築プロセスを終了します。

この関数の前に EssBeginStreamBuildDim を呼び出してから、EssSendString を 1 回以上呼び出し、Essbase サーバーにソース・レコードを送信する必要があります。

構文

```
ESS_FUNC_M EssEndStreamBuildDim (
hCtx, ErrorFileName, ErFileOverWrite
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ErrorFileName	ESS_STR_T	クライアントでのエラー出力ファイルの名前。
ErFileOverWrite	ESS_BOOL_T	ブール。値: • ESS_TRUE – 既存のエラー・ファイルを上書きします。 • ESS_FALSE – 上書きしません。既存のエラー・ファイルに追加します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_IncBuildDim( ESS_HCTX_T hCtx)
{
    ESS_STS_T   sts = 0;
    ESS_OBJDEF_T  RulesObj;
```

```
    ESS_OBJDEF_T  DataObj;
    ESS_STR_T   ErrorName;
    ESS_APPNAME_T appname;
    ESS_DBNAME_T  dbname;

 memset(&RulesObj,0,sizeof(ESS_OBJDEF_T));
 memset(&DataObj,0,sizeof(ESS_OBJDEF_T));
 strcpy(appname, "sample");
 strcpy(dbname,"basic");

 RulesObj.hCtx   = hCtx;
 RulesObj.FileName = "genref";
 RulesObj.AppName  = appname;
 RulesObj.DbName   = dbname;
 RulesObj.ObjType  = ESS_OBJTYPE_RULES;

 DataObj.hCtx    = hCtx;
 DataObj.FileName  = "genref";
 DataObj.AppName   = appname;
 DataObj.DbName    = dbname;
 DataObj.ObjType   = ESS_OBJTYPE_TEXT;

 ErrorName     = "builddim.err";

 sts = EssBeginIncrementalBuildDim(hCtx);

 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataObj,NULL,ErrorName,true,ESS_INCDIMBUILD_BU
ILD,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_VER
IFY,NULL);
 if (!sts)
   sts =
EssIncrementalBuildDim(hCtx,&RulesObj,&DataOb,NULL,ErrorName,true,ESS_INCDIMBUILD_SAV
EOTL,"tmpotl");


 sts = EssBeginStreamBuildDim(hCtx, &RulesObj,ESS_INCDIMBUILD_BUILD,"tmpotl");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-20\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-20    600-20-30\n");
 if (!sts)
   sts = EssSendString(hCtx, "600    600-40    600-40-20\n");
 sts = EssEndStreamBuildDim(hCtx,ErrorName,false);


 sts = EssEndIncrementalBuildDim(hCtx,ESS_DOR_ALLDATA,"tmpotl",ErrorName,false);
 return sts;
}
```

関連トピック

- EssIncrementalBuildDim
- EssBeginIncrementalBuildDim
- EssBeginStreamBuildDim
- EssEndIncrementalBuildDim
- EssEndStreamBuildDim

EssEndUpdate

アクティブなデータベースに送信される更新指定の終了をマークします。この関数は、EssSendString を使用して更新指定を送信した後に呼び出す必要があります。

構文

```
ESS_FUNC_M EssEndUpdate (
hCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

この関数より前に EssBeginUpdate を呼び出し、少なくとも 1 回は EssSendString を呼び出している必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

EssBeginUpdate の例を参照してください。

関連トピック

- EssBeginUpdate
- EssSendString
- EssUpdate

EssExport

データベースを ASCII ファイルにエクスポートします

構文

```
      ESS_FUNC_M EssExport (
      hCtx, AppName, DbName, PathName,
Level, Columns
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	エクスポートするアプリケーション名。
DbName	ESS_STR_T	エクスポートするデータベース名。
PathName	ESS_STR_T	エクスポートした情報を含むサーバー・ファイルのフル・パス名。
Level	ESS_SHORT_T	エクスポートするデータのレベルを制御します。次のいずれかの値にする必要があります: ● ESS_DATA_ALL – データのすべてのレベルをエクスポートします ● ESS_DATA_LEVEL0 – レベル 0 のブロックのデータのみすべてエクスポートします ● ESS_DATA_INPUT – 入力レベル・ブロックのデータのみエクスポートします
Columns	ESS_SHORT_T	列フォーマットのデータ・ブロックの出力を制御します(ルール・ファイルを作成するため)。列フォーマットの場合は非ゼロを使用し、列フォーマット以外の場合はゼロを使用します。

備考

スレッドのデータが 2GB を超える場合、Essbase によってエクスポート・データが複数のファイルに分割され、ファイル名には数値が追加されます。

追加エクスポート・ファイルの命名ルールは次のとおりです: _1、_2 などが追加ファイル名に付加されます。指定した出力ファイル名にピリオドが含まれている場合は、ピリオドの前に数値が付加されます。そうでない場合は、ファイル名の末尾に付加されます。

たとえば、指定したファイル名が/home/exportfile.txt の場合は、次の追加ファイルは/home/exportfile_1.txt になります。ファイル名が/home/exportfile の場合は、次の追加ファイルは/home/exportfile_1 になります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
```

```
ESS_Export  (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_SHORT_T   isLevel;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FileName;
 ESS_PROCSTATE_T pState;

 isLevel  = ESS_DATA_LEVEL0;
 AppName  = "Sample";
 DbName   = "Basic";

 sts = EssExport (hCtx, appName, dbName,
    "D:\\temp\\asofile.txt", ESS_DATA_LEVEL0, ESS_FALSE);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
      ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
 }
 return (sts);
}
```

関連トピック

- EssImport

EssFixIBH

データベース内の無効なブロック・ヘッダー破損を修復します。現在、すべての無効なブロックをデータベースから削除します。

構文

```
ESS_FUNC_M EssFixIBH (
hCtx
,
action
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
action;	ESS_IBH_ACTION	列挙タイプ。このリリースでは、有効な値は REMOVE のみです。

関連トピック

- EssLocateIBH
- EssGetIBH

EssFree

定義されたメモリー割当ての仕組みを使用して、以前に割り当てたメモリー・ブロックを解放します。

構文

```
ESS_FUNC_M EssFree (
hInstance, pBlock
);
```

パラメータ　データ型　　説明

hInstance　ESS_HINST_T API インスタンス・ハンドル。

pBlock　　ESS_PVOID_T 割り当てられているメモリー・ブロックへのポインタ。

備考

- この関数は、EssInit 関数に渡されたユーザー指定のメモリー管理関数を使用してメモリーを解放します。このような関数が指定されていない場合、デフォルトのメモリー解放関数(プラットフォーム依存)が使用されます。
- この関数は EssAlloc と EssRealloc 関数を使用して割り当てられたメモリーを解放するために使用されます。また、Essbase API 関数から戻された割当て済バッファの解放にも使用されます。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_GetAppActive (ESS_HCTX_T    hCtx,
        ESS_HINST_T    hInst
        )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   pDbName;
 ESS_STR_T   pAppName;
 ESS_ACCESS_T  Access;

 if ((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pAppName)) == 0)
 {
  if ((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pDbName)) == 0)
  {
   if ((sts = EssGetActive (hCtx, &pAppName, &pDbName, &Access)) == 0)
   {
    if (pAppName)
    {
     if (*pAppName)
```

```
        printf ("Current active application is [%s]\r\n",pAppName);
       else
        printf ("No active Application is set\r\n");
       printf ("\r\n");
      }
     }
     EssFree (hInst, pDbName);
   }
   EssFree (hInst, pAppName);
 }
 return (sts);
}
```

関連トピック

- EssAlloc
- EssInit
- EssOtlGetMemberCommentEx
- EssOtlSetMemberCommentEx
- EssRealloc

EssFreeMbrErr

ESS_MBRERR_T 構造体のリンク・リストに割り当てられているメモリーを解放します。

構文

```
ESS_FUNC_M EssFreeMbrErr (
hCtx,

ppMbrErr
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ppMbrErr	ESS_PMBRERR_T	ESS_MBRERR_T に含まれているリンク・リストへのポインタ。

備考

- この関数は、EssImport で使用された ESS_MBRERR_T に割り当てられたメモリーを解放するためにのみ使用できます。
- ppMbrErr の詳細は、EssImport を参照してください。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

EssImport の例を参照してください。

関連トピック

- EssImport

EssFreeStructure

文字列タイプの属性情報のために EssGetAttributeInfo および EssGetMemberInfo によって動的に割り当てられたメモリーを解放します。

構文

パラメータ	データ型	説明
hInst;	ESS_HINST_T	構造体を割り当てるために EssGetAttributeInfo または EssGetMemberInfo を呼び出したプロセスのインスタンス・ハンドル。
structId;	ESS_ULONG_T	構造体に対する次の定数識別子のいずれかになります: ● ESS_DT_STRUCT_ATTRIBUTEINFO ● ESS_DT_STRUCT_ATTRSPECS ● ESS_DT_STRUCT_MEMBERINFO
count;	ESS_ULONG_T	構造体の数。
structPtr;	ESS_PVOID_T	メモリーへのポインタ。

備考

- ローカル・ルーチンを離れる前に、必ずこの関数を呼び出して、EssGetAttributeInfo または EssGetMemberInfo によって割り当てられた構造体を解放してください。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    void  ESS_GetAttributeSpecifications()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PATTRSPECS_T  pAttrSpecs;

 sts = EssGetAttributeSpecifications(hCtx, &pAttrSpecs);

 printf("\n ---------Attribute Specifications--------\n\n");
 if (sts) return(sts);

 switch(pAttrSpecs->usGenNameBy)
 {
  case ESS_GENNAMEBY_PREFIX:
```

```
   printf("\n Prefix/Suffix  : Prefix");
   break;
  case ESS_GENNAMEBY_SUFFIX:
   printf("\n Prefix/Suffix  : Suffix");
    break;
  default:
   printf("\n Prefix/Suffix  : None");
   break;
}
switch(pAttrSpecs->usUseNameOf)
{
 case ESS_USENAMEOF_PARENT:
  printf("\n Use Name of   : Parent");
  break;
 case ESS_USENAMEOF_GRANDPARENTANDPARENT:
  printf("\n Use Name of   : Grand Parent and Parent");
  break;
 case ESS_USENAMEOF_ALLANCESTORS:
  printf("\n Use Name of   : All Ancestors");
  break;
 case ESS_USENAMEOF_DIMENSION:
  printf("\n Use Name of   : Dimension");
  break;
 case ESS_USENAMEOF_NONE:
  printf("\n Use Name of   : None");
  break;
 default:
  printf("\n Use Name of   : Invalid setting");
  break;
}
switch(pAttrSpecs->cDelimiter)
{
 case ESS_DELIMITER_PIPE:
  printf("\n Delimiter    : '|'");
  break;
 case ESS_DELIMITER_UNDERSCORE:
  printf("\n Delimiter    : '_'");
  break;
 case ESS_DELIMITER_CARET:
  printf("\n Delimiter    : '^'");
  break;
 default:
  printf("\n Delimiter    : Invalid setting");
  break;
}

switch(pAttrSpecs->usDateFormat)
{
 case ESS_DATEFORMAT_DDMMYYYY :
  printf("\n Date Format   : DD-MM-YYYY");
  break;
 case ESS_DATEFORMAT_MMDDYYYY :
  printf("\n Date Format   : MM-DD-YYYY");
  break;
 default:
  printf("\n Date Format   : Invalid setting");
  break;
```

```
 }
 switch(pAttrSpecs->usBucketingType)
 {
  case ESS_UPPERBOUNDINCLUSIVE :
   printf("\n Bucketing Type  : Upper Bound inclusive");
   break;
  case ESS_UPPERBOUNDNONINCLUSIVE :
   printf("\n Bucketing Type  : Upper Bound non-inclusive");
   break;
  case ESS_LOWERBOUNDINCLUSIVE :
   printf("\n Bucketing Type  : Lower Bound inclusive");
   break;
  case ESS_LOWERBOUNDNONINCLUSIVE :
   printf("\n Bucketing Type  : Lower Bound non-inclusive");
   break;
  default:
   printf("\n Bucketing Type  : Invalid setting");
   break;
 }

 printf("\n Default for TRUE    : %s",pAttrSpecs->pszDefaultTrueString);
 printf("\n Default for FALSE   : %s",pAttrSpecs->pszDefaultFalseString);
 printf("\n Default for Attr Calc  : %s",pAttrSpecs->pszDefaultAttrCalcDimName);
 printf("\n Default for Sum     : %s",pAttrSpecs->pszDefaultSumMbrName);
 printf("\n Default for Count   : %s",pAttrSpecs->pszDefaultCountMbrName);
 printf("\n Default for Average   : %s",pAttrSpecs->pszDefaultAverageMbrName);
 printf("\n Default for Min     : %s",pAttrSpecs->pszDefaultMinMbrName);
 printf("\n Default for Max     : %s",pAttrSpecs->pszDefaultMaxMbrName);
 printf("\n");

 EssFreeStructure(hInst, ESS_DT_STRUCT_ATTRSPECS, 1,(ESS_PVOID_T)pAttrSpecs);
}
```

関連トピック

- EssCheckAttributes
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssGetActive

呼出し元の現在のアクティブなアプリケーションとデータベースの名前を取得します。

構文

```
ESS_FUNC_M EssGetActive (
hCtx, pAppName, pDbName, pAccess
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pAppName	ESS_PSTR_T	割り当てられたアプリケーション名文字列を受け取るポインタのアドレス。
pDbName	ESS_PSTR_T	割り当てられたデータベース名文字列を受け取るポインタのアドレス。
pAccess;	ESS_PACCESS_T	選択したデータベースに対するユーザーのアクセス・レベルを受け取る変数のアドレス。このフィールドに使用できる値のリストは、102 ページの「ビットマスク・データ型(C)」についての説明を参照してください。

備考

pAppName と pDbName に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合、ユーザーの選択されたアクティブ・アプリケーションおよびデータベースが pAppName と pDbName に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_GetAppActive (ESS_HCTX_T    hCtx,
        ESS_HINST_T    hInst
       )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   pDbName;
 ESS_STR_T   pAppName;
 ESS_ACCESS_T  Access;
 if ((sts = EssAlloc (hInst, 80,
   (ESS_PPVOID_T)&pAppName)) == 0)
 {
  if ((sts = EssAlloc (hInst, 80,
    (ESS_PPVOID_T)&pDbName)) == 0)
  {
   if ((sts = EssGetActive (hCtx, &pAppName,
     &pDbName, &Access)) == 0)
```

```
    {
     if (pAppName)
     {
      if (*pAppName)
       printf ("Current active application is [%s]\r\n",pAppName);
      else
       printf ("No active Application is set\r\n");
      printf ("\r\n");
     }
    }
    EssFree (hInst, pDbName);
   }
   EssFree (hInst, pAppName);
  }
  return (sts);
}
```

関連トピック

- EssClearActive
- EssSetActive

EssGetAlias

1 人のユーザーについて、アクティブなデータベースからアクティブな別名テーブル名を取得します。

構文

```
ESS_FUNC_M EssGetAlias (
hCtx, pAliasName
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pAliasName	ESS_PSTR_T	アクティブな別名テーブルに割り当てられている名前を受け取るポインタのアドレス。

備考

pAliasName に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、アクティブな別名テーブルの名前が pAliasName に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetAlias (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AliasName;

 sts = EssGetAlias(hCtx, &AliasName);

 if(!sts && AliasName)
 {
  printf("AliasName: %s\r\n",AliasName);
  EssFree(hInst,AliasName);
 }

 return (sts);
}
```

関連トピック

- EssListAliases
- EssSetAlias

EssGetAPIVersion

現在のアプリケーションのコンパイルに使用される Essbase API のバージョンを戻します。

構文

```
ESS_FUNC_M EssGetAPIVersion (
Version
);
```

パラメータ	データ型	説明
Version	ESS_PULONG_T	API のバージョン番号。次のフォーマットの、C 記法の 16 進値: 0x00000000 • 右から最初の 4 つの数字(下位ワード): バージョン間のリリース番号 • 残りの数字(上位ワード): バージョン番号 たとえば、0x0004.0000 はリリース 4.0 を示し、0x0003.0002 はリリース 3.2 を示します。

備考

プログラムで特定のバージョンが必要な場合に、この関数で API のバージョンを確認できます。

例

```
    ESS_VOID_T
ESS_GetAPIVersion()
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_ULONG_T  Version;

 sts = EssGetAPIVersion(&Version);

 if(!sts)
  printf("API Version %x\n",Version);
}
```

関連トピック

- EssGetObjectInfo

EssGetApplicationAccess

アプリケーションへのユーザーのアクセス権情報が含まれているユーザー・アプリケーション・アクセス構造体のリストを取得します。

構文

```
ESS_FUNC_M EssGetApplicationAccess (
hCtx, UserName, AppName, pCount, ppUserApp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。NULL の場合は、指定されたアプリケーションのすべてのユーザーをリストします。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は、指定されたユーザーのすべてのアプリケーションをリストします。
pCount	ESS_PUSHORT_T	ユーザーのアプリケーション構造体のカウントを受け取る変数のアドレス。
ppUserApp	200 ページの「ESS_USERAPP_T、ESS_GROUPAPP_T」	割り当てられたユーザーのアプリケーション構造体の配列を受け取るポインタのアドレス。

備考

- UserName が NULL の場合、指定したアプリケーションのすべてのユーザーがリストされます。AppName が NULL の場合は、指定したユーザーのすべてのアプリケーションがリストされます。ただし、UserName および AppName の両方を NULL にすることはできません
- ユーザー・アプリケーションの構造体の Access フィールドは、アプリケーションに対してユーザーに与えられたアクセス権を表すのに使用されます。一方

MaxAccess フィールドは、すべてのソースから得られるユーザーの最高のアクセス権(たとえばグループを介したアクセス権やデフォルトのアプリケーション・アクセス権など)を表します。

- ppUserApp に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザーとアプリケーションのカウントが pCount に、ユーザーのアプリケーション構造体のリストが ppUserApp に戻されます。

アクセス

この関数を使用するには、独自のアプリケーションのアクセス情報を取得する場合を除き、呼出し元は指定したアプリケーションに対してアプリケーション・デザイナ権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetApplicationAccess (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    UserName;
 ESS_STR_T    AppName;
 ESS_USHORT_T  Count = 0;
 ESS_USHORT_T  ind;
 ESS_PUSERAPP_T  UserApp = NULL;

 UserName = "Admin";
 AppName  = "";

 sts = EssGetApplicationAccess(hCtx, UserName,
   AppName, &Count, &UserApp);
 if(!sts)
 {
  if(Count && UserApp)
  {
   printf ("\n------Application Access List----\n\n");
   for (ind = 0; ind < Count; ind++)
   {
   printf ("User->%s Application->%-10s
       Access->%-4d MaxAccess->%-6d\r\n",
       UserApp[ind].UserName,
       UserApp[ind].AppName,
       UserApp[ind].Access,
       UserApp[ind].MaxAccess);
   }
   EssFree (hInst, UserApp);
  }
  else
   printf ("\rUser Application list is empty\n\n");
 }
 return (sts);
}
```

関連トピック

- EssGetApplicationAccessEx
- EssGetDatabaseAccess
- EssListUsers
- EssSetApplicationAccess
- EssSetUser

EssGetApplicationAccessEx

アプリケーションへのユーザーまたはグループのアクセス権情報が含まれているユーザーまたはグループのアプリケーション・アクセス構造体のリストを取得します。EssGetApplicationAccess に似ていますが、ユーザー・ディレクトリの指定、または UserID の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssGetApplicationAccessEx (
hCtx
,
UserId
,
bIsIdentity
,
type
,
AppName
,
pCount
,
ppUserApp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
UserId	ESS_STR_T	ユーザー名またはグループ名(入力)。name@provider または一意の ID 属性として指定できます。NULL の場合は、指定したアプリケーションのすべてのユーザーまたはグループがリストされます。
bIsIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。
type	ESS_USHORT_T	エンティティのタイプ(入力)。UserID がグループかユーザーかを示します。次のいずれかになります: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
AppName	ESS_STR_T	アプリケーション名(入力)。NULL の場合は、指定されたユーザーのすべてのアプリケーションをリストします。

パラメータ	データ型	説明
pCount	ESS_PUSHORT_T	ユーザーのアプリケーション構造体のカウントを受け取る変数のアドレス(出力)。
ppUserApp	ESS_PPUSERAPPEX_T	割り当てられたユーザー・アプリケーション構造体の配列を受け取るポインタのアドレス(出力)。ユーザー・アプリケーション構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

- UserID が NULL の場合、指定したアプリケーションのすべてのユーザーがリストされます。AppName が NULL の場合は、指定したユーザーのすべてのアプリケーションがリストされます。ただし、UserID および AppName の両方を NULL にすることはできません。
- ユーザー・アプリケーション構造体の Access フィールドは、アプリケーションに対してユーザーに与えられたアクセス権を表すのに使用されます。一方、MaxAccess フィールドは、すべてのソースにおけるユーザーの最も高いアクセス権(たとえばグループを介したアクセス権、デフォルトのアプリケーション・アクセス権など)を表します。
- ppUserApp に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザーとアプリケーションのカウントが pCount に、ユーザーのアプリケーション構造体のリストが ppUserApp に戻されます。

アクセス

この関数を使用するには、独自のアプリケーションのアクセス情報を取得する場合を除き、呼出し元は指定したアプリケーションに対してアプリケーション・デザイナ権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
void DisplayUserAppInfo(ESS_PUSERAPPEX_T userApp, ESS_USHORT_T count)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T ind;

 printf ("\n------Application Access List----\n\n");
 for (ind = 0; ind < count; ind++)
 {
  printf("\tUser: %s\n", userApp[ind].UserName);
  printf("\tProvider Name: %s\n", userApp[ind].ProviderName);
  printf("\tConnection Param: %s\n", userApp[ind].connparam);
  printf("\tAppName: %s\n", userApp[ind].AppName);
  switch(userApp[ind].Access)
  {
   case ESS_PRIV_NONE:
    printf("\tAccess: %d - ESS_PRIV_NONE\n", userApp[ind].Access);
```

```
 break;
case ESS_PRIV_READ:
 printf("\tAccess: %d - ESS_PRIV_READ\n", userApp[ind].Access);
 break;
case ESS_PRIV_WRITE:
 printf("\tAccess: %d - ESS_PRIV_WRITE\n", userApp[ind].Access);
 break;
case ESS_PRIV_CALC:
 printf("\tAccess: %d - ESS_PRIV_CALC\n", userApp[ind].Access);
 break;
case ESS_PRIV_METAREAD:
 printf("\tAccess: %d - ESS_PRIV_METAREAD\n", userApp[ind].Access);
 break;
case ESS_PRIV_DBLOAD:
 printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userApp[ind].Access);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userApp[ind].Access);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userApp[ind].Access);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userApp[ind].Access);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userApp[ind].Access);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userApp[ind].Access);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userApp[ind].Access);
 break;

case ESS_ACCESS_READ:
 printf("\tAccess: %d - ESS_ACCESS_READ\n", userApp[ind].Access);
 break;
case ESS_ACCESS_WRITE:
 printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userApp[ind].Access);
 break;
case ESS_ACCESS_CALC:
 printf("\tAccess: %d - ESS_ACCESS_CALC\n", userApp[ind].Access);
 break;
case ESS_ACCESS_METAREAD:
 printf("\tAccess: %d - ESS_ACCESS_METAREAD\n", userApp[ind].Access);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userApp[ind].Access);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userApp[ind].Access);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userApp[ind].Access);
 break;
case ESS_ACCESS_APPCREATE:
```

```
    printf("\tAccess: %d - ESS_ACCESS_APPCREATE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_FILTER:
    printf("\tAccess: %d - ESS_ACCESS_FILTER\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_DBALL:
    printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_APPALL:
    printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_ADMIN:
    printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userApp[ind].Access);
    break;
  default:
    printf("\tAccess: Unknown\n");
}

switch(userApp[ind].MaxAccess)
{
  case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_PRIV_CALC\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_METAREAD:
    printf("\tMax Access: %d - ESS_PRIV_METAREAD\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_DBLOAD:
    printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_DBMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_DBCREATE:
    printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_APPLOAD:
    printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_APPMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_APPCREATE:
    printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_USERCREATE:
    printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userApp[ind].MaxAccess);
    break;
```

```
    case ESS_ACCESS_READ:
     printf("\tMax Access: %d - ESS_ACCESS_READ\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_WRITE:
     printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_CALC:
     printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_METAREAD:
     printf("\tMax Access: %d - ESS_ACCESS_METAREAD\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_DBMANAGE:
     printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_DBCREATE:
     printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_APPMANAGE:
     printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_APPCREATE:
     printf("\tMax Access: %d - ESS_ACCESS_APPCREATE\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_FILTER:
     printf("\tMax Access: %d - ESS_ACCESS_FILTER\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_DBALL:
     printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_APPALL:
     printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userApp[ind].MaxAccess);
     break;
    case ESS_ACCESS_ADMIN:
     printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userApp[ind].MaxAccess);
     break;
    default:
     printf("\tMax Access: Unknown\n");
   }

   printf("\n");
  }
}

ESS_FUNC_M ESS_GetApplicationAccessEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T userId;
 ESS_BOOL_T bIsIdentity;
 ESS_USHORT_T type;
 ESS_USHORT_T count = 0;
 ESS_USERAPPEX_T userApp[2];
 ESS_PUSERAPPEX_T pUserApp = ESS_NULL;

 count = 1;
 strcpy(userApp[0].UserName, "IDUser1");
```

```
 strcpy(userApp[0].ProviderName, "");
 strcpy(userApp[0].connparam, "");
 userApp[0].type = ESS_TYPE_USER;
 strcpy(userApp[0].AppName, AppName);
 userApp[0].Access = ESS_PRIV_APPMANAGE;
 userApp[0].MaxAccess = ESS_PRIV_APPMANAGE;
 sts = EssSetApplicationAccessEx(hCtx, count, &userApp);
 printf("EssSetApplicationAccessEx sts: %ld\n", sts);

 userId = "IDUser1";
 AppName = AppName;
 type = ESS_TYPE_GROUP;
 bIsIdentity = ESS_FALSE;
 sts = EssGetApplicationAccessEx(hCtx, userId, bIsIdentity, type, AppName, &count,
&pUserApp);
 printf("EssGetApplicationAccessEx sts: %ld\n", sts);
 if(!sts)
 {
  if(count && pUserApp)
  {
   DisplayUserAppInfo(pUserApp, count);
   sts = EssFree (hInst, pUserApp);
  }
  else
   printf ("\rUser Application list is empty\n\n");
 }
 return (sts);
}
```

関連トピック

- EssGetDatabaseAccessEx
- EssListUsersInfoEx
- EssSetApplicationAccessEx

EssGetApplicationInfo

ユーザーが構成不可能なアプリケーションのパラメータが含まれている、アプリケーションの情報構造体を取得します。

構文

```
ESS_FUNC_M EssGetApplicationInfo (
hCtx, AppName, ppAppInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(ログイン済)。
AppName	ESS_STR_T	アプリケーション名。

パラメータ	データ型	説明
ppAppInfo	120 ページの「ESS_APPINFO_T」	割り当てられたアプリケーション情報構造体を受け取るポインタのアドレス。

備考

- この関数は、サーバー上のアプリケーションに対してのみ呼び出せます。
- ppAppInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、割り当てられたアプリケーション情報構造体へのポインタが ppAppInfo に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定されたアプリケーションに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetAppInfo (ESS_HCTX_T    hCtx,
       ESS_HINST_T    hInst
       )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PAPPINFO_T  AppInfo;
 ESS_USHORT_T   ind;
 ESS_STR_T    AppName;
 AppName = "Sample";

 sts = EssGetApplicationInfo (hCtx, AppName, &AppInfo);
 if (!sts)
 {
  if (AppInfo)
  {
   printf ("\r\n-------Application Info-------\r\n\r\n");
   printf ("Name      : %s\r\n", AppInfo->Name);
   printf ("Server Name   : %s\r\n", AppInfo->Server);
   printf ("Status     : %d\r\n", AppInfo->Status);
   printf ("Users Connected : %d\r\n", AppInfo->nConnects);
   printf ("Number of DBs  : %d\r\n", AppInfo->nDbs);
   printf ("\r\n--List of Databases--\r\n\r\n");
   for (ind = 0; ind < AppInfo->nDbs; ind++)
    printf ("database(%d)   : %s\r\n", ind,
        AppInfo->DbNames [ind]);
   EssFree (hInst, AppInfo);
  }
 }

 return (sts);
}
```

関連トピック

- EssGetApplicationInfoEx
- EssGetApplicationState
- EssGetDatabaseInfo

EssGetApplicationInfoEx

1 つ以上のアプリケーションから情報を取得します。

構文

```
ESS_FUNC_M EssGetApplicationInfoEx (
hCtx, AppName, pusCount, ppAppInfoEx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(ログイン済)。
AppName	ESS_STR_T	情報が戻されるアプリケーション名。NULL の場合、すべてのアプリケーションについての情報が戻されます。
pusCount	ESS_PUSHORT_T	戻される情報構造体の数。
ppAppInfoEx	122 ページの「ESS_APPINFOEX_T」	割り当てられたアプリケーション情報構造体の配列へのポインタのアドレス。

備考

- この関数は、サーバー上のアプリケーションに対してのみ呼び出せます。
- ppAppInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、アプリケーション情報構造体の配列が ppAppInfo に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定されたアプリケーションに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetApplicationInfoEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 ESS_USHORT_T     ind;
 ESS_STR_T       AppName;
 ESS_USHORT_T     Count;
 ESS_PAPPINFOEX_T   AppInfoEx = NULL;
```

```
 AppName = "";
 sts = EssGetApplicationInfoEx (hCtx, AppName,
    &Count, &AppInfoEx);
 if(!sts)
 {
  if(AppInfoEx)
  {
printf("\n-----Application Info Ex -----\n\n");
    for (ind = 0; ind <Count; ind++)
    {
printf("Name:%s\r\n",AppInfoEx[ind].Name);
printf("Server Name:%s\r\n", AppInfoEx[ind].Server);
printf("Status:%d\r\n",AppInfoEx[ind].Status);
printf("Users Connected:%d\r\n",
    AppInfoEx[ind].nConnects);
printf("\r\n");
    }
   EssFree(hInst, AppInfoEx);
  }
 }
 return (sts);
}
```

関連トピック

- EssGetApplicationInfo
- EssGetApplicationState
- EssGetDatabaseInfo

EssGetApplicationState

ユーザーが構成可能なアプリケーションのパラメータが含まれている、アプリケーションの状態構造体を取得します。

構文

```
ESS_FUNC_M EssGetApplicationState (
hCtx, AppName, ppAppState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
ppAppState	123 ページの「ESS_APPSTATE_T」	割り当てられたアプリケーション状態構造体を受け取るポインタのアドレス。

備考

- この関数はローカル・アプリケーションに対しては呼び出せません。この関数はサーバーのアプリケーションに対してのみ呼び出せます。

- ppAppState に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、割り当てられたアプリケーション状態構造体へのポインタが ppAppState に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定されたアプリケーションに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetAppState (ESS_HCTX_T    hCtx,
        ESS_HINST_T   hInst
       )
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_PAPPSTATE_T  AppState;
 ESS_STR_T     AppName;
 AppName = "Sample";
 sts = EssGetApplicationState (hCtx, AppName,
    &AppState);
 if (!sts)
 {
  if (AppState)
  {
   EssFree (hInst, AppState);
  }
 }
 return (sts);
}
```

関連トピック

- EssGetApplicationInfo
- EssGetDatabaseState
- EssSetApplicationState

EssGetAssociatedAttributesInfo

指定した基本メンバーに関連付けられている属性メンバーを戻します。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
BaseMbrName;	ESS_STR_T	基本メンバー名。
AttrDimName;	ESS_STR_T	(オプション)属性次元名。

パラメータ	データ型	説明
pCount;	ESS_PULONG_T	戻された属性メンバー数。
ppAttrInfo;	124 ページの「ESS_ATTRIBUTEINFO_T」	属性情報。

備考

- この関数を呼び出すと、属性メンバーに関する情報を、EssQueryDatabaseMembers を使用して取得するよりも多く取得できます。
- AttrDimName を NULL に設定すると、基本メンバーに関連付けられているすべての属性メンバーが戻されます。
- オプションで、属性次元名を指定すると、基本メンバーに関連付けられているその次元メンバーに関する情報のみを取得できます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    //void  ESS_GetAssociateAttributeInfo();
ESS_GetAssociatedAttributesInfo ()
{
 ESS_STS_T       sts;
 ESS_ULONG_T      pCount=0;
 ESS_PATTRIBUTEINFO_T  pAttributeInfo;
 ESS_USHORT_T     index=0;
 ESS_CHAR_T      time_string[32];
 struct tm*      pTime;
 ESS_DATETIME_T    et;
 ESS_PATTRSPECS_T   pAttrSpecs;
 ESS_USHORT_T     usDateFormat;
 ESS_MBRNAME_T     attributeName;
 ESS_MBRNAME_T     dimensionName;

 pAttributeInfo = NULL;
 strcpy(attributeName, "100-10");
 strcpy(dimensionName, "\0");

 sts = EssGetAssociatedAttributesInfo(hCtx, attributeName, dimensionName, &pCount,
&pAttributeInfo);

 /* for handling time values */
 et = pAttributeInfo->Attribute.value.dtData;
 if (!sts)
 {
  printf ("\nAssociated Attr info for [%s]\n", attributeName);
  printf ("-----------------------------------\n");
  for (index=0; index<pCount; index++)
  {
   printf ("MbrName    : %s\n", pAttributeInfo[index].MbrName);
   printf ("DimName    : %s\n", pAttributeInfo[index].DimName);
```

```
    switch(pAttributeInfo[index].Attribute.usDataType)
    {
      case ESS_ATTRMBRDT_BOOL:
       printf ("Data Type   : Boolean \n");
       if ( pAttributeInfo[index].Attribute.value.bData)
        printf ("Data Value  : True \n");
       else
        printf ("Data Value  : False \n");
       break;

      case ESS_ATTRMBRDT_DOUBLE:
       printf ("Data Type   : Numeric(Double) \n");
       printf ("Data Value  : %g \n",pAttributeInfo[index].Attribute.value.dblData);
       break;

      case ESS_ATTRMBRDT_DATETIME:
       printf ("Data Type   : Date \n");
       sts = EssGetAttributeSpecifications(hCtx, &pAttrSpecs);
       if (sts)
        usDateFormat = ESS_DATEFORMAT_MMDDYYYY;
       else
        usDateFormat = pAttrSpecs->usDateFormat;

       pTime =  gmtime((time_t*)&et);
       switch(usDateFormat)
       {
        case ESS_DATEFORMAT_MMDDYYYY:
          sprintf(time_string, "MM-DD-YYYY %02i-%02i-%04i",
           pTime->tm_mon+1, pTime->tm_mday,pTime->tm_year+1900);
          break;
        case ESS_DATEFORMAT_DDMMYYYY :
          sprintf(time_string, "DD-MM-YYYY %02i-%02i-%04i",
           pTime->tm_mday,pTime->tm_mon+1, pTime->tm_year+1900);
          break;
       }
       printf ("Data Value  : %s \n", time_string);
       break;

      case ESS_ATTRMBRDT_STRING:
       printf ("Data Type   : String \n");
       printf ("Data Value  : %s \n", pAttributeInfo[index].Attribute.value.strData);
       EssFree(hInst, pAttributeInfo[index].Attribute.value.strData);
       break;
      }
    printf("\n");
   }
  }
  if (pAttributeInfo)
   EssFreeStructure(hInst, ESS_DT_STRUCT_ATTRIBUTEINFO, 1, pAttributeInfo);
  return (sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure

- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssGetAsyncProcLog

非同期データ・ロードまたは次元構築プロセスのエラー・ログを取得します。

構文

```
ESS_FUNC_M EssGetAsyncProcLog (
hCtx, ErrorFileName, ErFileOverWrite
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ErrorFileName	ESS_STR_T	エラー・ファイル名。
ErFileOverWrite	ESS_BOOL_T	TRUE の場合、エラー・ファイルを上書きします。

備考

EssAsyncImport または EssAsyncBuildDim を使用して非同期プロセスが開始された後に、この関数を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

例

EssAsyncBuildDim の例を参照してください。

関連トピック

- EssAsyncBuildDim
- EssAsyncImport
- EssAsyncImportASO
- EssGetAsyncProcState
- EssCancelAsyncProc

- EssCloseAsyncProc

EssGetAsyncProcState

非同期データ・ロードまたは次元構築プロセスの非同期プロセスの状態をクエリーします。

構文

```
ESS_FUNC_M EssGetAsyncProcState (
hCtx, pBldDlState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pBldDlState	ESS_PBLDDL_STATE_T	割り当てられたプロセス状態構造体を受け取るポインタのアドレス。

備考

EssAsyncImport または EssAsyncBuildDim を使用して非同期プロセスが開始された後に、この関数を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

例

EssAsyncBuildDim の例を参照してください。

関連トピック

- EssAsyncBuildDim
- EssAsyncImport
- EssAsyncImportASO
- EssGetAsyncProcLog
- EssCancelAsyncProc
- EssCloseAsyncProc

EssGetAttributeInfo

指定した属性メンバーまたは属性次元に関する属性情報を戻します。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
szAttributeName;	ESS_STR_T	属性メンバーまたは次元の名前。
pAttributeInfo;	124 ページの「ESS_ATTRIBUTEINFO_T」	属性情報。

備考

- この関数を呼び出した後は、EssFreeStructure を呼び出して、文字列タイプの属性情報に対してこの関数により動的に割り当てられたメモリーを解放します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    void  ESS_GetAttributeInfo()
{
 ESS_STS_T       sts;
 ESS_PATTRIBUTEINFO_T  pAttributeInfo;
 ESS_CHAR_T      time_string[32];
 struct tm*      pTime;
 ESS_DATETIME_T    et;
 ESS_PATTRSPECS_T   pAttrSpecs;
 ESS_USHORT_T     usDateFormat;

 /* sts = EssGetAttributeInfo(hCtx, "ounces_12", &pAttributeInfo); */
 /* sts = EssGetAttributeInfo(hCtx, "ounces", &pAttributeInfo); */
 /* sts = EssGetAttributeInfo(hCtx, "caffeinated_true", &pAttributeInfo); */
 /* sts = EssGetAttributeInfo(hCtx, "caffeinated", &pAttributeInfo); */
 sts = EssGetAttributeInfo(hCtx, "intro date_10-01-1996", &pAttributeInfo);
 /* sts = EssGetAttributeInfo(hCtx, "intro date", &pAttributeInfo); */
 /* sts = EssGetAttributeInfo(hCtx, "can", &pAttributeInfo); */
 /* sts = EssGetAttributeInfo(hCtx, "pkg type", &pAttributeInfo); */

 if(sts)
  fprintf(stderr,"Error in EssGetAttributeInfo(): %ld", sts);

 /* for handling time values */
 et = pAttributeInfo->Attribute.value.dtData;
 printf("Member name: %s\n", pAttributeInfo->MbrName);
 printf("Dimension name: %s\n", pAttributeInfo->DimName);
 /* printf("Attribute: %s\n", pAttributeInfo->Attribute); */
 switch(pAttributeInfo->Attribute.usDataType)
 {
   case ESS_ATTRMBRDT_BOOL:
    printf ("Data Type   : Boolean \n");
    if ( pAttributeInfo->Attribute.value.bData)
     printf ("Data Value  : True \n");
    else
     printf ("Data Value  : False \n");
    break;
```

```
    case ESS_ATTRMBRDT_DOUBLE:
     printf ("Data Type   : Numeric(Double) \n");
     printf ("Data Value  : %g \n",pAttributeInfo->Attribute.value.dblData);
     break;
    case ESS_ATTRMBRDT_DATETIME:
     printf ("Data Type   : Date \n");
    sts = EssGetAttributeSpecifications(hCtx, &pAttrSpecs);
     if (sts)
     usDateFormat = ESS_DATEFORMAT_MMDDYYYY;
     else
     usDateFormat = pAttrSpecs->usDateFormat;

     pTime =  gmtime((time_t*)&et);
     switch(usDateFormat)
     {
     case ESS_DATEFORMAT_MMDDYYYY:
      sprintf(time_string, "MM-DD-YYYY %02i-%02i-%04i",
       pTime->tm_mon+1, pTime->tm_mday,pTime->tm_year+1900);
      break;
     case ESS_DATEFORMAT_DDMMYYYY :
      sprintf(time_string, "DD-MM-YYYY %02i-%02i-%04i",
       pTime->tm_mday,pTime->tm_mon+1, pTime->tm_year+1900);
      break;
     }
     printf ("Data Value  : %s \n", time_string);
     break;
    case ESS_ATTRMBRDT_STRING:
     printf ("Data Type   : String \n");
     printf ("Data Value  : %s \n", pAttributeInfo->Attribute.value.strData);
     EssFree(hInst, pAttributeInfo->Attribute.value.strData);
     break;
    }
  EssFreeStructure(hInst, ESS_DT_STRUCT_ATTRIBUTEINFO, 1, pAttributeInfo);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssGetAttributeSpecifications

アウトラインの属性指定を取得します。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
pAttrSpecs;	125 ページの「ESS_ATTRSPECS_T」	属性指定。

備考

- EssOtlSetAttributeSpecifications を使用して、アウトラインの属性指定を設定します。
- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供

  表 6 を参照してください。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    void  ESS_GetAttributeSpecifications()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PATTRSPECS_T  pAttrSpecs;

 sts = EssGetAttributeSpecifications(hCtx, &pAttrSpecs);

 printf("\n ---------Attribute Specifications--------\n\n");
 if (sts) return(sts);

 switch(pAttrSpecs->usGenNameBy)
 {
  case ESS_GENNAMEBY_PREFIX:
   printf("\n Prefix/Suffix  : Prefix");
   break;
  case ESS_GENNAMEBY_SUFFIX:
   printf("\n Prefix/Suffix  : Suffix");
    break;
  default:
   printf("\n Prefix/Suffix  : None");
   break;
 }
```

```
switch(pAttrSpecs->usUseNameOf)
{
 case ESS_USENAMEOF_PARENT:
  printf("\n Use Name of   : Parent");
  break;
 case ESS_USENAMEOF_GRANDPARENTANDPARENT:
  printf("\n Use Name of   : Grand Parent and Parent");
  break;
 case ESS_USENAMEOF_ALLANCESTORS:
  printf("\n Use Name of   : All Ancestors");
  break;
 case ESS_USENAMEOF_DIMENSION:
  printf("\n Use Name of   : Dimension");
  break;
 case ESS_USENAMEOF_NONE:
  printf("\n Use Name of   : None");
  break;
 default:
  printf("\n Use Name of   : Invalid setting");
  break;
}
switch(pAttrSpecs->cDelimiter)
{
 case ESS_DELIMITER_PIPE:
  printf("\n Delimiter    : '|'");
  break;
 case ESS_DELIMITER_UNDERSCORE:
  printf("\n Delimiter    : '_'");
  break;
 case ESS_DELIMITER_CARET:
  printf("\n Delimiter    : '^'");
  break;
 default:
  printf("\n Delimiter    : Invalid setting");
  break;
}

switch(pAttrSpecs->usDateFormat)
{
 case ESS_DATEFORMAT_DDMMYYYY :
  printf("\n Date Format   : DD-MM-YYYY");
  break;
 case ESS_DATEFORMAT_MMDDYYYY :
  printf("\n Date Format   : MM-DD-YYYY");
  break;
 default:
  printf("\n Date Format   : Invalid setting");
  break;
}
switch(pAttrSpecs->usBucketingType)
{
 case ESS_UPPERBOUNDINCLUSIVE :
  printf("\n Bucketing Type  : Upper Bound inclusive");
  break;
 case ESS_UPPERBOUNDNONINCLUSIVE :
  printf("\n Bucketing Type  : Upper Bound non-inclusive");
  break;
```

```
   case ESS_LOWERBOUNDINCLUSIVE :
    printf("\n Bucketing Type  : Lower Bound inclusive");
    break;
   case ESS_LOWERBOUNDNONINCLUSIVE :
    printf("\n Bucketing Type  : Lower Bound non-inclusive");
    break;
   default:
    printf("\n Bucketing Type   : Invalid setting");
    break;
  }

  printf("\n Default for TRUE    : %s",pAttrSpecs->pszDefaultTrueString);
  printf("\n Default for FALSE    : %s",pAttrSpecs->pszDefaultFalseString);
  printf("\n Default for Attr Calc  : %s",pAttrSpecs->pszDefaultAttrCalcDimName);
  printf("\n Default for Sum     : %s",pAttrSpecs->pszDefaultSumMbrName);
  printf("\n Default for Count    : %s",pAttrSpecs->pszDefaultCountMbrName);
  printf("\n Default for Average   : %s",pAttrSpecs->pszDefaultAverageMbrName);
  printf("\n Default for Min     : %s",pAttrSpecs->pszDefaultMinMbrName);
  printf("\n Default for Max     : %s",pAttrSpecs->pszDefaultMaxMbrName);
  printf("\n");

  EssFreeStructure(hInst, ESS_DT_STRUCT_ATTRSPECS, 1,(ESS_PVOID_T)pAttrSpecs);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssGetCalcList

ユーザーがアクセス可能な計算スクリプト・オブジェクトのリストを取得します。

構文

```
ESS_FUNC_M EssGetCalcList (
hCtx, UserName, AppName, DbName, pAllCalcs, pCount, ppCalcList)
;
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pAllCalcs	ESS_PBOOL_T	すべての計算を許可するためのフラグを受け取るための変数のアドレス。TRUE の場合、ユーザーはすべての計算スクリプトにアクセスできます。それ以外の場合は、CalcList 引数で指定されている計算スクリプトにのみアクセスできます。
pCount	ESS_PUSHORT_T	アクセス可能な計算スクリプト・オブジェクト数のカウントを受け取る変数のアドレス。
ppCalcList	ESS_PPOBJNAME_T	割り当てられた計算スクリプト・オブジェクト名の配列を受け取るポインタのアドレス。

備考

- 計算スクリプト・オブジェクトにアクセスするには、指定されたユーザーが、少なくとも適切なデータベースに対する計算アクセス権を持っている必要があります。
- pAllCalcs フラグが TRUE に設定されている場合は、pCount は 0 で、ppCalcList は NULL です。
- ppCalcList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合、ユーザーのすべての計算の許可設定が pAllCalcs に、アクセス可能な計算スクリプト・オブジェクトのカウントが pCount に、計算スクリプト・オブジェクト名のリストが ppCalcList に戻されます。

アクセス

この関数を使用するには、呼出し元が自身の計算リストを取得しないかぎり、指定されたデータベースに対するデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetCalcList (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    UserName;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_BOOL_T    AllCalcs;
 ESS_USHORT_T  Count, ind;
 ESS_POBJNAME_T  pCalcList = NULL;

 UserName = "Admin";
```

```
 AppName  = "Sample";
 DbName   = "Basic";
 sts = EssGetCalcList(hCtx, UserName, AppName,
    DbName, &AllCalcs, &Count, &pCalcList);
 if(!sts && pCalcList)
 {
printf("-------- Get Calc List -----------\r\n");
   for (ind = 0; ind < Count; ind ++)
    printf(" %s\r\n",pCalcList[ind]);

   EssFree(hInst, pCalcList);
 }

 return (sts);
}
```

関連トピック

- EssListObjects
- EssListUsers
- EssSetCalcList

EssGetCookie

初期化時に Cookie が作成された場合に、現在のセッションに関連付けられている Cookie を取得します。詳細は、ESS_INIT_T で使用できるカスタム・コールバック関数 CookieCreateFunc を参照してください。

構文

```
ESS_FUNC_M EssGetCookie (ESS_HCTX_T, ESS_PPVOID_T);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ppCookie	ESS_PPVOID_T	Cookie を受け取るポインタのアドレス。

戻り値

正常終了すると、初期化時に作成された Cookie を戻します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

ESS_INIT_T の項にある、クエリーの取消しの例を参照してください。

EssGetCellDrillThruReports

データ・セルに関連付けられたドリルスルー・レポートを、セルのメンバーの組合せを使用し、URL XML のリストとして取得します。

構文

```
ESS_FUNC_M EssGetCellDrillThruReports (
hCtx, noMbrs, pMbrs, nURLXML, ppURLXMLLen, ppURLXML
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
noMbrs	ESS_USHORT_T	メンバー・リスト pMbrs に含まれるメンバー数。
pMbrs	ESS_PSTR_T	メンバー名(または別名)のリストへのポインタ。配列サイズは次元カウントであるとみなされます。
nURLXML	ESS_PUSHORT_T	戻された URL XML の数。
ppURLXMLLen	ESS_PPUSHORT_T	生成された URL XML の長さが戻されます。
ppURLXML	ESS_PPVOID_T	URL XML バイト・ストリームへのポインタが戻されます。

備考

この呼出しを行うためには、アプリケーション・データベースをアクティブに設定する必要があります。クライアントで必要とされる追加情報をサポートするには、この関数を拡張する必要があります。

戻り値

- 正常に処理されると、URL XML のリストが取得されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESS_PRIV_READ)を持っている必要があります。
- 呼出し側は EssSetActive を使用して、指定したデータベースをアクティブ・データベースとして選択しておく必要があります。

例

```
    /* Sample Code for EssGetCellDrillThruReports */

ESS_STS_T sts = ESS_STS_NOERR;
ESS_SHORT_T    numMbrs = 0;
ESS_STR_T    *pMbrs = ESS_NULL;
ESS_USHORT_T    numURLXML, i = 0;
ESS_USHORT_T    *URLXMLLen = ESS_NULL;
ESS_PPVOID_T    *URLXML = ESS_NULL;
ESS_CHAR_T    pTmpXML[XML_CHAR_MAX];
```

```
/* Valid case */

numMbrs = 5;
sts = EssAlloc (hInst, sizeof(ESS_STR_T) * numMbrs , &pMbrs);
pMbrs[0] = "Jul";
pMbrs[1] = "100-10";
pMbrs[2] = "Actual";
pMbrs[3] = "New York";
pMbrs[4] = "Sales";
sts = EssGetCellDrillThruReports(hCtx, numMbrs, pMbrs, &numURLXML, &URLXMLLen,
&URLXML);
printf("EssGetCellDrillThruReports sts: %ld\n",sts);
 if(!sts)
 {
  printf("\nNumber of URL XML: %d", numURLXML);
  for (i = 0; i < numURLXML; i++)
  {
   memset(pTmpXML, 0, XML_CHAR_MAX);
   memcpy(pTmpXML, URLXML[i], URLXMLLen[i]);

   if ( URLXML[i] != ESS_NULL )
    printf("\tXML [%d] : %s\n", i, pTmpXML );
   else
    printf("\tXML [%d] : NULL STRING \n", i );
  if ( URLXML[i] != ESS_NULL )
   EssFree(hInst, URLXML[i]);
  }
 if ( URLXML != ESS_NULL )
  EssFree(hInst, URLXML);
 if ( URLXMLLen != ESS_NULL )
  EssFree(hInst, URLXMLLen);
 }
```

EssGetCurrencyRateInfo

アクティブ・データベース・アウトライン内の、タグ付き通貨パーティション次元のすべてのメンバーについてのレート情報が含まれている構造体のリストを取得します。

構文

```
ESS_FUNC_M EssGetCurrencyRateInfo (
hCtx, pCount, ppRateInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCount	ESS_PLONG_T	レート情報構造体のカウントを受け取る変数のアドレス。

パラメータ	データ型	説明
ppRateInfo	189 ページの「ESS_RATEINFO_T」	割り当てられた通貨レート情報構造体の配列を受け取るポインタのアドレス。

備考

- ppRateInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- この関数は、関連通貨データベースが指定されている標準のデータベースに対して呼び出せます。

戻り値

正常終了の場合、構造体のカウントが pCount に、割り当てられた通貨レート情報構造体の配列が ppRateInfo に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetCrRate (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_LONG_T    count, i, j;
 ESS_PRATEINFO_T  pRateInfoList = NULL;
 ESS_CHAR_T    rateStr[(2 + ESS_MBRNAMELEN) * ESS_CRDB_MAXDIMNUM];
sts = EssGetCurrencyRateInfo (hCtx, &count, &pRateInfoList);
if (!sts)
 {
  if (count)
  {
   for (i = 0; i < count; i++)
   {
    rateStr[0] = '\0';
    for (j = 0; j < ESS_CRDB_MAXDIMNUM; j++)
    {
     if (pRateInfoList[i].RateMbr[j][0])
     {
      if (rateStr[0])
       strcat(rateStr, "->");

      strcat(rateStr, pRateInfoList[i].RateMbr[j]);
     }
    }
    if (!rateStr[0])
     strcpy(rateStr, "(LOCAL)");
    if (i == 0)
    {
```

```
       /* 1st is always DB rate */
       printf ("database [%s] : %s\r\n", pRateInfoList[i].MbrName, rateStr);
      }
      else
      {
       printf ("Partition [%s] : %s\r\n", pRateInfoList[i].MbrName, rateStr);
      }
     }
    }
  }
  if (pRateInfoList)
    EssFree (hInst, pRateInfoList);
 return (sts);
}
```

関連トピック

- EssListCurrencyDatabases
- EssSetActive

EssGetDatabaseAccess

データベースへのユーザーのアクセス権情報が含まれている、ユーザーのデータベース・アクセス構造体のリストを取得します。

構文

```
ESS_FUNC_M EssGetDatabaseAccess (
hCtx, UserName, AppName, DbName, pCount, ppUserDb
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。NULL の場合は、指定されたアプリケーションとデータベースのすべてのユーザーをリストします。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は、指定されたユーザーのアプリケーションとデータベースをすべてリストします。
DbName	ESS_STR_T	データベース名。NULL の場合は、指定されたユーザーまたはアプリケーションのすべてのデータベースをリストします。
pCount	ESS_PUSHORT_T	ユーザー・データベース構造体のカウントを受け取る変数のアドレス。
ppUserDb	202 ページの「ESS_USERDB_T、ESS_GROUPDB_T」	割り当てられたユーザー・データベース構造体の配列を受け取るポインタのアドレス。

備考

- UserName、AppName、DbName のいずれかが NULL の場合、ワイルドカードとして扱われ、該当するタイプのすべてのアイテムがリストされます。

AppName が NULL の場合、DbName も NULL とみなされます。これらの引数のうち 2 つが NULL でもかまいませんが、3 つすべてを NULL にすることはできません。

- ユーザー・データベース構造体の Access フィールドは、データベースに対してユーザーに与えられたアクセス権を表します。一方 MaxAccess フィールドは、すべてのソースから得られるユーザーの最も高いアクセス権(たとえばグループを介したアクセス権やデフォルトのデータベース・アクセス権など)を表します。
- ppUserDb に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- フィルタ・アクセス権限は、ESS_PRIV_DBLOAD 権限と同等です。

戻り値

正常終了の場合は、ユーザーとデータベースのカウントが pCount に、ユーザー・データベース構造体のリストが ppUserDb に戻されます。

アクセス

この関数を使用するには、独自のデータベース・アクセス情報を取得する場合を除き、呼出し元は指定されたデータベースに対するデータベース設計権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetDatabaseAccess (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    UserName;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_USHORT_T   Count = 0;
 ESS_USHORT_T   ind;
 ESS_PUSERDB_T  UserDb = NULL;

 UserName = "Admin";
 AppName  = "Sample";
 DbName   = "";

 sts = EssGetDatabaseAccess(hCtx, UserName,
     AppName, DbName, &Count, &UserDb);
 if(!sts)
 {
  if(Count && UserDb)
  {
printf ("\r\n-------Database Access List-------
        \r\n\r\n");
    for (ind = 0; ind < Count; ind++)
    {
printf("User -> %s\r\n",UserDb[ind].UserName);
printf("Application -> %s\r\n",
   UserDb[ind].AppName);
printf("Database -> %s\r\n",UserDb[ind].DbName);
```

```
printf("Access -> %d\r\n",UserDb[ind].Access);
printf("MaxAccess -> %d\r\n",
    UserDb[ind].MaxAccess);
printf("FilterName -> %s\r\n",
    UserDb[ind].FilterName);
printf("==================================\r\n");
    }
    EssFree (hInst, UserDb);
  }
  else
printf ("\r\nDatabase list is empty\r\n\r\n");
 }
 return (sts);
}
```

関連トピック

- EssGetDatabaseAccessEx
- EssGetApplicationAccess
- EssGetUser
- EssListUsers
- EssSetDatabaseAccess

EssGetDatabaseAccessEx

データベースへのユーザーのアクセス権情報が含まれている、ユーザーのデータベース・アクセス構造体のリストを取得します。EssGetDatabaseAccess に似ていますが、ユーザー・ディレクトリの指定、または UserID の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssGetDatabaseAccessEx (
hCtx
,
UserId
,
bIsIdentity
,
type
,
AppName
,
DbName
,
pCount
,
ppUserDb
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
UserId	ESS_STR_T	ユーザー名またはグループ名(入力)。name@provider または一意の ID 属性として指定できます。NULL の場合は、指定したデータベースのすべてのユーザーまたはグループがリストされます。
bIsIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。
type	ESS_USHORT_T	エンティティのタイプ(入力)。UserID がグループかユーザーかを示します。次のいずれかになります: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
AppName	ESS_STR_T	アプリケーション名(入力)。NULL の場合は、指定されたユーザーのアプリケーションとデータベースをすべてリストします。
DbName	ESS_STR_T	データベース名(入力)。NULL の場合は、指定されたユーザーまたはアプリケーションのすべてのデータベースをリストします。
pCount	ESS_PUSHORT_T	ユーザーのデータベース構造体のカウントを受け取る変数のアドレス(出力)。
ppUserDb	ESS_PPUSERDBEX_T	割り当てられたユーザー・データベース構造体の配列を受け取るポインタのアドレス(出力)。ユーザー・データベース構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

- UserID、AppName、DbName のいずれかが NULL の場合、ワイルドカードとして扱われ、該当するタイプのすべてのアイテムがリストされます。AppName が NULL の場合、DbName も NULL とみなされます。これらの引数のうち 2 つが NULL でもかまいませんが、3 つすべてを NULL にすることはできません。
- ユーザー・データベース構造体の Access フィールドは、データベースに対してユーザーに与えられたアクセス権を表します。一方 MaxAccess フィールドは、すべてのソースから得られるユーザーの最も高いアクセス権(たとえばグループを介したアクセス権やデフォルトのデータベース・アクセス権など)を表します。
- ppUserDb に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- フィルタ・アクセス権限は、ESS_PRIV_DBLOAD 権限と同等です。

戻り値

正常終了の場合は、ユーザーとデータベースのカウントが pCount に、ユーザー・データベース構造体のリストが ppUserDb に戻されます。

アクセス

この関数を使用するには、独自のデータベース・アクセス情報を取得する場合を除き、呼出し元は指定されたデータベースに対するデータベース設計権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void DisplayUserDbInfo(ESS_PUSERDBEX_T userDb, ESS_USHORT_T count)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T ind;

 printf ("\n------Database Access List----\n\n");
 for (ind = 0; ind < count; ind++)
 {
  printf("\tUser: %s\n", userDb[ind].UserName);
  printf("\tProvider Name: %s\n", userDb[ind].ProviderName);
  printf("\tConnection Param: %s\n", userDb[ind].connparam);
  printf("\tApp Name: %s\n", userDb[ind].AppName);
  printf("\tDb Name: %s\n", userDb[ind].DbName);
  switch(userDb[ind].Access)
  {
   case ESS_PRIV_NONE:
    printf("\tAccess: %d - ESS_PRIV_NONE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_READ:
    printf("\tAccess: %d - ESS_PRIV_READ\n", userDb[ind].Access);
    break;
   case ESS_PRIV_WRITE:
    printf("\tAccess: %d - ESS_PRIV_WRITE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_CALC:
    printf("\tAccess: %d - ESS_PRIV_CALC\n", userDb[ind].Access);
    break;
   case ESS_PRIV_METAREAD:
    printf("\tAccess: %d - ESS_PRIV_METAREAD\n", userDb[ind].Access);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userDb[ind].Access);
    break;
   case ESS_PRIV_DBMANAGE:
    printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_DBCREATE:
    printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_APPLOAD:
    printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userDb[ind].Access);
    break;
   case ESS_PRIV_APPMANAGE:
    printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_APPCREATE:
    printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_USERCREATE:
    printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userDb[ind].Access);
    break;
```

```
 case ESS_ACCESS_READ:
  printf("\tAccess: %d - ESS_ACCESS_READ\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_WRITE:
  printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_CALC:
  printf("\tAccess: %d - ESS_ACCESS_CALC\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_METAREAD:
  printf("\tAccess: %d - ESS_ACCESS_METAREAD\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_DBMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_DBCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_APPMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_APPCREATE:
  printf("\tAccess: %d - ESS_ACCESS_APPCREATE\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_FILTER:
  printf("\tAccess: %d - ESS_ACCESS_FILTER\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_DBALL:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_APPALL:
  printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userDb[ind].Access);
  break;
 case ESS_ACCESS_ADMIN:
  printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userDb[ind].Access);
  break;
 default:
  printf("\tAccess: Unknown\n");
}

switch(userDb[ind].MaxAccess)
{
 case ESS_PRIV_NONE:
  printf("\tMax Access: %d - ESS_PRIV_NONE\n", userDb[ind].MaxAccess);
  break;
 case ESS_PRIV_READ:
  printf("\tMax Access: %d - ESS_PRIV_READ\n", userDb[ind].MaxAccess);
  break;
 case ESS_PRIV_WRITE:
  printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userDb[ind].MaxAccess);
  break;
 case ESS_PRIV_CALC:
  printf("\tMax Access: %d - ESS_PRIV_CALC\n", userDb[ind].MaxAccess);
  break;
 case ESS_PRIV_METAREAD:
  printf("\tMax Access: %d - ESS_PRIV_METAREAD\n", userDb[ind].MaxAccess);
  break;
```

```
case ESS_PRIV_DBLOAD:
 printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userDb[ind].MaxAccess);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userDb[ind].MaxAccess);
 break;

case ESS_ACCESS_READ:
 printf("\tMax Access: %d - ESS_ACCESS_READ\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_WRITE:
 printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_CALC:
 printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_METAREAD:
 printf("\tMax Access: %d - ESS_ACCESS_METAREAD\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_APPCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_APPCREATE\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_FILTER:
 printf("\tMax Access: %d - ESS_ACCESS_FILTER\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_DBALL:
 printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_APPALL:
 printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userDb[ind].MaxAccess);
 break;
case ESS_ACCESS_ADMIN:
 printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userDb[ind].MaxAccess);
```

```
      break;
     default:
      printf("\tMax Access: Unknown\n");
    }

    printf("\tFilter Name: %s\n", userDb[ind].FilterName);
    printf("\n");
  }
}


ESS_FUNC_M ESS_GetDatabaseAccessEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_STR_T userId;
  ESS_BOOL_T bIsIdentity;
  ESS_USHORT_T type;
  ESS_USHORT_T count = 0;
  ESS_USERDBEX_T userDb[2];
  ESS_PUSERDBEX_T pUserDb = ESS_NULL;

  count = 2;
  strcpy(userDb[0].UserName, "IDUser1");
  strcpy(userDb[0].ProviderName, "");
  strcpy(userDb[0].connparam, "");
  userDb[0].type = ESS_TYPE_USER;
  strcpy(userDb[0].AppName, AppName);
  strcpy(userDb[0].DbName, DbName);
  userDb[0].Access = ESS_PRIV_READ;
  userDb[0].MaxAccess = ESS_PRIV_READ;

  strcpy(userDb[1].UserName, "");
  strcpy(userDb[1].ProviderName, "");
  strcpy(userDb[1].connparam, "");
  userDb[1].type = ESS_TYPE_USER;
  strcpy(userDb[1].AppName, AppName);
  strcpy(userDb[1].DbName, DbName);
  userDb[1].Access = ESS_ACCESS_ADMIN;
  userDb[1].MaxAccess = ESS_ACCESS_ADMIN;

  sts = EssSetDatabaseAccessEx(hCtx, count, &userDb);
  printf("EssSetDatabaseAccessEx sts: %ld\n\n", sts);

  userId = "IDUser1";
  AppName  = AppName;
  DbName  = DbName;
  type = ESS_TYPE_USER;
  bIsIdentity = ESS_TRUE;
  sts = EssGetDatabaseAccessEx(hCtx, userId, bIsIdentity, type, AppName, DbName,
&count, &pUserDb);
  printf("EssGetDatabaseAccessEx sts: %ld\n", sts);
  if(!sts)
  {
   if(count && pUserDb)
   {
    DisplayUserDbInfo(pUserDb, count);
```

```
    sts = EssFree (hInst, pUserDb);
   }
   else
    printf ("\rUser Application list is empty\n\n");
  }


 return (sts);
}
```

関連トピック

- EssGetApplicationAccessEx
- EssListUsersInfoEx
- EssSetDatabaseAccessEx

EssGetDatabaseInfo

ユーザーが構成不可能なデータベースのパラメータが含まれている、データベースの情報構造体を取得します。

構文

```
ESS_FUNC_M EssGetDatabaseInfo (
hCtx, AppName, DbName, ppDbInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
ppDbInfo	130 ページの「ESS_DBINFO_T」	割り当てられたデータベース情報構造体を受け取るポインタのアドレス。

備考

- ppDBInfo 構造体に割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- この関数は、サーバー・データベースの情報構造体のみを取得します。

戻り値

正常終了の場合は、割り当てられたデータベース情報構造体へのポインタが ppDbInfo に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定したデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetDbInfo (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PDBINFO_T  DbInfo;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";

 sts = EssGetDatabaseInfo (hCtx, AppName,
    DbName, &DbInfo);
 if (!sts)
 {
  if (DbInfo)
  {
   EssFree (hInst, DbInfo);
  }
 }
 return(sts);
}
```

関連トピック

- EssGetApplicationInfo
- EssGetDatabaseInfoEx
- EssGetDatabaseState
- EssGetDatabaseStats

EssGetDatabaseInfoEx

ユーザーが構成できないデータベース用パラメータを含む、1 つ以上のデータベースに関する情報を取得します。

構文

```
ESS_FUNC_M EssGetDatabaseInfoEx (
hCtx, AppName, DbName, pusCount
;
ppDbInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	データベース情報が戻されるアプリケーション名。NULL の場合、すべてのアプリケーションとデータベースについての情報が戻されます。

パラメータ	データ型	説明
DbName	ESS_STR_T	データベース情報が戻されるデータベース名。NULL の場合、すべてのデータベースについての情報が戻されます。
pusCount	ESS_PUSHORT_T	戻される情報構造体の数
ppDbInfo	130 ページの「ESS_DBINFO_T」	情報構造体の配列へのポインタ。

備考

- ppDBInfo 構造体に割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- この関数は、サーバー・データベースの情報構造体のみ取得できます。

戻り値

正常終了の場合は、データベース情報構造体の配列が戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定したデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetDatabaseInfoEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_PDBINFO_T  DbInfo = NULL;
 ESS_USHORT_T  Count;
 ESS_USHORT_T  ind;

 AppName = "Sample";
 DbName  = "";

 sts = EssGetDatabaseInfoEx(hCtx, AppName, DbName,
    &Count, &DbInfo);

 if(!sts && DbInfo)
 {
printf("\r\n------- Database Info Ex --------\r\n\r\n");
   for(ind = 0; ind < Count; ind++)
   {
printf("AppName: %s\r\n",DbInfo[ind].AppName);
printf("DbName: %s\r\n",DbInfo[ind].Name);
printf("DbType: %d\r\n",DbInfo[ind].DbType);
printf("Status: %d\r\n",DbInfo[ind].Status);
printf("nConnects: %d\r\n",DbInfo[ind].nConnects);
printf("nLocks: %d\r\n",DbInfo[ind].nLocks);
printf("-----------------------------------\r\n\r\n");
   }
   EssFree(hInst, DbInfo);
 }
```

```
    return (sts);
}
```

関連トピック

- EssGetApplicationInfo
- EssGetDatabaseInfo
- EssGetDatabaseState
- EssGetDatabaseStats

EssGetDatabaseNote

データベースの最新情報に関するメッセージを取得します。

構文

```
ESS_FUNC_M EssGetDatabaseNote (
hCtx, AppName, DbName, pDbNote
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pDbNote	ESS_PSTR_T	割り当てられたデータベース・ノート文字列を受け取るポインタのアドレス。

備考

- 最新情報に関するメッセージを使用して、ユーザーがデータベースに接続する前に、データベースに関する有用な情報(データがロードされているかどうか、データが最後に計算されたのはいつかなど)を表示できます。
- データベース・ノート文字列の長さは常に、64KB 未満である必要があります。
- データベースのノートは、EssSetDatabaseNote によって設定されます。
- pDbNote に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、割り当てられたデータベース・ノート文字列へのポインタが pDbNote に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定したデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

関連トピック

- EssSetDatabaseNote

EssGetDatabaseState

ユーザーが構成可能なデータベースのパラメータが含まれている、データベースの状態構造体を取得します。

構文

```
ESS_FUNC_M EssGetDatabaseState (
hCtx, AppName, DbName, ppDbState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
ppDbState	134 ページの「ESS_DBSTATE_T」	割り当てられたデータベース状態構造体を受け取るポインタのアドレス。

備考

- この関数は、サーバー・データベースの状態構造体のみを取得します。
- ppDbState 構造体に割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、割り当てられたデータベース状態構造体へのポインタが ppDbState に戻されます。

アクセス

データベースの状態構造体を取得するには、接続されているユーザーが少なくともデータベースに対する読取りアクセス権を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetCrType (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
       )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PDBSTATE_T  pDbState;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";
 sts = EssGetDatabaseState (hCtx, AppName,
    DbName, &pDbState);
 if (!sts)
 {
  if (pDbState)
```

```
   {
    if (pDbState->CrDbName)
    {
printf ("Currency Conversion Type Member:    %s\r\n", pDbState->CrTypeMember);
     if (pDbState->CrConvType ==
       ESS_CRCTYPE_DIV)
printf ("Currency Conversion Type:       %s\r\n", "ESS_CRCTYPE_DIV");
     else if (pDbState->CrConvType ==
         ESS_CRCTYPE_MULT)
printf ("Currency Conversion Type:       %s\r\n", "ESS_CRCTYPE_MULT");
    }
    else
printf ("No Currency database is set\r\n");
    EssFree (hInst, pDbState);
   }
  }
  return (sts);
}
```

関連トピック

- EssGetApplicationState
- EssGetDatabaseInfo
- EssSetDatabaseState
- EssGetDatabaseStats

EssGetDatabaseStats

データベースに関する統計情報を含むデータベースの統計構造体を取得します。

構文

```
ESS_FUNC_M EssGetDatabaseStats (
hCtx, AppName, DbName, ppDbStats
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
ppDbStats	137 ページの「ESS_DBSTATS_T」	割り当てられたデータベース統計構造体のポインタを受け取るポインタのアドレス。

備考

- この関数は、サーバー・データベースに対してのみ呼び出せます。
- データベースがまだロードされていない場合、この関数がデータベースをロードします。

- ppDbStats に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、割り当てられたデータベース統計構造体へのポインタが ppDbStats に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetDbStats (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
       )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PDBSTATS_T  pDbStats;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";

 sts = EssGetDatabaseStats (hCtx, AppName,
    DbName, &pDbStats);
 if (!sts)
 {
  if (pDbStats)
  {
   EssFree (hInst, pDbStats);
  }
 }

 return(sts);
}
```

関連トピック

- EssGetDatabaseInfo
- EssGetDatabaseState

EssGetDefaultCalc

アクティブなデータベースのデフォルトの計算スクリプトを取得します。

構文

```
     ESS_FUNC_M EssGetDefaultCalc (
     hCtx, pCalcScript
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCalcScript	ESS_PSTR_T	割り当てられた計算スクリプト文字列を受け取るポインタのアドレス。

戻り値

成功の場合、データベースのデフォルト計算スクリプトが pCalcScript に戻されます。

- 戻される計算スクリプト文字列の長さは、64KB 未満です。
- pCalcScript に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

アクセス

この関数では、呼出し元が少なくともデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持ち、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetDefaultCalc (ESS_HCTX_T  hCtx,
         ESS_HINST_T hInst
        )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   cstr = NULL;
 sts = EssGetDefaultCalc(hCtx, &cstr);
 if (!sts)
 {
  if (cstr)
  {
   printf  ("Default Calc Script --\r\n%s\r\n", cstr);
   EssFree (hInst, cstr);
  }
 }
 return (sts);
}
```

関連トピック

- EssDefaultCalc
- EssSetActive
- EssSetDefaultCalc
- EssSetDefaultCalcFile

EssGetDimensionInfo

次元に関する情報を取得します。

構文

```
ESS_FUNC_M EssGetDimensionInfo (
hCtx, MbrName, pDims, ppDimInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
MbrName	ESS_STR_T	情報が戻される次元メンバー名。NULL の場合、各次元についての情報が戻されます。メンバー名が無効の場合は、エラーになります。
pDims	ESS_PULONG_T	戻される情報構造体の数へのポインタ。
ppDimInfo	139 ページの「ESS_DIMENSIONINFO_T」	情報構造体の配列へのポインタ。

備考

- 139 ページの「ESS_DIMENSIONINFO_T」構造体の DimTag フィールドの定数値 ESS_TTYPE_ATTRIBUTE と ESS_TTYPE_ATTRCALC は、次元が属性次元であることを示します。
- ESS_DIMENSIONINFO_T 構造体の DimDataType フィールドは、属性次元のタイプを示します。

戻り値

正常終了の場合、次元の情報構造体の配列が戻されます。

例

```
    ESS_FUNC_M
ESS_GetDimensionInfo(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M sts = ESS_STS_NOERR;
 ESS_STR_T MbrName;
 ESS_ULONG_T nDims, ind;
 ESS_PDIMENSIONINFO_T DimInfo = NULL;

 MbrName = "Year";
 sts = EssGetDimensionInfo(hCtx, MbrName, &nDims,
    &DimInfo);

 if(!sts && DimInfo)
 {
printf("-------- Dimension Information --------\r\n\r\n");
   for(ind = 0; ind < nDims; ind++)
   {
printf("Dimension Name: %s\r\n",
    DimInfo[ind].DimName);
```

```
printf("Dimension Number: %d\r\n",
    DimInfo[ind].DimNumber);

     switch (DimInfo[ind].DimType)
     {
      case ESS_DIMTYPE_DENSE:
printf("Dimension Type: %s\r\n","DENSE");
       break;
      default:
printf("Dimension Type: %s\r\n","SPARSE");
       break;
     }
printf("\r\n");
   }
    EssFree(hInst, DimInfo);
  }
  return (sts);
}
```

関連トピック

- EssBuildDimension
- EssGetApplicationInfo
- EssGetApplicationInfoEx
- EssGetDatabaseInfo
- EssGetDatabaseInfoEx

EssGetDrillThruURL

アクティブなデータベース・アウトライン内のドリルスルー URL を取得します。

構文

```
ESS_FUNC_M EssGetDrillThruURL (
hCtx, URLName, &pUrl
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
URLName	ESS_STR_T	ドリルスルー URL 名。
pUrl	ESS_PDURLINFO_T	URL 定義。

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内のドリルスルー URL が取得されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESS_PRIV_READ)を持っている必要があります。
- 呼出し側は EssSetActive を使用して、指定したデータベースをアクティブ・データベースとして選択しておく必要があります。

例

```
    static void DisplayUrlDefn (ESS_PDURLINFO_T pUrls )
{
 ESS_UINT_T    i;

 printf("\tUrlname    : %s\n", pUrls->cpURLName );
 if (pUrls->bIsLevel0)
  printf("\tUrl Is Level-0 slice : Yes\n");
 else
  printf("\tUrl Is Level-0 slice : No\n");

 printf("\tUrlXmlsize   : %i\n", pUrls->iURLXmlSize );
 printf("\tUrlXml    : %s\n", (ESS_STR_T) pUrls->cpURLXml);

 printf("\tNumber of drill region(s): %d\n", pUrls->iCountOfDrillRegions);
 for ( i = 0; i < pUrls->iCountOfDrillRegions; i++ )
 {
 printf("\t\tDrillRegion[%d]: %s\n", i, pUrls->cppDrillRegions[i] );
 }
 printf("\n");
}
ESS_STS_T sts = ESS_STS_NOERR;
ESS_STR_T urlName = "";
ESS_USHORT_T usCountOfURLs, i;
ESS_PDURLINFO_T urlInfo;

/* Valid case*/

urlName = "Drill Through to EPMI";
sts = EssGetDrillThruURL(hCtx, urlName, &urlinfo);
printf("EssGetDrillThruURL sts: %ld\n",sts);
if(!sts)
 DisplayUrlDefn(urlInfo);

EssFreeStructure (hInst, ESS_DT_STRUCT_URLINFO, 1, (ESS_PVOID_T)urlInfo);
```

EssGetEssbaseSecurityMode

使用されるネイティブまたは Shared Services モードのいずれかのセキュリティのタイプを表示します。

構文

```
     ESS_FUNC_M EssGetEssbaseSecurityMode (
```

```
hCtx
,
pMode
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
pMode	ESS_PSECURITY_MODE_T	使用中のセキュリティのタイプを受け取る変数のアドレス。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    /*
ESS_FUNC_M EssGetEssbaseSecurityMode (ESS_HCTX_T hCtx,
                 ESS_PSECURITY_MODE_T mode);
*/
ESS_FUNC_M ESS_SS_GetEssbaseSecurityMode(ESS_HCTX_T  hCtx)
{
 ESS_STS_T      sts = ESS_STS_NOERR;
 ESS_SECURITY_MODE_T mode;

 sts = EssGetEssbaseSecurityMode(hCtx, &mode);

 if(sts)
 {
  printf("Failed to get Essbase Security mode.\n");
 }
 else
 {
  printf("Essbase Security Mode      : %d\n", mode);
 }
 return(sts);
}
```

拡張された付録 B も参照してください

EssGetExtUser

外部認証ユーザーに関する情報を戻します。

構文

```
ESS_FUNC_M EssGetExtUser (hCtx, UserName, ppExtUserInfo);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
UserName	ESS_STR_T	ユーザーの名前。
ppExtUserInfo	207 ページの「ESS_USERINFOEX_T」	割り当てられたユーザー情報構造体を受け取るポインタのアドレス。

備考

ppExtUserInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー情報構造体が ppExtUserInfo に戻されます。

アクセス

この関数を使用するには、独自のユーザー情報を取得する場合を除き、呼出し元はログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

関連トピック

- EssListExtUsers
- EssSetExtUser
- 207 ページの「ESS_USERINFOEX_T」

EssGetFilter

フィルタのコンテンツの取得を開始します。

構文

```
ESS_FUNC_M EssGetFilter (
hCtx, AppName, DbName, FilterName, pActive, pAccess
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
FilterName	ESS_STR_T	フィルタ名。
pActive	ESS_PBOOL_T	フィルタのアクティブ・フラグを受け取る変数のアドレス。TRUE の場合、フィルタは指定されたデータベースに対して現在有効です。
pAccess	ESS_PACCESS_T	デフォルトのフィルタ・アクセス・レベルを受け取る変数のアドレス。可能な値については、102 ページの「ビットマスク・データ型(C)」に関する説明を参照してください。

備考

この呼出しの後に EssGetFilterRow を続けて呼び出して、フィルタの行をフェッチする必要があります。

戻り値

正常終了の場合、フィルタのアクティブ・フラグが pActive に、そしてデフォルトのフィルタ・アクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetFilter (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FilterName;
 ESS_BOOL_T    Active;
 ESS_ACCESS_T   Access;
 ESS_STR_T    RowString = NULL;
 ESS_STR_T    Acc_Str;

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "Test";

 /**************
  * Get Filter *
  **************/
 sts = EssGetFilter(hCtx, AppName, DbName,
    FilterName, &Active, &Access);
 /*******************
  * Get Filter Rows *
  *******************/
 if(!sts)
 {
  sts = EssGetFilterRow(hCtx, &RowString,
     &Access);
  if(!sts && RowString)
  {
   printf("%s Filter Rows\r\n",FilterName);
   while(RowString)
   {
    switch (Access)
    {
     case ESS_ACCESS_NONE:
      Acc_Str = "NONE";
      break;
     case ESS_ACCESS_READ:
      Acc_Str = "READ";
```

```
        break;
       default:
        Acc_Str = "WRITE";
        break;
      }

     printf("%s - %s\r\n",Acc_Str,RowString);
     sts = EssGetFilterRow(hCtx, &RowString,
        &Access);
    }
    EssFree(hInst, RowString);
   }
  }
  return (sts);
}
```

関連トピック

- EssGetFilterRow
- EssListFilters
- EssSetFilter

EssGetFilterList

フィルタを割り当てられたユーザーのリストを取得します。

構文

```
ESS_FUNC_M EssGetFilterList (
hCtx, AppName, DbName, FilterName, pCount, ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
FilterName	ESS_STR_T	フィルタ名。
pCount	ESS_PUSHORT_T	このフィルタに割り当てられているユーザーのカウントを受け取る変数のアドレス。
ppUserList	ESS_PPUSERNAME_T	割り当てられたユーザー名の配列を受け取るポインタのアドレス。

備考

ppUserList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、このフィルタに割り当てられているユーザーのカウントが pCount に、ユーザー名の配列が ppUserList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_STS_T
ESS_GetFilterList (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FilterName;
 ESS_USHORT_T   Count = 0;
 ESS_USHORT_T   ind;
 ESS_PUSERNAME_T UserList = NULL;

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "NewFilter";

 sts = EssGetFilterList(hCtx, AppName, DbName,
    FilterName, &Count, &UserList);
 if(!sts)
 {
printf("--------%s User List---------\r\n\r\n",
       FilterName);
   if(Count && UserList)
   {
    for (ind = 0; ind < Count; ind++)
     printf("%s\r\n",UserList[ind]);
    EssFree(hInst, UserList);
   }
   printf("\r\n");
 }
 return (sts);
}
```

関連トピック

- EssGetFilter
- EssListFilters
- EssSetFilterList

EssGetFilterRow

フィルタの次の行を取得します。

構文

```
ESS_FUNC_M EssGetFilterRow (
hCtx, pRowString, pAccess
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRowString	ESS_PSTR_T	このフィルタの次の行を受け取るポインタのアドレス。
pAccess	ESS_PACCESS_T	フィルタ行のアクセス・レベルを受け取る変数のアドレス。可能な値は、102 ページの「ビットマスク・データ型(C)」に関する説明にリストされています。

備考

- この関数は、EssGetFilter を呼び出した後に、NULL 行文字列ポインタが戻されるまで繰り返し呼び出す必要があります。
- pRowString に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合は、次のフィルタ行(ある場合)が pRowString に、行アクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

EssGetFilter の例を参照してください。

関連トピック

- EssGetFilter
- EssListFilters

EssGetGlobalState

システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を取得します。

構文

```
ESS_FUNC_M EssGetGlobalState (
hCtx, ppGlobal
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ppGlobal	147 ページの「ESS_GLOBAL_T」	割り当てられたグローバル状態構造体を受け取るポインタのアドレス。

備考

ppGlobal に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、サーバーのグローバル状態構造体の現在の状態が ppGlobal に戻されます。

アクセス

この関数を使用するには、呼出し元がスーパーバイザである必要があります。

例

```
    ESS_FUNC_M
ESS_GetGlobalState (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_PGLOBAL_T  pGlobal = NULL;

 sts = EssGetGlobalState(hCtx, &pGlobal);

 if(!sts && pGlobal)
 {
  printf("--------- Global State ----------\n\n");
  printf("Security->%d Logins->%d\r\n",
      pGlobal->Security, pGlobal->Logins);
  printf("Access->%ld Validity->%d\r\n",
      pGlobal->Access, pGlobal->Validity);
  printf("Currency->%d PwMin->%d\r\n",
      pGlobal->Currency,pGlobal->PwMin);
  printf("InactivityTime->%ld InactivityCheck->%ld\r\n", pGlobal->InactivityTime,
pGlobal->InactivityCheck);

   EssFree(hInst, pGlobal);
 }
 return (sts);
}
```

関連トピック

- EssSetGlobalState

EssGetGroup

グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。

構文

```
ESS_FUNC_M EssGetGroup (
hCtx, GroupName, ppGroupInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	グループ名。
ppGroupInfo	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	割り当てられたグループ情報構造体を受け取るポインタのアドレス(出力)。

備考

ppGroupInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、グループ情報構造体が ppGroupInfo に戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

関連トピック

- EssGetGroupInfoEx
- EssListGroups
- EssSetGroup

EssGetGroupInfoEx

グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。EssGetGroup に似ていますが、ユーザー・ディレクトリの指定、または GroupID の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssGetGroupInfoEx (
hCtx
,
GroupId
,
bisIdentity
,
ppGroupInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupId	ESS_STR_T	グループ名(入力)。groupname@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。GroupId が名前か ID かを示します。TRUE の場合、GroupId は ID です。
ppGroupInfo	ESS_PGROUPINFOID_T	割り当てられたグループ情報構造体を受け取るポインタのアドレス(出力)。グループ・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

ppGroupInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、グループ情報構造体が ppGroupInfo に戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
void DisplayGroupsInfoEx(ESS_GROUPINFOID_T groupInfo)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_BOOL_T isDefined = ESS_TRUE;

 printf("\tUser Name: %s\n", groupInfo.Name);
 printf("\tProvider Name: %s\n", groupInfo.ProviderName);
 printf("\tIdentity: %s\n", groupInfo.connparam);
 printf("\tDescription: %s\n", groupInfo.Description);
 printf("\tEMail Identification: %s\n", groupInfo.EMailID);

 if (groupInfo.LockedOut)
  printf("\tLocked out: Yes\n");
 else
  printf("\tLocked out: No\n");

 if (groupInfo.PwdChgNow)
  printf("\tChange the password now: Yes\n");
 else
  printf("\tChange the password now: No\n");

 printf("\tPassword: %s\n", groupInfo.Password);
 printf("\tApplication: %s\n", groupInfo.AppName);
 printf("\tDatabase: %s\n", groupInfo.DbName);

 if (groupInfo.Login)
  printf("\tLogged in: Yes\n");
```

```
else
 printf("\tLogged in: No\n");

switch(groupInfo.Access)
{
 case ESS_ACCESS_ADMIN:
  printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", groupInfo.Access);
  break;
 case ESS_ACCESS_APPALL:
  printf("\tAccess: %d - ESS_ACCESS_APPALL\n", groupInfo.Access);
  break;
 case ESS_ACCESS_DBALL:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
  break;
 case ESS_ACCESS_APPCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
  break;
 case ESS_ACCESS_APPMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", groupInfo.Access);
  break;
 case ESS_ACCESS_DBCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", groupInfo.Access);
  break;
 case ESS_ACCESS_DBMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", groupInfo.Access);
  break;
 case ESS_ACCESS_CALC:
  printf("\tAccess: %d - ESS_ACCESS_CALC\n", groupInfo.Access);
  break;
 case ESS_ACCESS_WRITE:
  printf("\tAccess: %d - ESS_ACCESS_WRITE\n", groupInfo.Access);
  break;
 case ESS_ACCESS_READ:
  printf("\tAccess: %d - ESS_ACCESS_READ\n", groupInfo.Access);
  break;
 case ESS_PRIV_USERCREATE:
  printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", groupInfo.Access);
  break;
 case ESS_PRIV_APPCREATE:
  printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", groupInfo.Access);
  break;
 case ESS_PRIV_APPMANAGE:
  printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", groupInfo.Access);
  break;
 case ESS_PRIV_APPLOAD:
  printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", groupInfo.Access);
  break;
 case ESS_PRIV_DBCREATE:
  printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", groupInfo.Access);
  break;
 case ESS_PRIV_DBMANAGE:
  printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", groupInfo.Access);
  break;
 case ESS_PRIV_DBLOAD:
  printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", groupInfo.Access);
  break;
 case ESS_PRIV_CALC:
```

```
   printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
   break;
  case ESS_PRIV_WRITE:
   printf("\tAccess: %d - ESS_PRIV_WRITE\n", groupInfo.Access);
   break;
  case ESS_PRIV_READ:
   printf("\tAccess: %d - ESS_PRIV_READ\n", groupInfo.Access);
   break;
  case ESS_PRIV_NONE:
   printf("\tAccess: %d - ESS_PRIV_NONE\n", groupInfo.Access);
   break;
  default:
   printf("\tAccess: Unknown\n");
}

switch(groupInfo.MaxAccess)
{
  case ESS_ACCESS_ADMIN:
   printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPALL:
   printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBALL:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_CALC:
   printf("\tMax Access: %d - ESS_ACCESS_CALC\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_WRITE:
   printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_READ:
   printf("\tMax Access: %d - ESS_ACCESS_READ\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_USERCREATE:
   printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPCREATE:
   printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPMANAGE:
   printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPLOAD:
```

```
    printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_DBCREATE:
    printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_DBMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", groupInfo.MaxAccess);
    break;
   default:
    printf("\tMax Access: Unknown\n");
  }

  printf("\tPassword Expiration in Dates: %d\n",groupInfo.Expiration);
  printf("\tFailed Login Attempts Since Then: %d\n", groupInfo.FailCount);
  printf("\tLogin ID: %d\n", groupInfo.LoginId);
  printf("\tProtocol: %s\n", groupInfo.protocol);
  printf("\tConnection Parameter: %s\n", groupInfo.connparam);
  printf( "\n");

}


ESS_FUNC_M ESS_GetGroupInfoEx (ESS_HCTX_T hCtx)
{
  ESS_STS_T sts  = ESS_STS_NOERR;
  ESS_STR_T groupId;
  ESS_BOOL_T bisIdentity;
  ESS_PGROUPINFOID_T groupInfo;

  groupId = "IDAdminGroup@ldap";
  bisIdentity = ESS_TRUE;
  sts = EssGetGroupInfoEx(hCtx, groupId, bisIdentity, &groupInfo);
  printf("EssGetGroupInfoEx sts: %ld\n", sts);
  if(!sts && groupInfo)
  {
   DisplayGroupsInfoEx(*groupInfo);
  }

  return (sts);
}
```

関連トピック

- EssListGroupsInfoEx

EssGetGroupList

グループのメンバーであるユーザーのリスト(またはユーザーが属するグループのリスト)を取得します。

構文

```
ESS_FUNC_M EssGetGroupList (
hCtx, GroupName, pCount, ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_USERNAME_T	ユーザー名またはグループ名。
pCount	ESS_PUSHORT_T	ユーザー名のカウントを受け取る変数のアドレス。
ppUserList	ESS_PPUSERNAME_T	割り当てられたユーザー名文字列の配列を受け取るポインタのアドレス。

備考

- この関数を使用すると、ユーザー名を GroupName 引数として使用することによって、ユーザーが属するグループのリストを取得することもできます。
- ppUserList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、ユーザー名のカウントが pCount に、ユーザー名文字列の配列が ppUserList に戻されます。

アクセス

この関数を使用するには、呼出し元が自分自身のグループ・リストを取得するユーザーでないかぎり、ログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ListGroupUsers (ESS_HCTX_T  hCtx,
         ESS_HINST_T hInst
        )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PUSERNAME_T  UserList = NULL;
 ESS_USHORT_T   ind;
 ESS_USHORT_T   Items;
```

```
 ESS_USERNAME_T  GroupName;
 strcpy(Groupname, "PowerUsers");
  sts = EssGetGroupList (hCtx, GroupName, &Items, &UserList);
 if (!sts)
 {
  if (Items && UserList)
  {
   printf ("\r\n-------%s User List-------\r\n\r\n", GroupName);
   for (ind = 0; ind < Items; ind++)
   {
     if (UserList [ind])
      printf ("%s\r\n", UserList [ind]);
   }
   EssFree (hInst, UserList);
  }
  else
   printf ("\r\nUsers list is empty\r\n\r\n");
 }

 return (sts);
}
```

関連トピック

- EssGetGroupListEx
- EssAddToGroup
- EssDeleteFromGroup
- EssListGroups
- EssSetGroupList

EssGetGroupListEx

グループのメンバーであるユーザーのリストまたはユーザーが属するグループのリストを取得します。EssGetGroupList に似ていますが、ユーザー・ディレクトリの指定、または GroupName の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssGetGroupListEx (
hCtx
,
GroupName
,
bIsIdentity
, entityType,
pCount
,
bOutputInIds
,
ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
GroupName	ESS_STR_T	グループ名または ID（入力)。groupname@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。GroupName が名前か ID かを示します。TRUE の場合、GroupName は ID です。
entityType	ESS_USHORT_T	エンティティのタイプ(入力)。次のいずれかになります: ● ESS_TYPE_USER - このユーザーが属するグループのリストを戻します ● ESS_TYPE_GROUP - このグループが属するユーザーのリストを戻します
pCount	ESS_PUSHORT_T	ユーザー名のカウントを受け取る変数のアドレス(出力)。
bOutputInIds	ESS_BOOL_T	入力。出力を ID にする必要があるかどうかを示します。TRUE の場合、ppUserList に ID の配列が戻されます。
ppUserList	ESS_PSTR_T	割り当てられたユーザー名文字列または ID の配列を受け取るポインタのアドレス(出力)。

備考

- この関数を使用すると、ユーザー名を GroupName 引数として使用することによって、ユーザーが属するグループのリストを取得することもできます。
- ppUserList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、ユーザー名のカウントが pCount に、ユーザー名文字列または ID の配列が ppUserList に戻されます。

アクセス

この関数を使用するには、呼出し元が自分自身のグループ・リストを取得するユーザーでないかぎり、ログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
void DisplayUserList(ESS_USHORT_T count, ESS_PSTR_T UserList)
{
    ESS_USHORT_T i;

    for (i = 0; i < count; i++)
    {
        if (UserList [i])
            printf ("%s\n", UserList[i]);
    }
}
```

```
ESS_FUNC_M ESS_ListGroupUsers (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T groupId;
 ESS_BOOL_T bisIdentity;
 ESS_USHORT_T type;
 ESS_USHORT_T count;
 ESS_BOOL_T bUsingIdentity;
 ESS_PSTR_T pUserList;

 groupId = "IDAdminGroup";
 bisIdentity = ESS_TRUE;
 type = ESS_TYPE_GROUP;
 sts = EssGetGroupListEx(hCtx, groupId, bisIdentity, type, &count, &bUsingIdentity,
&pUserList);
 printf("EssGetGroupListEx sts: %ld\n", sts);
 if(!sts)
 {
  if(pUserList)
  {
   printf ("\n---User/Group list for %s:\n", groupId);
   DisplayUserList(count, pUserList);
  }
  else
   printf ("\tUser list is empty\n");
 }

 return (sts);
}
```

関連トピック

- EssAddToGroupEx
- EssDeleteFromGroupEx
- EssListGroupsInfoEx

EssGetIBH

無効なブロック・ヘッダーを含むブロックのすべてのインデックス組合せを持つローカル・ログ・ファイルを作成します。データベース管理者はこの情報を使用して、破損したと特定されたデータポイントをリロードできます。

構文

```
ESS_FUNC_M EssGetIBH (
hCtx
,
destFileName
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
destFileName;	ESS_STR_T	クライアント側で IBH 情報を保管するファイル名。

関連トピック

- EssLocateIBH
- EssFixIBH

EssGetLocalPath

クライアント上にある特定のオブジェクト・ファイルの完全なローカル・ファイル・パスを取得します。

構文

```
ESS_FUNC_M EssGetLocalPath (
hCtx, ObjType, AppName, DbName, ObjName, Create, pPath
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	EssCreateLocalContext で戻された API コンテキスト・ハンドル。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。オブジェクト・タイプのリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名または NULL (ESS_NULL)。NULL の場合は、この関数がファイル名を想定し、pPath に ObjName をそのまま戻します。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	オブジェクト名またはファイル名。AppName が NULL の場合は、ObjName が正しいかどうか解析されません。パスに接尾辞は追加されません。
Create	ESS_BOOL_T	ディレクトリ・フラグの作成。TRUE の場合は、必要に応じて適切なアプリケーションとデータベース・サブディレクトリが作成されます。FALSE の場合でディレクトリが存在しない場合は、エラーが発生します。
pPath	ESS_PSTR_T	割り当てられたローカル・パス名文字列を受け取るポインタのアドレス。

備考

pPath に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、適切なオブジェクト・ファイルのフル・パス名が pPath に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_VOID_T
ESS_GetLocalPath (ESS_HINST_T hInst)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_HCTX_T     hLocalCtx;
 ESS_STR_T      AppName;
 ESS_STR_T      DbName;
 ESS_STR_T      ObjName;
 ESS_OBJTYPE_T   ObjType;
 ESS_BOOL_T     Create;
 ESS_STR_T      Path;

 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Basic";
 ObjType = ESS_OBJTYPE_OUTLINE;
 Create  = ESS_TRUE;
 sts = EssCreateLocalContext(hInst, NULL, NULL,
   &hLocalCtx);
 if(!sts && hLocalCtx)
 {
  sts = EssGetLocalPath(hLocalCtx, ObjType,
   AppName, DbName, ObjName, Create, &Path);
  if(!sts)
  {
   if(*Path)
   {
    printf("Path: %s\r\n",Path);
    EssFree(hInst,Path);
   }
  }
 }

 if(hLocalCtx)
  sts = EssDeleteLocalContext(hLocalCtx);
}
```

関連トピック

- EssCreateLocalContext
- EssListObjects

EssGetLocationAliasList

現在定義されているすべてのロケーション別名のリストを戻します。同時に、そのロケーション別名がマッピングされているホスト名、アプリケーション名、データベース名、およびユーザー名のリストも戻します。

構文

```
     ESS_FUNC_M EssGetLocationAliasList (
```

```
hCtx
,
pusListCnt
,
ppAliasNames
,
ppHostNames
,
ppAppNames
,
ppDbNames
,
ppUserNames
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
pusListCnt;	ESS_PUSHORT_T	戻されたロケーション別名の数。
ppAliasNames;	ESS_PSTR_T *	ロケーション別名バッファ。
ppHostNames;	ESS_PSTR_T *	ホスト名バッファ。
ppAppNames;	ESS_PSTR_T *	アプリケーション名バッファ。
ppDbNames;	ESS_PSTR_T *	データベース名バッファ。
ppUserNames;	ESS_PSTR_T *	ユーザー・ログイン名バッファ。

備考

- hCtx は入力専用パラメータです。
- pusListCnt、ppAliasNames、ppHostNames、ppAppNames、ppDbNames および ppUserNames は出力パラメータで、値が戻されます。
- この関数を呼び出した後は、EssFree を呼び出して、戻されたリストによって使用されていたメモリーを解放する必要があります。

関連トピック

- EssCreateLocationAlias
- EssDeleteLocationAlias

EssGetLogFile

アプリケーション・ログ・ファイル(appname.log)または Essbase サーバー・ログ・ファイル(essbase.log)の全体または一部を、サーバーからクライアントにコピーします。

構文

```
ESS_FUNC_M EssGetLogFile (
```

```
hCtx, AppName, TimeStamp, LocalName
};
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名または NULL。NULL の場合、この関数は Essbase サーバー・ログ・ファイル(essbase.log)にアクセスします。
TimeStamp	ESS_TIME_T	必要な最も古いログ・ファイル・エントリの日付と時刻を示すタイム・スタンプ。TimeStamp が 0 に設定されている場合、この関数はログ・ファイル全体をコピーします。
LocalName	ESS_STR_T	クライアント上のローカル・コピー先ファイルのフル・パス名。

備考

- TimeStamp は、グリニッジ標準時の 1970 年 1 月 1 日の午前 0 時(00:00:00)からの経過秒数を示します。この関数は、TimeStamp で指定された日付と時間の後に発生したログ・ファイル・エントリのみをクライアントにコピーします。
- essbase.log および appname.log の場所は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

正常終了の場合は、LocalName で指定されたローカル・ファイルにファイルがコピーされます。

アクセス

この関数を使用するには、呼出し元が、指定されたアプリケーションまたはその任意のデータベースに対する、アプリケーション・デザイン権限(ESS_PRIV_APPDESIGN)またはデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetLogFile (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;
 ESS_TIME_T   TimeStamp;
 ESS_STR_T   LocalName;

 AppName  = "Sample";
 LocalName = "C:\\Hyperion\\products\\Essbase\\EssbaseServer\\test.log";

 /* Get entire log file */
 TimeStamp = 0;

 sts = EssGetLogFile(hCtx, AppName, TimeStamp,
    LocalName);
 return (sts);
```

```
}
```

関連トピック

- EssDeleteLogFile
- EssLogSize
- EssWriteToLogFile

EssGetMemberCalc

アクティブ・データベース・アウトライン内の、特定のメンバーの計算式を取得します。

構文

```
ESS_FUNC_M EssGetMemberCalc (
hCtx, MbrName, pCalcStr, pLastCalcStr
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
MbrName	ESS_STR_T	メンバー名。
pCalcStr	ESS_PSTR_T	割り当てられたメンバー計算文字列を受け取るポインタのアドレス。
pLastCalcStr	ESS_PSTR_T	割り当てられたメンバーの最終計算文字列を受け取るポインタのアドレス。

備考

- 最後の計算文字列は、データベースを最後に計算したときにメンバーを計算するために使用した式です。計算スクリプトを使用してデータベースの計算を行った場合は、pCalStr が残る場合があります。
- この関数は、リレーショナル・スパンのブール式が設定されているかどうかを確認し、添付されたリレーショナル・データ・セットに格納されているメンバーが計算文字列を持っているかどうかを判断できますが、計算文字列のかわりに NULL 文字列を戻します。
- pCalcStr と pLastCalcStr に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合は、計算文字列と最終計算文字列が pCalcStr と pLastCalcStr に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetMbrCalc (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   calcStr, lastCalcStr;

 calcStr = lastCalcStr = NULL;
sts = EssGetMemberCalc(hCtx, "Year", &calcStr, &lastCalcStr);
 if (!sts)
 {
  if (calcStr)
  {
   printf ("Outline Defined Calc Equation -- [%s]\r\n", calcStr);
  }
else
  {
   printf ("Outline Defined Calc Equation -- [Default Rollup]\r\n");
  }

  if (lastCalcStr)
  {
   printf ("Last Calculated Calc Equation -- [%s]\r\n", lastCalcStr);
  }
  else
  {
   if (calcStr)
    printf ("Last Calculated Calc Equation -- [%s]\r\n", calcStr);
   else
    printf ("Last Calculated Calc Equation -- [Default Rollup]\r\n");
  }

 }
 if (calcStr)
  EssFree (hInst, calcStr);
 if (lastCalcStr)
  EssFree (hInst, lastCalcStr);

 return (sts);
}
```

関連トピック

- EssGetMemberInfo
- EssSetActive

EssGetMemberInfo

アクティブ・データベース・アウトライン内の、特定のメンバーに関する情報が含まれている構造体を取得します。

構文

```
ESS_FUNC_M EssGetMemberInfo (
hCtx, MbrName, ppMbrInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
MbrName	ESS_STR_T	メンバー名。
ppMbrInfo	157 ページの「ESS_MEMBERINFO_T」	割り当てられたメンバー情報構造体を受け取るポインタのアドレス。

備考

- EssFree または EssFreeStructure を呼び出して、ppMbrInfo に動的に割り当てられたメモリーを解放します。

  タイプ ESS_ATTRMBRDT_STRING の属性メンバーの場合は、EssFreeStructure を呼び出して ppMbrInfo に動的に割り当てられたメモリーを解放する必要があります。ESS_DT_STRUCT_MBRINFO を構造体 ID として指定します。EssFree は、属性の文字列の値に動的に割り当てられたメモリーを解放しません。
- 157 ページの「ESS_MEMBERINFO_T」構造体の Status フィールドの ESS_MBRSTS_ATTRIBUTE 定数は、次元またはメンバーが属性次元または属性メンバーであることを示します。
- ESS_MEMBERINFO_T 構造体の次の 2 つのフィールドは、属性のみです:
  - fAttributed
  - Attribute
- この関数は、リレーショナル・スパンのブール式が設定されており (EssSetSpanRelationalPartition で設定)、リレーショナル・ストア内のメンバーに関する情報を戻すことができるかどうかを確認します。
- 157 ページの「ESS_MEMBERINFO_T」構造体の次の 2 つのフィールドは、リレーショナル・ストア内のメンバーにのみ使用されます:
  - fHasRelDesc
  - fHasRelPartEnabled
- 768 ページの「ESS_MBRINFO_T」構造体の次の 2 つのフィールドは、リレーショナル・ストア内のメンバーにのみ使用されます:
  - fHasRelDesc
  - fHasRelPartEnabled

戻り値

正常終了の場合は、この関数は割り当てられたメンバー情報構造体である ppMbrInfo を戻します。メンバーの親がない場合は、この関数は ESS_MEMBERINFO_T 構造体の ParentMbrName フィールドに空の文字列を戻します。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_GetMbrInfo (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
       )
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_MEMBERINFO_T  *pMbrInfo = NULL;
 sts = EssGetMemberInfo(hCtx, "Profit",
       &pMbrInfo);
 if (!sts)
 {
  if (pMbrInfo)
  {
   EssFreeStructure(hCtx, structId, count, structPtr);
  }
 }
 return (sts);
}
```

関連トピック

- EssCheckMemberName
- EssFreeStructure
- EssGetMemberCalc
- EssQueryDatabaseMembers
- EssSetActive

EssGetObject

サーバーまたはクライアントのオブジェクト・システムからローカル・ファイルにオブジェクトをコピーし、オプションでロックします。

構文

```
ESS_FUNC_M EssGetObject (
hCtx, ObjType, AppName, DbName, ObjName, LocalName, Lock
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。

パラメータ	データ型	説明
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	取得するオブジェクト名。
LocalName	ESS_STR_T	クライアント上のローカル・コピー先ファイルのフル・パス名。
Lock	ESS_BOOL_T	オブジェクトのロックを制御するフラグ。TRUE の場合は、オブジェクトがロックされ、他のユーザーによる更新を防止します。

備考

オブジェクトをロックするには、そのオブジェクトがサーバーに存在している必要があり、かつ他のユーザーによってロックされていないことが必要です。クライアント上でのロックはサポートされていません。

戻り値

正常終了の場合は、オブジェクトは LocalName で指定されているローカル・ファイルにコピーされます。

アクセス

この関数を使用するには、呼出し元はオブジェクトが含まれている指定したアプリケーション、データベースのいずれか、またはその両方に対して、適切なレベルのアクセス権(オブジェクト・タイプにより異なる)を持っている必要があります。オブジェクトをロックするには(lock フラグが TRUE)、呼出し元はオブジェクトを含む指定のアプリケーションまたはデータベースに対するアプリケーションまたはデータベース・デザイナ権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)が必要です。

例

```
    ESS_FUNC_M
ESS_GetObject (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     ObjName;
 ESS_OBJTYPE_T   ObjType;
 ESS_STR_T     LocalName;
 ESS_BOOL_T     Lock;


 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Basic";
 ObjType = ESS_OBJTYPE_OUTLINE;
 LocalName = "C:\\Hyperion\\products\\Essbase\\EssbaseClient\\client\\Basic.otl";
 Lock   = ESS_TRUE;

 sts = EssGetObject (hCtx, ObjType, AppName,
```

```
    DbName, ObjName, LocalName, Lock);
 return (sts);
}
```

関連トピック

- EssGetObjectInfo
- EssListObjects
- EssLockObject
- EssPutObject
- EssUnlockObject

EssGetObjectInfo

サーバー上またはローカルのクライアント上にある特定のオブジェクトに関する情報を取得します。

構文

```
ESS_FUNC_M EssGetObjectInfo (
hCtx, ObjType, AppName, DbName, ObjName, ppObject
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	オブジェクト名。
ppObject	ESS_PPOBJINFO_T	割り当てられたオブジェクト情報構造体を受け取るポインタのアドレス。

備考

ppObject に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、適切なオブジェクトに関する情報を含むオブジェクト構造体が ppObject に戻されます。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベース(あるいはその両方)に対して、呼出し元が(オブジェクト・タイプに応じて)適切なレベルのアクセス権を持っている必要があります。

関連トピック

- EssGetObject
- EssListObjects

EssGetProcessState

計算またはデータ・インポートなどの、非同期プロセスの現在の状態を取得します。

構文

```
ESS_FUNC_M EssGetProcessState (
hCtx, pProcState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pProcState	188 ページの「ESS_PROCSTATE_T」	プロセス状態構造体へのポインタ。

備考

- pProcState に ESS_STATE_DONE が戻されるまで、この関数を定期的に(5-10 秒間隔)呼び出す必要があります。
- 非同期データベース操作(たとえば計算)が正しく開始される前にこの関数を呼び出すと、エラーが発生します。
- pProcState に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

この関数がプロセスの状態を取得できない場合は、エラーが戻されます。エラーが原因で処理が終了すると、そのエラー・コードが戻されます。それ以外の場合は、ESS_STS_NOERR が戻され、状態構造体 pProcState に現在のプロセスの状態が示されます。pProcState の値:

- ESS_STATE_DONE - 0 =完了
- ESS_STATE_INPROGRESS - 1 = 進行中
- ESS_STATE_FINALSTAGE - 5 = 最終段階。取消しできません

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_RunCalc (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_SHORT_T   isResponse;
 ESS_HCTX_T  hSrcCtx;
 ESS_BOOL_T  isObject = ESS_FALSE;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;
 ESS_PROCSTATE_T pState;

 hSrcCtx  = hCtx;
 AppName  = "Sample";
 DbName   = "Basic";
 FileName = "Test";

 sts = EssCalcFile (hCtx, hSrcCtx, AppName,
    DbName, FileName, ESS_TRUE);
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while(!sts && (pState.State !=
     ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 return(sts);
}
```

関連トピック

- EssBeginCalc
- EssCalc
- EssCancelProcess
- EssImport

EssGetRuntimeSubVars

この関数は、計算スクリプトが実行されるクライアントへのインタフェースとして実装されます。この関数は、指定された計算スクリプトの SET RUNTIMESUBVARS 計算コマンドにおけるランタイム代替変数の宣言で指定済のすべての情報(名前、値および説明)を取得します。ランタイム代替変数の宣言に <RTSV_HINT>rtsv_description</RTSV_HINT>タグが含まれている場合 (rtsv_description は、ランタイム代替変数のデータ型とデータ入力制限を表す文字列)、ランタイムでユーザーに値の入力を求めるために、または計算スクリプトに値が渡される前に入力データを検証するために、この文字列を使用できます。

構文

```
     ESS_FUNC_M EssGetRuntimeSubVars (
```

```
hCtx
,
hSrcCtx

AppName
,
DbName
,
FileName
,
pulRtSVCount
,
pRtSVList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	計算スクリプト・ファイルの場所の API コンテキスト・ハンドル。計算スクリプト・ファイルは、クライアント・コンピュータ上またはターゲット・データベースと同じ Essbase サーバー上に存在することができます。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	アプリケーションのデータベースの名前。
CalcScript	ESS_STR_T	計算スクリプト・ファイルの名前または計算文字列。 この引数は、bClientCalcFile 引数とともに使用されます。
bClientCalcFile	ESS_BOOL_T	CalcScript 引数がクライアント側の計算スクリプト・ファイルを参照するか(TRUE)、計算文字列を参照するか(FALSE)を示すフラグ。 計算スクリプト・ファイルが Essbase サーバー上にある場合、この引数は無視されます。
pulRtSVCount	ESS_ULONG_T	計算スクリプトのランタイム代替変数構造体(キー/値ペア)のカウントを取得するための変数のアドレス。
pRtSVList	ESS_RUNTIMESUBVARS_DESC_T	計算スクリプトのランタイム代替変数構造体のリスト(配列)。各構造体には、ランタイム代替変数のキー/値ペアが含まれます。オプションで、ランタイム代替変数のデータ型とデータ入力制限(たとえば、100 以下の整数)を表す<RTSV_HINT>rtsv_description</RTSV_HINT>タグの文字列を各構造体で指定できます。 API は内部的に配列を割当て、API 呼出し元はメモリーを解放します。

戻り値

なし。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    void Ess_GetRuntimeSubVars(ESS_HINST_T hInst, ESS_HCTX_T hCtx)
{
 ESS_STS_T    sts;
 ESS_STR_T AppName = "Sample";
 ESS_STR_T DbName = "Basic";
 ESS_STR_T FileName = "D:\\temp\\testrt.csc"; //Client side calc file
 //ESS_STR_T FileName = "SET RUNTIMESUBVARS { myCOGS; myProduct = 333; myMarket =
<RTSV_HINT> myMarket is an initialized Runtime Substitution Variables </RTSV_HINT>;
myCity=   \"Sunnyvale\", \"Santa Clara\", \"San Jose\" <RTSV_HINT> list of non-
numeric values </RTSV_HINT>; myCOGS; myProduct = 333; product = 100 <RTSV_HINT>
initialize to string 100 </RTSV_HINT>; baseProduct = @LEVMBRS(\"Product\",0)
<RTSV_HINT> list of members</RTSV_HINT>; myCOGS; myProduct = 333; mySales = 777
<RTSV_HINT> uninitialized; mySales should be specified during runtime if there is no
substitution variable mySales set at Essbase level</RTSV_HINT>; myCOGS;myProduct =
333;mySal;};FIX (\"100-10\", \"New York\") COGS=&mySV; Sales = &mySales;
ENDFIX;" ; // Client side calc string
 //ESS_STR_T FileName = "testrt"; \\ server side calc file
 ESS_BOOL_T Calculate = TRUE;
 ESS_ULONG_T  i;
 ESS_ULONG_T* pulRTParamsCount = NULL;
 ESS_RUNTIMESUBVARS_DESC_T* pRTParamList = NULL;
 ESS_RUNTIMESUBVARS_DESC_T** ppRTParamList = NULL;

 ESS_RUNTIMESUBVARS_DESC_T* pRTParams = NULL;

 ESS_STS_T    status;
 ESS_PROCSTATE_T pState;

 ESS_HCTX_T   hLocalCtx = ESS_INVALID_HCTX;
 sts = EssCreateLocalContext (hInst, ESS_NULL, ESS_NULL, &hLocalCtx);

 status = EssAlloc(hInst, sizeof(ESS_ULONG_T), (ESS_PPVOID_T)&pulRTParamsCount);

 memset(pulRTParamsCount, 0, sizeof(ESS_ULONG_T));

 ppRTParamList= &pRTParamList;

 //For server side calc file
 sts = EssGetRuntimeSubVars(hCtx, hCtx, AppName, DbName, FileName, FALSE,
pulRTParamsCount, ppRTParamList);

 //For client side calc file
 sts = EssGetRuntimeSubVars(hCtx, hLocalCtx, NULL, NULL, FileName, TRUE,
pulRTParamsCount, ppRTParamList);

 //For client side calc strings
 sts = EssGetRuntimeSubVars(hCtx, hLocalCtx, NULL, NULL, FileName, FALSE,
pulRTParamsCount, ppRTParamList);

 pRTParams = &pRTParamList[0];
```

```
 for (i=0; i < *pulRTParamsCount; i++)
  {
   printf("***** information for Runtime Parameter - %d *****\n", i+1);

   printf("  Param Name  -  %s \n", (pRTParams+i)->rtsvName);
   printf("  Param Value -  %s \n", (pRTParams+i)->rtsvVal);
   printf("  RTP_HINT   -  %s \n", (pRTParams+i)->rtsvDesc);

   printf("\n");
  }

 if(sts)
  printf("API could not be executed.");

 if (pulRTParamsCount)
  EssFree(hInst, pulRTParamsCount);

 return(status);
}
```

関連トピック

- ESS_RUNTIMESUBVARS_DESC_T
- EssCalcWithRuntimeSubVars
- EssCalcFileWithRuntimeSubVars

EssGetServerLocaleString

サーバーのロケール記述。たとえば、English_UnitedStates.US-ASCII@Default を取得します。

構文

```
ESS_FUNC_M EssGetServerLocaleString (
hCtx
,
localeString
);
```

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
localeString;	ESS_PSTR_T	サーバーのロケール記述に割り当てられた文字列を受け取るポインタのアドレス。

備考

localeString に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合は、localeString にサーバーのロケール記述が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_GetServerLocaleString (ESS_HCTX_T  hCtx)
{
  ESS_FUNC_M   sts = ESS_STS_NOERR;
  ESS_STR_T localeStr= NULL;

  sts = EssGetServerLocaleString(hCtx, &localeStr);

  if (localeStr)
  {
   printf("server locale: %s\r\n",localeStr);
   EssFree(hInst,localeStr);
  }
  return sts;
}
```

EssGetServerMode

Essbase サーバーが Unicode モードと非 Unicode モードのどちらであるかを示す値を戻します。

構文

```
ESS_FUNC_M EssGetServerMode(
hCtx
, *
bUnicode
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(ログイン済)。
bUnicode	ESS_BOOL_T	戻り値、bUnicode。bUnicode は次のどちらかの値になります: • ESS_TRUE – Essbase サーバーは Unicode モードです。Essbase サーバーは、Unicode モード・アプリケーションの作成、または非 Unicode モードから Unicode モードへのアプリケーションの移行が可能です。 • ESS_FALSE – Essbase サーバーは非 Unicode モードです。Essbase サーバーでは Unicode モードのアプリケーション作成や、非 Unicode モードから Unicode モードへのアプリケーションの移行は行えません。

戻り値

なし。

アクセス

この関数を使用するには、呼出し側が特別な権限を持っている必要はありません。

関連トピック

- EssSetServerMode

EssGetSpoolFile

データベースについての特定のトリガー・ログ・ファイルを戻します。

構文

```
ESS_FUNC_M EssGetSpoolFile (
hCtx, AppName, DbName, SplName, LocalName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
SplName	ESS_STR_T	戻される特定のスプール・ファイル名。
LocalName	ESS_STR_T	サーバーでのスプール・ファイルの新しい名前。

戻り値

正常終了の場合は、データベースについての特定のスプール・ファイルを戻します。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssDisplayTriggers
- EssListSpoolFiles
- EssDeleteAllSplFiles
- EssDeleteSplFile
- EssMdxTrig

EssGetSrvOutlineInfo

Essbase サーバーに保管されたアウトライン情報を取得します。この関数を使用する前にアウトライン・クエリー・モードで開くための要件はありません。

構文

```
ESS_FUNC_M EssGetSrvOutlineInfo (
hCtx
,
AppName
,
DbName
,
pSvrOutlineInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pSvrOutlineInfo	779 ページの「ESS_SVROTLINFO_T」	Essbase サーバーに保管されるアウトライン情報を含む構造体へのポインタ。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_GetSrvOutlineInfo()
{

ESS_STS_T   sts = 0;
ESS_INT_T   i;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_SVROTLINFO_T  SvrOutlineInfo;



memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;
```

```
sts = EssGetSrvOutlineInfo (hCtx, szAppName, szDbName, &SvrOutlineInfo);

if (!sts)
{
  printf("\nCase sensitivity is set to: %d", (SvrOutlineInfo).fCaseSensitive);
  printf("\nOutline type is set to: %d", (SvrOutlineInfo).usOutlineType);
  printf("\nOutline allows duplicate names is set to: %d",
(SvrOutlineInfo).fNonUniqueName);
  printf("\nNumber of alias tables is: %d", (SvrOutlineInfo).usNumAliasTables);
  printf("\nNames of the alias tables are:");

  for (i= 0; i <  (SvrOutlineInfo).usNumAliasTables; ++i)
    printf("\n  %s", (SvrOutlineInfo).pAliasTables[i]);
}

return sts;
}
```

前述の例の出力は、次のとおりです:

```
    Case sensitivity is set to: 0
Outline type is set to: 0
Outline allows duplicate names is set to: 1
Number of alias tables is: 2
Names of the alias tables are:
 Default
 Long Names
```

EssGetStatBufSize

EssDumpPerfStats で取得したパフォーマンス統計テーブルに必要なバッファのサイズに対するポインタを戻します。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
pBufSize;	ESS_PULONG_T	パフォーマンス統計テーブルを保持する文字配列に必要なバッファのサイズに対するポインタ。

備考

- EssDumpPerfStats を呼び出す前にこの関数を呼び出して、pStatBuf で指定されたアドレスにパフォーマンス統計テーブル用に割り当てるためのメモリー量を確認します。

- パフォーマンス統計がこれまで使用可能になっていない場合、つまり EssResetPerfStats の persistence が 4 に設定されたことがない場合、pBufSize で指定されたバッファ・サイズは 0 になります。

戻り値

- 成功の場合、
  - この関数は 0 を戻します。
  - pBufSize は、EssDumpPerfStats によって取得されたパフォーマンス統計テーブルを保持する文字配列に必要なバッファのサイズへのポインタを含みます。
- パフォーマンス統計テーブルの詳細は、Oracle Essbase テクニカル・リファレンスの MaxL のパフォーマンス統計に関するトピックを参照してください。

アクセス

この関数を使用するには、スーパーバイザ・アクセス権が必要です。

例

EssGetStatBufSize を呼び出すコード例は、EssDumpPerfStats の例を参照してください。

関連トピック

- EssDumpPerfStats
- EssResetPerfStats

EssGetString

アクティブ・データベースから文字列データを取得します。データが戻される場合、この関数は EssReport または EssEndReport の後に呼び出す必要があります。

構文

```
ESS_FUNC_M EssGetString (
hCtx, pString
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pString	ESS_PSTR_T	割り当てられて戻されたデータ文字列を受け取るポインタのアドレス。

備考

- レポートの実行に成功した後以外にこの関数を呼び出すと、エラーが発生します。
- 戻される文字列の長さは、64KB 未満です。

- このコマンドを使用するときは必ず改行してください。改行しないと、エラーになります。
- NULL 文字列が戻されるまでこの関数を呼び出す必要があります。
- pString に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

データ文字列に割り当てられたポインタが pString に戻されます。このポインタは、戻すデータがこれ以上ない場合は NULL になります。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
     ESS_HINST_T  hInst
    )
{
 ESS_FUNC_M    sts   = ESS_STS_NOERR;
 ESS_STR_T   rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");
 if (!sts)
  sts = EssEndReport (hCtx);
 /***************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

関連トピック

- EssEndReport
- EssReport
- EssQueryDatabaseMembers

EssGetUser

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。

構文

```
ESS_FUNC_M EssGetUser (
hCtx, UserName, ppUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
ppUserInfo	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	割り当てられたユーザー情報構造体を受け取るポインタのアドレス。

備考

ppUserInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー情報構造体が ppUserInfo に戻されます。

アクセス

この関数を使用するには、独自のユーザー情報を取得する場合を除き、呼出し元はログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetUserInfo (ESS_HCTX_T  hCtx,
        ESS_HINST_T hInst
        )
{
 ESS_FUNC_M     sts  = ESS_STS_NOERR;
 ESS_PUSERINFO_T  User = NULL;

 sts = EssGetUser (hCtx, "Jim Smith", &User);
 if (!sts)
 {
  if (User)
    EssFree (hInst, User);
 }

 return (sts);
}
```

関連トピック

- EssGetUserInfoEx
- EssGetApplicationAccess
- EssListUsers
- EssSetUser

EssGetUserEx

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。

構文

```
ESS_FUNC_M EssGetUserEx (
hCtx, UserName, ppUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
ppUserInfoEx	207 ページの「ESS_USERINFOEX_T」	外部認証ユーザーの情報構造体を受け取るポインタのアドレス。

備考

ppUserInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー情報構造体が ppUserInfo に戻されます。

アクセス

この関数を使用するには、独自のユーザー情報を取得する場合を除き、呼出し元はログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_GetUserInfo (ESS_HCTX_T  hCtx,
        ESS_HINST_T hInst
        )
{
 ESS_FUNC_M    sts  = ESS_STS_NOERR;
 ESS_PUSERINFO_T  User = NULL;

 sts = EssGetUserEx (hCtx, "Jim Smith", &User);
 if (!sts)
 {
  if (User)
```

```
    EssFree (hInst, User);
  }
  return (sts);
}
```

関連トピック

- EssGetApplicationAccess
- EssCreateExtUser
- EssListUsers
- EssSetUser
- EssSetUserEx
- 207 ページの「ESS_USERINFOEX_T」

EssGetUserInfoEx

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。EssGetUser に似ていますが、ユーザー・ディレクトリの指定、または UserID の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssGetUserInfoEx (
hCtx
,
UserId
,
bIsIdentity
,
ppUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
UserName	ESS_STR_T	ユーザー名(入力)。username@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。
ppUserInfo	ESS_PUSERINFOID_T	割り当てられたユーザー情報構造体を受け取るポインタのアドレス(出力)。ユーザー・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

ppUserpInfo に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー情報構造体が ppUserInfo に戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
void DisplayUserInfoID(ESS_PUSERINFOID_T userInfo)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_BOOL_T isDefined = ESS_TRUE;

 printf("\tUser Name: %s\n", userInfo->Name);
 printf("\tProvider Name: %s\n", userInfo->ProviderName);
 printf("\tConnparam: %s\n", userInfo->connparam);
 printf("\tDescription: %s\n", userInfo->Description);
 printf("\tEMail Identification: %s\n", userInfo->EMailID);

 if (userInfo->LockedOut)
  printf("\tLocked out: Yes\n");
 else
  printf("\tLocked out: No\n");

 if (userInfo->PwdChgNow)
  printf("\tChange the password now: Yes\n");
 else
  printf("\tChange the password now: No\n");

 printf(  "\tConnected Application: %s\n", userInfo->AppName);
 printf(  "\tConnected Database: %s\n",  userInfo->DbName);

 if (userInfo->Login)
  printf("\tLogged in: Yes\n");
 else
  printf("\tLogged in: No\n");

 switch(userInfo->Access)
 {
  case ESS_ACCESS_ADMIN:
   printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userInfo->Access);
   break;
  case ESS_ACCESS_APPALL:
   printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userInfo->Access);
   break;
  case ESS_ACCESS_DBALL:
   printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo->Access);
   break;
  case ESS_ACCESS_APPCREATE:
   printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo->Access);
   break;
  case ESS_ACCESS_APPMANAGE:
   printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userInfo->Access);
   break;
  case ESS_ACCESS_DBCREATE:
   printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userInfo->Access);
   break;
```

```
 case ESS_ACCESS_DBMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userInfo->Access);
  break;
 case ESS_ACCESS_CALC:
  printf("\tAccess: %d - ESS_ACCESS_CALC\n", userInfo->Access);
  break;
 case ESS_ACCESS_WRITE:
  printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userInfo->Access);
  break;
 case ESS_ACCESS_READ:
  printf("\tAccess: %d - ESS_ACCESS_READ\n", userInfo->Access);
  break;
 case ESS_PRIV_USERCREATE:
  printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userInfo->Access);
  break;
 case ESS_PRIV_APPCREATE:
  printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userInfo->Access);
  break;
 case ESS_PRIV_APPMANAGE:
  printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userInfo->Access);
  break;
 case ESS_PRIV_APPLOAD:
  printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userInfo->Access);
  break;
 case ESS_PRIV_DBCREATE:
  printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userInfo->Access);
  break;
 case ESS_PRIV_DBMANAGE:
  printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userInfo->Access);
  break;
 case ESS_PRIV_DBLOAD:
  printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userInfo->Access);
  break;
 case ESS_PRIV_CALC:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo->Access);
  break;
 case ESS_PRIV_WRITE:
  printf("\tAccess: %d - ESS_PRIV_WRITE\n", userInfo->Access);
  break;
 case ESS_PRIV_READ:
  printf("\tAccess: %d - ESS_PRIV_READ\n", userInfo->Access);
  break;
 case ESS_PRIV_NONE:
  printf("\tAccess: %d - ESS_PRIV_NONE\n", userInfo->Access);
  break;
 default:
  printf("\tAccess: Unknown\n");
}

switch(userInfo->MaxAccess)
{
 case ESS_ACCESS_ADMIN:
  printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userInfo->MaxAccess);
  break;
 case ESS_ACCESS_APPALL:
  printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userInfo->MaxAccess);
  break;
```

```
case ESS_ACCESS_DBALL:
 printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_APPCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_CALC:
 printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_WRITE:
 printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userInfo->MaxAccess);
 break;
case ESS_ACCESS_READ:
 printf("\tMax Access: %d - ESS_ACCESS_READ\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_DBLOAD:
 printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_CALC:
 printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_WRITE:
 printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_READ:
 printf("\tMax Access: %d - ESS_PRIV_READ\n", userInfo->MaxAccess);
 break;
case ESS_PRIV_NONE:
 printf("\tMax Access: %d - ESS_PRIV_NONE\n", userInfo->MaxAccess);
 break;
```

```
   default:
     printf("\tMax Access: Unknown\n");
  }

  printf("\tPassword Expiration in Dates: %d\n",userInfo->Expiration);
  //EssSdCTime(NULL, userInfo->LastLogin, sizeof(time_string), time_string);
  //printf("\tLast Successful Login:      %s\n", time_string);
  printf("\tFailed Login Attempts Since Then: %d\n", userInfo->FailCount);
  printf("\tLogin ID: %d\n", userInfo->LoginId);
  printf( "\n");
}


ESS_FUNC_M ESS_GetUserInfoEx (ESS_HCTX_T hCtx)

{
  ESS_STS_T sts  = ESS_STS_NOERR;
  ESS_STR_T userId;
  ESS_BOOL_T bUsingIdentity;
  ESS_PUSERINFOID_T userInfo = NULL;

  bUsingIdentity = ESS_TRUE;
  sts = EssGetUserInfoEx (hCtx, userId, bUsingIdentity, &userInfo);
  printf("EssGetUserInfoEx sts: %ld\n", sts);
  if (userInfo)
  {
    DisplayUserInfoID(userInfo);
  }


  return (sts);
}
```

関連トピック

- EssListUsersInfoEx

EssGetUserType

ユーザーのアプリケーション・アクセス・タイプを検索できます。

構文

```
ESS_FUNC_M EssGetUserType (
hCtx
,
UserName
,
UserType
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
UserName	ESS_STR_T	ユーザーの名前。
UserType	ESS_PUSER_TYPE_T	指定した UserName に定義されているアプリケーション・アクセス・タイプ。

戻り値

API のステータスを戻します。

例

```
    ESS_FUNC_M
ESS_GetUserType (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M          sts = ESS_STS_NOERR;
 ESS_STR_T          UserName="jsmith";
 ESS_PUSER_TYPE_T       UserType;

 sts = EssGetUserType (hCtx, UserName, UserType);
 printf("user type for the user %s is %d\n", UserName, *UserType);

 return (sts);
}
```

関連トピック

- EssSetUserType

EssGetVariable

代替変数の値を取得します。

構文

```
ESS_FUNC_M EssGetVariable (
hCtx, pVariable
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pVariable	209 ページの「ESS_VARIABLE_T」	指定された代替変数の説明を含む構造体を指すポインタ。

戻り値

成功の場合、この関数により構造体 ESS_VARIABLE_T の VarValue フィールドに代替変数の値が戻されます。

例

```
    /*
** ESS_GetVariable() gets the substitution variable value using
** the API function EssGetVariable.
*/
ESS_FUNC_M
ESS_GetVariable (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_VARIABLE_T Variable;
 printf("\n ************************************");
 printf("\n **** An example of using EssGetVariable");
 printf("\n ************************************");

 /*******************************/
 /* Get the Value of QuarterName */
 /*******************************/
 strcpy(Variable.VarName, "QuarterName");
 strcpy(Variable.Server,  "Local");
 strcpy(Variable.AppName, "Sample");
 strcpy(Variable.DbName,  "Basic");
 sts = EssGetVariable(hCtx, &Variable);
 if (sts == ESS_STS_NOERR)
 {
  printf("\n------- Substitution Variable 'QuarterName' Information \n");
  printf("Variable name   : %s\n",  Variable.VarName);
  printf("Server name    : %s\n",  Variable.Server);
  printf("Application name : %s\n",  Variable.AppName);
  printf("Database name   : %s\n",  Variable.DbName);
  printf("Variable value  : %s\n\n", Variable.VarValue);
 }
 /************************************************************/
 /* Get the Value of MarketName at the level of the Server/App */
 /************************************************************/
 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName, "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName, "Sample");
  strcpy(Variable.DbName,  "");
  sts = EssGetVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
  {
   printf("\n------- Substitution Variable 'MarketName' Information \n");
   printf("Variable name   : %s\n",  Variable.VarName);
   printf("Server name    : %s\n",  Variable.Server);
   printf("Application name : %s\n",  Variable.AppName);
   printf("Database name   : %s\n",  Variable.DbName);
   printf("Variable value  : %s\n\n", Variable.VarValue);
  }
 }
 /******************************************************/
 /* Get the Value of MarketName at the level of the Server */
 /******************************************************/
 if (sts == ESS_STS_NOERR)
```

```
 {
  strcpy(Variable.VarName, "MarketName");
  strcpy(Variable.Server,  "Local");
  strcpy(Variable.AppName, "");
  strcpy(Variable.DbName,  "");
  sts = EssGetVariable(hCtx, &Variable);
  if (sts == ESS_STS_NOERR)
  {
   printf("\n------- Substitution Variable 'MarketName' Information \n");
   printf("Variable name   : %s\n",  Variable.VarName);
   printf("Server name     : %s\n",  Variable.Server);
   printf("Application name : %s\n",  Variable.AppName);
   printf("Database name   : %s\n",  Variable.DbName);
   printf("Variable value  : %s\n\n", Variable.VarValue);
  }
 }

 if (sts == ESS_STS_NOERR)
  printf("\n --> No Errors in EssGetVariable\n\n\n");
 else
  printf("\n --> Error in EssGetVariable number: %d\n\n\n", sts);

 return (sts);
} /* End ESS_GetVariable */
```

出力

```
    ************************************
**** An example of using EssGetVariable
************************************
------- Substitution Variable 'QuarterName' Information
Variable name   : QuarterName
Server name     : Local
Application name : Sample
Database name   : Basic
Variable value  : Qtr2

------- Substitution Variable 'MarketName' Information
Variable name   : MarketName
Server name     : Local
Application name : Sample
Database name   :
Variable value  : East

------- Substitution Variable 'MarketName' Information
Variable name   : MarketName
Server name     : Local
Application name :
Database name   :
Variable value  : Market
 --> No Errors in EssGetVariable
```

関連トピック

- 209 ページの「ESS_VARIABLE_T」

- EssCreateVariable
- EssDeleteVariable
- EssListVariables

EssGetVersion

接続されている Essbase サーバーの完全なバージョン番号を、11.1.2 のようなリリース.バージョン.改訂の形式で取得します。

構文

```
ESS_FUNC_M EssGetVersion (
hCtx, pRelease, pVersion, pRevision
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRelease	ESS_PUSHORT_T	リリース番号を受け取る変数のアドレス。
pVersion	ESS_PUSHORT_T	バージョン番号を受け取る変数のアドレス。
pRevision	ESS_PUSHORT_T	改訂番号を受け取る変数のアドレス。

備考

Essbase サーバー・バージョンがプログラムで使用されているすべての機能をサポートすることを確認するため、サーバーに接続してからこの関数を呼び出せます。

戻り値

正常終了の場合、完全な Essbase のバージョン番号が pRelease、pVersion および pRevision の形式で戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_GetVersion (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_USHORT_T  Release;
 ESS_USHORT_T  Version;
 ESS_USHORT_T  Revision;

 sts = EssGetVersion (hCtx, &Release, &Version,
   &Revision);
 if (!sts)
 {
  printf ("\r\nEssbase Application Server - Version %d.%d.%d\r\n", Release, Version,
```

```
Revision);
  }

 return (sts);
}
```

関連トピック

- EssInit
- EssGetAPIVersion

EssImport

様々なソースから Essbase サーバーへのデータのインポートを許可します。

構文

```
ESS_FUNC_M EssImport (
hCtx, pRules, pData, ppMbrErr, pMbrUser, abortOnError
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRules	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pData	159 ページの「ESS_OBJDEF_T」	データ・ファイル・オブジェクト定義構造体へのポインタ。
ppMbrErr	156 ページの「ESS_MBRERR_T」	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● ESS_MBRERR_BADDIM ● ESS_MBRERR_BADGEN ● ESS_MBRERR_UNKNOWN ● ESS_MBRERR_BADACCESS ● ESS_MBRERR_BADSYNTAX
pMbrUser	157 ページの「ESS_MBRUSER_T」	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
abortOnError	ESS_USHORT_T	TRUE の場合、最初のエラーでインポートが停止します。それ以外の場合は続行します。

備考

- SQL ソース以外では、pRules および pData の ESS_OBJDEF_T 構造体の AppName および DbName フィールドが NULL である場合は、hCtx にローカル・コンテキスト・ハンドルを指定し、ESS_OBJDEF_T の FileName フィールドにはファイルへの完全修飾パスを指定する必要があります。

- ローカル・オブジェクトが使用される場合は、EssCreateLocalContext を最初に呼び出す必要があります。
- ppMbrErr に対して割り当てられたメモリーは、EssFreeMbrErr を使用して解放する必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Import(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_SHORT_T  isAbortOnError;
 ESS_OBJDEF_T Rules;
 ESS_OBJDEF_T Data;
 ESS_MBRUSER_T User;
 ESS_PMBRERR_T pMbrErr = NULL;

 Data.hCtx    = hCtx;
 Data.AppName  = "Olap";
 Data.DbName   = "Demo";
 Data.ObjType  = ESS_OBJTYPE_TEXT;
 Data.FileName  = "Actuals";

 Rules.hCtx   = hCtx;
 Rules.AppName  = "Olap";
 Rules.DbName  = "Demo";
 Rules.ObjType  = ESS_OBJTYPE_RULES;
 Rules.FileName = "Actmap";
 /**********************/
 /* Running conditions */
 /**********************/
 isAbortOnError = ESS_TRUE;

 sts = EssImport (hCtx, &Rules, &Data, &pMbrErr,
    NULL, isAbortOnError);
 if(pMbrErr)
  EssFreeMbrErr(hCtx, pMbrErr);
 /*********************************************************************/
 /*                                */
 /* When a SQL data source is defined in the rules file, define   */
 /* the variables in the ESS_OBJDEF_T Data structure as follows:   */
 /*   Data.hCtx    = hCtx;                    */
 /*   Data.AppName  = NULL;                     */
 /*   Data.DbName   = NULL;                    */
 /*   Data.ObjType  = ESS_OBJTYPE_NONE;              */
 /*   Data.FileName = NULL;                     */
```

```
 /*                                */
 /* Also, provide strings for the variables in the ESS_MBRUSER_T   */
 /* User structure; for example:                  */
 /*   User.User    = "Dbusernm";                */
 /*   User.Password = "Dbpasswd";                 */
 /*                                */
 /* Use a blank string for User and Password, if the SQL source   */
 /* does not require user and password information; for example:   */
 /*   User.User    = "";                        */
 /*   User.Password = "";                         */
 /*                                */
 /* Also, define sts as follows:                  */
 /*   sts =  EssImport (hCtx, &Rules, &Data, &pMbrErr,       */
 /*      &User, isAbortOnError);                */
 /*                                */
 /****************************************************************/
}
```

関連トピック

- EssExport
- EssBuildDimension
- EssFreeMbrErr

EssImportASO

様々なソースから Essbase 集約ストレージ・データベースへのデータのインポートを許可します。

構文

```
ESS_FUNC_M EssImportASO (
hCtx, pRules, pData, ppMbrErr, pUser, usabortOnError, ulBufferId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRules	159 ページの「ESS_OBJDEF_T」	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pData	159 ページの「ESS_OBJDEF_T」	データ・ファイル・オブジェクト定義構造体へのポインタ。
ppMbrErr	156 ページの「ESS_MBRERR_T」	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● ESS_MBRERR_BADDIM ● ESS_MBRERR_BADGEN ● ESS_MBRERR_UNKNOWN ● ESS_MBRERR_BADACCESS ● ESS_MBRERR_BADSYNTAX

パラメータ	データ型	説明
pUser	157 ページの「ESS_MBRUSER_T」	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
usabortOnError	ESS_USHORT_T	TRUE の場合、最初のエラーでインポートが停止します。それ以外の場合は続行します。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID（1 から 999,999 までの数)。データのロードが完了する前にバッファを廃棄するには、バッファを初期化する場合に使用したものと同じ ulBufferId 番号を使用する必要があります。

備考

- SQL ソース以外では、pRules および pData の ESS_OBJDEF_T 構造体の AppName および DbName フィールドが NULL である場合は、hCtx にローカル・コンテキスト・ハンドルを指定し、ESS_OBJDEF_T の FileName フィールドにはファイルへの完全修飾パスを指定する必要があります。
- ローカル・オブジェクトが使用される場合は、EssCreateLocalContext を最初に呼び出す必要があります。
- ppMbrErr に対して割り当てられたメモリーは、EssFreeMbrErr を使用して解放する必要があります。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    void TestImportASO(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_SHORT_T   isAbortOnError;
    ESS_OBJDEF_T   Rules;
    ESS_OBJDEF_T   Data;
    ESS_PMBRERR_T  pMbrErr = NULL;
    ESS_PMBRUSER_T  pMbrUser = NULL;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T   ulDuplicateAggregationMethod;
    ESS_ULONG_T   ulOptionsFlags;
    ESS_ULONG_T   ulSize;
    ESS_ULONG_T   ulBufferCnt;
    ESS_ULONG_T   ulCommitType ;
    ESS_ULONG_T   ulActionType;
    ESS_ULONG_T   ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];

    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
```

```
    ulBufferId = 10;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
     ulDuplicateAggregationMethod, ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* Server object */
    Rules.hCtx   = hCtx;
    Rules.AppName  = AppName;
    Rules.DbName  = DbName;
    Rules.ObjType  = ESS_OBJTYPE_RULES;
    Rules.FileName = "Dataload";
    Data.hCtx    = hCtx;
    Data.AppName  = AppName;
    Data.DbName   = DbName;
    Data.ObjType  = ESS_OBJTYPE_TEXT;
    Data.FileName  = "Dataload";
    isAbortOnError = ESS_TRUE;

    sts = EssImportASO (hCtx, &Rules, &Data, &pMbrErr, pMbrUser,     isAbortOnError,
ulBufferId);
    printf("EssImportASO sts: %ld\n",sts);
    if(pMbrErr)
       EssFreeMbrErr(hCtx, pMbrErr);

    /* Commit and delete the buffer */
    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\nCommit data to the main slice:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt,     ulBufferIdAry,
ulCommitType, ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);
}
```

関連トピック

- EssLoadBufferInit
- EssBeginDataloadASO
- EssSendString
- EssEndDataload
- EssLoadBufferTerm
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssListExistingLoadBuffers
- EssMergeDatabaseData

EssIncrementalBuildDim

指定したルール・ファイルとデータ・ソースを使用して次元を構築します。増分次元構築プロトコル内では、複数回呼び出せます。

EssBeginIncrementalBuildDim は、この関数より先に呼び出す必要があります。

構文

```
      ESS_FUNC_M EssIncrementalBuildDim(
      hCtx, RulesObj, DataObj, MbrUser, ErrorName, bOverwrite, usBuildOption,
szTmpOtlFile
      )
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RulesObj	ESS_POBJDEF_T	ルール・ファイル・オブジェクト定義構造体へのポインタ。
DataObj	ESS_POBJDEF_T	データ・ファイル・オブジェクト定義構造体へのポインタ。
MbrUser	ESS_PMBRUSER_T	SQL ユーザー構造体(データ・ソースが SQL データベースの場合)。 SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
ErrorName	ESS_STR_T	クライアントでのエラー出力ファイルの名前。
bOverwrite	ESS_BOOL_T	ブール。値: ● ESS_TRUE – 既存のエラー・ファイルを上書きします。 ● ESS_FALSE – 上書きしません。既存のエラー・ファイルに追加します。
usBuildOption	ESS_USHORT_T	有効な値: ● ESS_INCDIMBUILD_BUILD メンバーの構築のみ行います。 ● ESS_INCDIMBUILD_VERIFY メンバーを構築してアウトラインを確認します。 ● ESS_INCDIMBUILD_SAVEOTL メンバーを構築して、アウトラインを一時アウトライン・ファイルに保存します。 ● ESS_INCDIMBUILD_ALL メンバーを構築し、アウトラインを確認し、再構築します。
szTmpOtlFile	ESS_STR_T	一時アウトライン・ファイル名。この回の次元構築で得られたアウトラインにアウトライン確認エラーがある場合、Essbase は拡張子が.otb の一時アウトライン・ファイルを作成します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

EssBeginIncrementalBuildDim を参照してください。

関連トピック

- EssIncrementalBuildDim
- EssBeginIncrementalBuildDim
- EssBeginStreamBuildDim

- EssEndIncrementalBuildDim
- EssEndStreamBuildDim

EssInit

API およびメッセージ・データベースを初期化します。

構文

```
ESS_FUNC_M EssInit (
pInitStruct, phInstance
);
```

パラメータ	データ型	説明
pInitStruct	148 ページの「ESS_INIT_T」	API 初期化構造体へのポインタ。
phInstance	ESS_PHINST_T	Essbase API インスタンス・ハンドルへのポインタ。

備考

- この関数は、他のすべての Essbase API 関数よりも前に呼び出す必要があります。
- 初期化構造体のいずれかのフィールドが NULL または値 0(適切な場合)の場合は、API はこれらのパラメータにデフォルト値を使用します。

戻り値

この関数に渡される ESS_INIT_T 構造体は、多数の初期化パラメータを含んでいます。これらのパラメータとしてメッセージ・データベースの名前、割り当てられるクライアント・バッファの最大サイズ、ユーザー定義のメモリー解放割り当て、エラー・コールバック関数へのポインタ、ヘルプ・ファイルの名前、場所およびバージョン番号が含まれます。

この関数は、複数のアプリケーションが独立して API にアクセスできるようにする phInstance インスタンス・ハンドルを戻します(DLL の場合のみ)。インスタンス・ハンドルは保存し、EssLogin、EssTerm およびメモリー割当て関数に渡す必要があります。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_VOID_T ESS_Init()
{
 ESS_HINST_T  hInst;
 ESS_INIT_T InitStruct =     /* Define init */
           /* structure */
 {
  ESS_API_VERSION,    /* Version of API */
  USER,    /* user-defined message context */
  0,     /* max handles   */
```

```
  0L,     /* max buffer size */
  "C:\\Essbase", /* local path */
  /* The following parameters use defaults  */
  NULL,    /* message db path */
  NULL,    /* allocation function pointer */
  NULL,    /* reallocation function pointer */
  NULL,    /* free function pointer */
  NULL,  /* error handling function pointer */
  NULL,    /* path name of user-defined */
       /* Application help file */
  NULL,    /* Reserved for internal use. */
       /* Set to NULL */
 };
/* Initialize the API  */
if ((sts = EssInit (&InitStruct, &hInst)) != ESS_STS_NOERR)
{
  /* error initializing API */
  exit ((ESS_USHORT_T) sts);
 }
```

関連トピック

- EssLogin
- EssAutoLogin
- EssTerm

EssKillRequest

特定の Essbase ユーザー・セッションまたは要求を終了します。

構文

```
ESS_FUNC_M EssKillRequest (
hCtx, pRequestInfoStruct
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pRequestInfoStruct	ESS_PREQUESTINFO_T	要求情報構造体を指すポインタ。

備考

この関数は、189 ページの「ESS_REQUESTINFO_T」内の現行セッションに関する情報を使用して、特定のユーザー・セッションの終了を要求します。この関数は、ユーザー・セッション中に、アプリケーション、データベースまたはシステムに対して行われているアクティブな要求を終了(ユーザーのログアウトは行わずに)する場合にも使用できます。

セッションとは、Essbase ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。

要求とは、ユーザーまたは他のプロセスが Essbase に対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。

この関数は、ESS_REQUESTINFO_T 構造体の UserName、AppName および DbName で指定されたセッションおよび要求を終了します。これらのフィールドが NULL の場合、この関数はこのプロセス(ユーザー)によって起動されたすべてのセッションと要求を終了します。アプリケーション・プログラムは、ESS_REQUESTINFO_T によって使用されるメモリーの割当てと解放を行います。

自分のアクティブ要求を切断する(クエリーの取消し)には、Essbase 初期化構造体に関する項の説明に従ってカスタム・コールバック関数を設定します。ESS_INIT_T を参照してください。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    #include
#include

ESS_FUNC_M ESS_ListRequest ()
{
  ESS_FUNC_M    sts   = ESS_STS_NOERR;
  ESS_STR_T   rString = NULL;
  ESS_HCTX_T  hCtx;
  ESS_USHORT_T Items;
  ESS_PAPPDB_T pAppsDbs = NULL;
  ESS_HINST_T  hInst ;
  ESS_ACCESS_T Access;
  ESS_USHORT_T numRequest;
  ESS_PREQUESTINFO_T requestInfo;

  ESS_INIT_T InitStruct =    /* Define init */
         /* structure */
  {
    ESS_API_VERSION,    /* Version of API */
    NULL,    /* user-defined message context */
    0,     /* max handles   */
    0L,     /* max buffer size */
    NULL, /* local path */
    /* The following parameters use defaults  */
    NULL,    /* message db path */
    NULL,    /* allocation function pointer */
    NULL,    /* reallocation function pointer */
    NULL,    /* free function pointer */
    NULL,  /* error handling function pointer */
    NULL,    /* path name of user-defined */
       /* Application help file */
    NULL,    /* Reserved for internal use. */
       /* Set to NULL */
  };
```

```
  EssInit (&InitStruct, &hInst);

  sts = EssLogin (hInst, "local", "admin", "password", &Items, &pAppsDbs, &hCtx);

  sts = EssSetActive ( hCtx, "sample", "basic", &Access );

  sts = EssListRequests( hCtx, NULL, NULL, NULL, &numRequest, &requestInfo);

  printf ( "Total requests on the server %d\n", numRequest );

  if ( !sts && requestInfo )
  {
    ESS_USHORT_T index = 0;

    while ( index < numRequest )
    {
      printf ( "login ID = %ul\n", requestInfo[index].LoginId );
      printf ( "user name = %s\n", requestInfo[index].UserName );
      printf ( "login machine = %s\n", requestInfo[index].LoginSourceMachine );
      printf ( "AppName = %s\n", requestInfo[index].AppName );
      printf ( "DbName = %s\n", requestInfo[index].DbName );
      printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
      printf ( "RequestString = %s\n", requestInfo[index].RequestString );
      printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
      printf ( "State = %d\n", requestInfo[index].State );
      printf ( "\n\n---------------------------------------\n\n",
requestInfo[index].State );

      sts = EssKillRequest (hCtx, &requestInfo[index] );

      index++;
    }

    EssFree ( hInst, requestInfo );
  }

  EssLogout (hCtx);
  EssTerm (hInst);
  return(sts);
}

void main()
{
  ESS_ListRequest ();
}
```

関連トピック

- EssKillRequestEx
- EssListRequests
- 189 ページの「ESS_REQUESTINFO_T」
- 196 ページの「ESS_REQ_STATE_T」

EssKillRequestEx

特定の Essbase ユーザー・セッションまたは要求を終了します。EssKillRequest に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

構文

```
ESS_FUNC_M EssKillRequestEx (
hCtx
,
pRequestInfoStruct
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
pRequestInfoStruct	ESS_PREQUESTINFOEX_T	要求情報構造体へのポインタ(入力)。

備考

この関数は ESS_REQUESTINFO_T 内の現行セッションに関する情報を使用して、特定のユーザー・セッションの終了を要求します。この関数は、ユーザー・セッション中に、アプリケーション、データベースまたはシステムに対して行われているアクティブな要求を終了(ユーザーのログアウトは行わずに)する場合にも使用できます。

セッションとは、Essbase ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。

要求とは、ユーザーまたは他のプロセスが Essbase に対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。

この関数は、ESS_REQUESTINFOEX_T 構造体の UserName、AppName、および DbName で指定されたセッション/要求を終了します。これらのフィールドが NULL の場合、この関数はこのプロセス(ユーザー)によって起動されたすべてのセッションと要求を終了します。アプリケーション・プログラムは、ESS_REQUESTINFOEX_T によって使用されるメモリーの割当てと解放を行います。

自分のアクティブ要求を切断する(クエリーの取消し)には、Essbase 初期化構造体に関する項の説明に従ってカスタム・コールバック関数を設定します。ESS_INIT_T を参照してください。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
void ListRequestsEx ()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_STR_T userId;
    ESS_BOOL_T bIsIdentity;
    ESS_USHORT_T numRequest;
    ESS_PREQUESTINFOEX_T requestInfo;
    ESS_USHORT_T index = 0;

    userId = "admin";
    bIsIdentity = ESS_FALSE;
    sts = EssListRequestsEx(hCtx, userId, bIsIdentity, "Sample", "Sample1",
&numRequest, &requestInfo);
    printf("\nEssListRequestsEx sts: %ld\n", sts);
    printf ( "Total requests on the server: %d\n", numRequest );
    if ( !sts && requestInfo )
    {
        while ( index < numRequest )
        {
            printf ( "login ID = %ul\n", requestInfo[index].LoginId );
            printf ( "user name = %s\n", requestInfo[index].UserName );
            printf ( "login machine = %s\n",
requestInfo[index].LoginSourceMachine );
            printf ( "AppName = %s\n", requestInfo[index].AppName );
            printf ( "DbName = %s\n", requestInfo[index].DbName );
            printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
            printf ( "RequestString = %s\n", requestInfo[index].RequestString );
            printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
            printf ( "State = %d\n", requestInfo[index].State );
            printf ( "\n\n--------------------------------------\n\n",
requestInfo[index].State );

            sts = EssKillRequestEx (hCtx, &requestInfo[index] );

            index++;
        }

        EssFree ( hInst, requestInfo );
    }

    userId = " native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER ";
    bIsIdentity = ESS_TRUE;
    sts = EssListRequestsEx(hCtx, userId, bIsIdentity, "Sample", "Sample1",
&numRequest, &requestInfo);
    printf("\nEssListRequestsEx sts: %ld\n", sts);
    printf ( "Total requests on the server: %d\n", numRequest );
    if ( !sts && requestInfo )
    {
        while ( index < numRequest )
        {
            printf ( "login ID = %ul\n", requestInfo[index].LoginId );
            printf ( "user name = %s\n", requestInfo[index].UserName );
            printf ( "login machine = %s\n",
requestInfo[index].LoginSourceMachine );
            printf ( "AppName = %s\n", requestInfo[index].AppName );
            printf ( "DbName = %s\n", requestInfo[index].DbName );
            printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
```

```
            printf ( "RequestString = %s\n", requestInfo[index].RequestString );
            printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
            printf ( "State = %d\n", requestInfo[index].State );
            printf ( "\n\n---------------------------------------\n\n",
requestInfo[index].State );

            sts = EssKillRequestEx (hCtx, &requestInfo[index] );

            index++;
        }

        EssFree ( hInst, requestInfo );
    }
}
```

関連トピック

- EssListRequestsEx

EssListAliases

アクティブ・データベース内のすべての別名テーブルをリストします。

構文

```
ESS_FUNC_M EssListAliases (
hCtx, pCount ppAliasList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCount	ESS_PUSHORT_T	別名テーブルのカウントを受け取る変数のアドレス。
ppAliasList	ESS_PPALIASNAME_T	割り当てられた別名テーブル名の配列を受け取るポインタのアドレス。

備考

ppAliasList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、別名テーブルのカウントが pCount に戻され、割り当てられた別名テーブル名の配列が ppAliasList に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_ListAliases (ESS_HCTX_T  hCtx,
        ESS_HINST_T hInst
       )
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 ESS_USHORT_T    Count;
 ESS_USHORT_T    ind;
 ESS_PALIASNAME_T  Altlist = NULL;

 sts = EssListAliases (hCtx, &Count, &Altlist);
 if (!sts)
 {
  if (Count && Altlist)
  {
   printf ("\r\n-----List of Aliases-----\r\n\r\n");

   for (ind = 0; ind < Count; ind++)
   {
     if (Altlist [ind] != NULL)
      printf ("%s\r\n", Altlist[ind]);
   }
   EssFree (hInst, Altlist);
  }
  else
   printf ("\r\nAlias List is Empty\r\n\r\n");
 }

 return (sts);
}
```

関連トピック

- EssDisplayAlias
- EssSetActive

EssListApplications

呼出し元がアクセスできる、すべてのアプリケーションをリストします。

構文

```
     ESS_FUNC_M EssListApplications (
     hCtx, pCount, ppAppList
     );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCount	ESS_PUSHORT_T	戻されるアプリケーションのカウントを受け取る変数のアドレス。

パラメータ	データ型	説明
ppAppList	ESS_PPAPPNAME_T	割り当てられたアプリケーション名文字列の配列を受け取るポインタのアドレス。

備考

ppAppList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、アクセス可能なアプリケーション数のカウントが pCount に戻され、アプリケーション名の文字列の配列が ppAppList に戻されます。配列内には「カウント」数のアイテムがあります。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアプリケーションにアクセスした場合、サーバー・アプリケーションのみがリストされることに注意してください。

例

```
    ESS_FUNC_M
ESS_ListApps (ESS_HCTX_T  hCtx,
      ESS_HINST_T hInst
     )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PAPPNAME_T strp = NULL;
 ESS_USHORT_T  Items;
 ESS_USHORT_T  ind;

 sts = EssListApplications (hCtx, &Items, &strp);
 if (!sts)
 {
  if (Items && strp)
  {
   printf ("Applications availables\r\n");
   for (ind = 0; ind < Items; ind++)
   {
     if (strp [ind] != NULL)
      printf ("%s\r\n", strp [ind]);
   }
   EssFree (hInst, strp);
  }
  else
   printf ("\r\nApplication List is Empty\r\n\r\n");
 }
 printf ("\r\n");

 return (sts);
}
```

関連トピック

- EssListDatabases
- EssListObjects

EssListCalcFunctions

アクティブなアプリケーションで使用可能なすべての計算関数をリストします。これにはすべてのネイティブ関数と、カスタム定義関数(CDF)およびカスタム定義マクロ(CDM)が含まれます。

構文

```
ESS_FUNC_M EssListCalcFunctions (
hCtx, pCalcFunc
);
```

パラメータ　データ型　　説明

hCtx　　ESS_HCTX_T API コンテキスト・ハンドル。

pCalcFunc　ESS_PSTR_T 使用可能な計算関数を含む文字列へのポインタ。この文字列は XML 形式です。

備考

この関数を使用するには、(通常は管理者に付与される)スーパーバイザ権限が必要です。ユーザーがこのリストを入手するためには、データベースへのアクセス権も必要です。エラーを避けるため、この関数を呼び出すプログラムの実行にはスーパーバイザ権限とデータベースへのアクセス権の両方が必要です。

EssGetCalcList によって戻される文字列のコンテンツは XML でフォーマットされ、XML ユーティリティでレンダリングするか、構文解析によって実際のテキストのみ表示する必要があります。すべての XML タグは山カッコで囲まれています(たとえば、<xml_tag>)。

典型的な XML 出力ファイルを短縮した例:

```
     ESSBASE API v.62000
1051034: Logging in user admin
1051035: Last login on Tuesday, May 22, 2001 10:31:19 AM
<list>
<group name="Boolean">
 <function>
  <name><![CDATA[@ISACCTYPE]]
     ></name>
  <syntax>
   <![CDATA[@ISACCTYPE(tag)]]
     >
  </syntax>
  <comment>
   <![CDATA[returns TRUE if the current member has the associated accounts tag]]
     >
  </comment>
 </function>
```

```
</group>
<group name="Relationship Functions">
 <function>
  <name><![CDATA[@ANCESTVAL]]
      ></name>
  <syntax>
   <![CDATA[@ANCESTVAL (dimName, genLevNum [, mbrName])]]
      >
  </syntax>
  <comment>
   <![CDATA[returns the ancestor values of a specified member combination]]
      >
  </comment>
</group>
<group name="Custom">
</group>

</list>
```

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    #include <iostream.h>
#include <fstream.h>
#include "windows.h"
#include "essbase.h"
#include "essapi.h"
#include "essotl.h"
#include "stdio.h"

/* globals - handles to different ESS objects */

ESS_HINST_T   hInst = 0;
ESS_HCTX_T    hCtx = 0;

ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;

ESS_HOUTLINE_T  hOutline = 0;

/* end globals */

/* forward declarations of functions */
void apiInit();
void apiTerm();

ESS_STS_T apiAutoLogin();
ESS_STS_T apiLogout();
/* end forward declarations */

ESS_FUNC_M   MessageHandler (ESS_PVOID_T   UserContext,   /* user context pointer */
              ESS_LONG_T    MessageNumber,  /* Essbase message number */
              ESS_USHORT_T   Level,      /* message level */
```

```
                ESS_STR_T    LogString,    /* message log string */
                ESS_STR_T    MessageString   /* message string */)
{
 printf( "%d: %s\n", MessageNumber, MessageString );
 return 0;
}

void apiInit()
{
 ESS_STS_T  sts;
 ESS_INIT_T InitStruct = {
  ESS_API_VERSION,
  NULL,
  0L,
  255,
  NULL,
  NULL,
  NULL,
  NULL,
  NULL,
  (ESS_PFUNC_T)MessageHandler,
  NULL,
  0L
 };

 printf( "ESSBASE API v.%x\n", ESS_API_VERSION );

 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {
  printf( "API init failure: %d\n", sts);
  exit(1);
 }

}

ESS_STS_T apiAutoLogin ()
{
 ESS_CHAR_T SvrName[ESS_SVRNAMELEN];
 ESS_CHAR_T UserName[ESS_USERNAMELEN];
 ESS_CHAR_T Password[ESS_PASSWORDLEN];

 ESS_USHORT_T  Option;
 ESS_ACCESS_T  Access ;

 /* Initialize parameters */
 strcpy(SvrName,"localhost");
 strcpy(UserName, "");
 strcpy(Password, "");
 strcpy(szAppName, "");
 strcpy(szDbName, "");

 Option = AUTO_DEFAULT;

 /* Login to Essbase Server */
 return EssAutoLogin (hInst, SvrName, UserName, Password,
  szAppName, szDbName, Option, &Access, &hCtx);
```

```
}

void apiTerm()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 if ( hCtx )
  sts = apiLogout();

 if ( !sts && hInst )
  sts = EssTerm(hInst);

 if ( sts )
 {
  printf( "API shutdown failure: %d\n", sts );
  exit(1);
 }
}

ESS_STS_T apiLogout()
{
 return EssLogout(hCtx);
}

int main(int argc, char **argv)
{
 ESS_STS_T    status;
 ESS_STR_T    pszCalcFunctionList;

 apiInit();

 status = apiAutoLogin();
 if ( status )
  return 1;

 status = EssListCalcFunctions( hCtx, &pszCalcFunctionList );
 if ( status )
  return 1;

 printf( "%s\n", pszCalcFunctionList );

 apiTerm();

 return 0;
}
```

関連トピック

- EssGetFilterList

EssListConnections

現在ログインしているサーバーまたはアプリケーションに接続されているユーザーをすべてリストします。

構文

```
ESS_FUNC_M EssListConnections (
hCtx, pCount, ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数。
ppUserList	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	ユーザー情報構造体の配列へのポインタ。この配列は API によって割り当てられます。

備考

- pCount には、ppUserList 配列内の要素の数が含まれます。
- hCtx がスーパーバイザの場合、ppUserList はサーバーにログインしているユーザーのリストになります。hCtx がアプリケーション・デザイナの場合、ppUserList は hCtx がアプリケーション・デザイナになっているアプリケーションに接続しているユーザーのリストになります。
- ppUserList に割り当てられたバッファを解放するには、EssFree を使用します。

戻り値

成功の場合、0 が戻されます。

アクセス

この関数を使用するには、呼出し元がスーパーバイザまたはアプリケーション・デザイナ権限を持っている必要があります。

例

```
    ESS_FUNC_M ESS_ListUserConnections (ESS_HCTX_T    hCtx, ESS_HINST_T hInst)
{
    ESS_FUNC_M     sts = ESS_STS_NOERR;
    ESS_USHORT_T   usrcnt;
    ESS_PUSERINFO_T  users;
    sts = EssListConnections(hCtx, &usrcnt,   &users);
    if(!sts)
    EssFree(hInst, users);
    return(sts);
}
```

関連トピック

- EssListConnectionsEx

EssListConnectionsEx

現在ログインしているサーバーまたはアプリケーションに接続されているユーザーをすべてリストします。EssListConnections に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。

構文

```
ESS_FUNC_M EssListConnectionsEx (
hCtx
,
pCount
,
ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス(出力)。
ppUserList	ESS_PPUSERINFOID_T	ユーザー情報構造体の配列へのポインタ(出力)。情報構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

- pCount には、ppUserList 配列内の要素の数が含まれます。
- hCtx がスーパーバイザの場合、ppUserList はサーバーにログインしているユーザーのリストになります。hCtx がアプリケーション・デザイナの場合、ppUserList は hCtx がアプリケーション・デザイナになっているアプリケーションに接続しているユーザーのリストになります。
- ppUserList に割り当てられたバッファを解放するには、EssFree を使用します。

戻り値

成功の場合、0 が戻されます。

アクセス

この関数を使用するには、呼出し元がスーパーバイザまたはアプリケーション・デザイナ権限を持っている必要があります。

例

```
void DisplayUserInfoID2(ESS_USERINFOID_T userInfo)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_BOOL_T    isDefined = ESS_TRUE;

 printf("\tUser Name: %s\n", userInfo.Name);
 printf("\tProvider Name: %s\n", userInfo.ProviderName);
 printf("\tConnparam: %s\n", userInfo.connparam);
```

```
printf("\tDescription: %s\n", userInfo.Description);
printf("\tEMail Identification: %s\n", userInfo.EMailID);

if (userInfo.LockedOut)
 printf("\tLocked out: Yes\n");
else
 printf("\tLocked out: No\n");

if (userInfo.PwdChgNow)
 printf("\tChange the password now: Yes\n");
else
 printf("\tChange the password now: No\n");

printf(  "\tConnected Application: %s\n", userInfo.AppName);
printf(  "\tConnected Database: %s\n",  userInfo.DbName);

if (userInfo.Login)
 printf("\tLogged in: Yes\n");
else
 printf("\tLogged in: No\n");

switch(userInfo.Access)
{
 case ESS_ACCESS_ADMIN:
  printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPALL:
  printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBALL:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userInfo.Access);
  break;
 case ESS_ACCESS_CALC:
  printf("\tAccess: %d - ESS_ACCESS_CALC\n", userInfo.Access);
  break;
 case ESS_ACCESS_WRITE:
  printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userInfo.Access);
  break;
 case ESS_ACCESS_READ:
  printf("\tAccess: %d - ESS_ACCESS_READ\n", userInfo.Access);
  break;
 case ESS_PRIV_USERCREATE:
  printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userInfo.Access);
  break;
 case ESS_PRIV_APPCREATE:
```

```
   printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userInfo.Access);
   break;
  case ESS_PRIV_APPMANAGE:
   printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userInfo.Access);
   break;
  case ESS_PRIV_APPLOAD:
   printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userInfo.Access);
   break;
  case ESS_PRIV_DBCREATE:
   printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userInfo.Access);
   break;
  case ESS_PRIV_DBMANAGE:
   printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userInfo.Access);
   break;
  case ESS_PRIV_DBLOAD:
   printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userInfo.Access);
   break;
  case ESS_PRIV_CALC:
   printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
   break;
  case ESS_PRIV_WRITE:
   printf("\tAccess: %d - ESS_PRIV_WRITE\n", userInfo.Access);
   break;
  case ESS_PRIV_READ:
   printf("\tAccess: %d - ESS_PRIV_READ\n", userInfo.Access);
   break;
  case ESS_PRIV_NONE:
   printf("\tAccess: %d - ESS_PRIV_NONE\n", userInfo.Access);
   break;
  default:
   printf("\tAccess: Unknown\n");
}

switch(userInfo.MaxAccess)
{
  case ESS_ACCESS_ADMIN:
   printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPALL:
   printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBALL:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_CALC:
```

```
    printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userInfo.MaxAccess);
    break;
   case ESS_ACCESS_WRITE:
    printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userInfo.MaxAccess);
    break;
   case ESS_ACCESS_READ:
    printf("\tMax Access: %d - ESS_ACCESS_READ\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_USERCREATE:
    printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_APPCREATE:
    printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_APPMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_APPLOAD:
    printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_DBCREATE:
    printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_DBMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", userInfo.MaxAccess);
    break;
   default:
    printf("\tMax Access: Unknown\n");
  }

  printf("\tPassword Expiration in Dates: %d\n",userInfo.Expiration);
  //EssSdCTime(NULL, userInfo.LastLogin, sizeof(time_string), time_string);
  //printf("\tLast Successful Login:      %s\n", time_string);
  printf("\tFailed Login Attempts Since Then: %d\n", userInfo.FailCount);
  printf("\tLogin ID: %ld\n", userInfo.LoginId);
  printf( "\n");
}


ESS_FUNC_M ESS_ListUserConnectionsEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
```

```
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_HCTX_T hLocalCtx1;
 ESS_HCTX_T hLocalCtx2;
 ESS_USHORT_T usercount, i = 0;
 ESS_PUSERINFOID_T userInfo = ESS_NULL;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = ESS_NULL;


 usercount = 0;
 memset(&userInfo, '\0', sizeof(userInfo));
 sts = EssListConnectionsEx(hCtx, &usercount, &userInfo);
 printf("EssListConnectionsEx sts: %ld\n", sts);
 if(!sts)
 {
  printf("\nConnection count(s): %d\n", usercount);
  for(i = 0; i < usercount; i++)
  {
   DisplayUserInfoID2(userInfo[i]);
  }
 }



   return(sts);
}
```

EssListCurrencyDatabases

呼出し元がアクセス可能な、特定のアプリケーション内のすべての通貨データベースをリストします。

構文

```
ESS_FUNC_M EssListCurrencyDatabases (
hCtx, AppName, pCount, ppDbList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
pCount	ESS_PUSHORT_T	通貨データベースのカウントを受け取る変数のアドレス。
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース構造体の配列を受け取るポインタのアドレス。

備考

- この関数は、クライアントではなくサーバーのアプリケーション内部の通貨データベースをリストする場合にのみ使用できます。

- ppDbList 引数は、アプリケーションおよびデータベース名文字列の一致する対を含む構造体の配列を戻します。
- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、アクセス可能な通貨データベース数のカウントが pCount に、アプリケーションおよび通貨データベース名のリストが ppDbList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアクセスできる場合、サーバーの通貨データベースのみがリストされることに注意してください。

例

```
    ESS_FUNC_M
ESS_ListCurrencyDatabases (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_USHORT_T  Items;
 ESS_USHORT_T  ind;
 ESS_STR_T   AppName;
 ESS_PAPPDB_T  pDbsList = NULL;
 AppName = "Sample";
 sts = EssListCurrencyDatabases(hCtx, AppName,
    &Items, &pDbsList);

if(!sts)
 {
  if(Items && pDbsList)
  {
   printf("\r\n---- Currency Databases ----\r\n\r\n");
   for (ind = 0; ind<Items; ind++)
   {
    if((pDbsList+ind) !=NULL)
    {
     if(pDbsList[ind].DbName != NULL)
     {
      printf("%s",AppName);
      printf(" ==> ");
      printf("%s",pDbsList[ind].DbName);
      printf("\n\r");
     }
    }
   }
   EssFree(hInst, pDbsList);
  }
  else
   printf("\r\nCurrency Database List is Empty\r\n\r\n");
 }
 return (sts);
}
```

関連トピック

- EssGetDatabaseInfo
- EssGetDatabaseState
- EssListApplications
- EssListDatabases
- EssListObjects

EssListDatabases

呼出し元がアクセス可能な、特定のアプリケーション内またはサーバー全体の、すべてのデータベースをリストします。

構文

```
ESS_FUNC_M EssListDatabases (
hCtx, AppName, pCount, ppDbList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
pCount	ESS_PUSHORT_T	アプリケーションおよびデータベースのカウントを受け取る変数のアドレス。
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース名構造体の配列を受け取るポインタのアドレス。

備考

- AppName 引数が NULL の場合は、この関数はサーバー上のアクセス可能なアプリケーションとデータベースをすべてリストします。
- ppDbList 引数は、アプリケーションおよびデータベース名文字列の一致する対を含む構造体の配列を戻します。
- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、アクセス可能なデータベース数のカウントが pCount に、アプリケーションおよびデータベース名のリストが ppDbList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアクセスする場合、サーバー・データベースのみがリストされることに注意してください。

例

```
ESS_FUNC_M
```

```
ESS_ListDbs (ESS_HCTX_T  hCtx,
      ESS_HINST_T hInst
     )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_USHORT_T  Items;
 ESS_USHORT_T  ind;
 ESS_PAPPDB_T  pAppsDbs = NULL;
 sts = EssListDatabases (hCtx, NULL, &Items,
    &pAppsDbs);
 if (!sts)
 {
  if (Items && pAppsDbs)
  {
printf ("\r\n-----Applications/databases available-----\r\n");
    for (ind = 0; ind < Items; ind++)
    {
     if ((pAppsDbs+ind) != NULL)
      {
       if ((pAppsDbs[ind].AppName != NULL) &&
          (pAppsDbs[ind].DbName  != NULL))
        {
printf ("%s", pAppsDbs[ind].AppName);
printf (" ==> ");
printf ("%s", pAppsDbs[ind].DbName);
printf ("\n\r");
        }
      }
    }
    EssFree (hInst, pAppsDbs);
  }
  else
printf ("\r\ndatabaseList is Empty\r\n\r\n");
 }

 return(sts);
}
```

関連トピック

- EssGetDatabaseInfo
- EssGetDatabaseState
- EssListApplications
- EssListCurrencyDatabases
- EssListObjects

EssListExistingLoadBuffers

集約ストレージ・データベースの既存のデータ・ロード・バッファを記述する構造体のリストを戻します。

この関数は既存のバッファ数および記述子構造体の配列を戻します。配列のメモリーは EssFree を使用して解放する必要があります。

構文

```
ESS_FUNC_M EssListExistingLoadBuffers (
hCtx, AppName, DbName, pCount, paLoadBuffers
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	NULL を使用します。関数は常に現在選択されているデータベースに適用されます。
DbName	ESS_STR_T	NULL を使用します。関数は常に現在選択されているデータベースに適用されます。
pCount	ESS_PULONG_T	ロード・バッファのカウントを受け取る変数のアドレス。
paLoadBuffers	153 ページの「ESS_LOAD_BUFFER_T」 **	ロード・バッファ情報構造体へのポインタ。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

例

```
    void TestListExistingLoadBuffers(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T
DbName)
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_LOAD_BUFFER_T *LoadBuffers;
    ESS_ULONG_T i;
    ESS_ULONG_T   Count;

    /* EssListExistingLoadBuffers */
    sts = EssListExistingLoadBuffers(hCtx, AppName, DbName, &Count, &LoadBuffers);
    printf("EssListExistingLoadBuffers sts: %ld\n",sts);
    printf("\tNumber of buffers: %d", Count);
    if(Count > 0)
    {
        for(i = 0; i < Count; i++)
        {
               printf("\n\tBuffer Id: %d", LoadBuffers[i].ulBufferId);
               printf("\n\tDuplicate Agg Method: %d",
LoadBuffers[i].ulDuplicateAggregationMethod);
               printf("\n\tOption Flags: %d", LoadBuffers[i].ulOptionFlags);
               printf("\n\tSize (1-100): %d", LoadBuffers[i].ulSize);
               printf("\n\tInternal: %d", LoadBuffers[i].bInternal);
               printf("\n\tActive: %d", LoadBuffers[i].bActive);
               printf("\n");
        }
    }
}
```

関連トピック

- EssLoadBufferInit
- EssBeginDataloadASO
- EssSendString
- EssEndDataload
- EssLoadBufferTerm
- EssImportASO
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssMergeDatabaseData

EssListDbFiles

指定したインデックスおよびデータ・ファイルに関する情報を取得します。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
szAppName;	ESS_STR_T	アプリケーション名。
szDbName;	ESS_STR_T	データベース名。
usFileType;	ESS_USHORT_T	戻される次のファイル・タイプのいずれかになります: ● ESS_FILETYPE_INDEX ● ESS_FILETYPE_DATA ● ESS_FILETYPE_INDEX | ESS_FILETYPE_DATA
pNmbrOfFiles;	ESS_PUSHORT_T	戻されるインデックス・ファイルとデータ・ファイルの数へのポインタ。
ppDbInfoArray;	129 ページの「ESS_DBFILEINFO_T」	戻されるデータベース・ファイル情報構造体の配列へのポインタ。

戻り値

- 成功の場合、
- この関数は 0 を戻します
- pNmbrOfFiles は、戻されたインデックス・ファイルとデータ・ファイルの数へのポインタを含みます
- ppDbInfoArray は、戻されたデータベース・ファイル情報構造体の配列へのポインタを含みます

例

```
    ESS_STS_T ListDbFiles( ESS_HCTX_T hCtx )
{
 ESS_STS_T     sts     = ESS_STS_NOERR;
```

```
ESS_APPNAME_T   pszAppName;
ESS_DBNAME_T    pszDbName;
ESS_PDBFILEINFO_T aDbFileInfo = NULL;
ESS_PDBFILEINFO_T pDbFileInfo = NULL;
ESS_USHORT_T    usFileType  = ( ESS_FILETYPE_INDEX | ESS_FILETYPE_DATA );
ESS_USHORT_T    usFileCount = 0;
ESS_USHORT_T    usFileIx;

/*************************************************************
* Prompt for the type of files to list: index, data or both, *
* and assign the user's file type choice to usFileType    *
*************************************************************
* This function uses ESS_FILETYPE_INDEX | ESS_FILETYPE_DATA  *
* as the default value                  *
*************************************************************/
             .
             .
             .

/*************************************************************
* Prompt for application and database names, and assign the *
* user's choices to pszAppName and pszDbName, respectively  *
*************************************************************/
             .
             .
             .

/*************************************************************************
* Get an array of persistent database file information from Essbase *
* for the selected file type, application and database       *
*************************************************************************/
sts = EssListDbFiles( hCtx, pszAppName, pszDbName, usFileType,
          &usFileCount, &aDbFileInfo );
if ( sts )
{
 goto exit;
}

/*********************************************
* Format and display the information in the  *
* persistent database file information array *
*********************************************/
if ( ( usFileCount ) && ( aDbFileInfo ) )
{
 printf( "Application Name:   %s\n",
     aDbFileInfo[ 0 ].AppName );

 printf( "Database Name:     %s\n",
     aDbFileInfo[ 0 ].DbName  );

 for ( ( usFileIx = 0, usFileType = 0 );
    usFileIx < usFileCount;
    usFileIx++ )
 {
   /*************************************************
    * Format and display the information in the current  *
    * persistent database file information array element *
```

```
    ***************************************************/
    pDbFileInfo = &( aDbFileInfo[ usFileIx ] );

    printf( "\nFile %lu:\n",
        pDbFileInfo->FileSequenceNum );

    printf( "  File Name:     %s\n",
        pDbFileInfo->FilePath );

    printf( "  File Type:     " );

        if ( pDbFileInfo->FileType == ESS_FILETYPE_INDEX )
    {
     printf( "INDEX\n" );
    }
    else
    {
     printf( "DATA\n" );
    }

    printf( "  File Number:    %lu of %lu\n",
         pDbFileInfo->FileSequenceNum, pDbFileInfo->FileCount );

    printf( "  File Size:     %lu\n",
        pDbFileInfo->FileSize );

    printf( "  File Opened:    %c\n",
         ( pDbFileInfo->FileOpen ) ? 'Y' : 'N' );
  } /* FOR usFileIx */

  /***********************************************
   * Free the memory allocated for the persistent *
   * database file information array       *
   ***********************************************/
  free( aDbFileInfo );
}

else
{
  printf( "Application Name:   %s\n",
      AppName );

  printf( "Database Name:     %s\n",
      DbName  );

  switch ( usFileType )
  {
  case ESS_FILETYPE_INDEX:
    printf( "\nNo existing INDEX files.\n" );
    break;
  case ESS_FILETYPE_DATA:
    printf( "\nNo existing DATA files.\n" );
    break;
  case ( ESS_FILETYPE_INDEX | ESS_FILETYPE_DATA ):
    printf( "\nNo existing INDEX or DATA files.\n" );
    break;
  default:
```

```
    printf( "\nNo existing database files of the selected type.\n" );
    break;
   } /* SWITCH usFileType */
  }

 printf( "\n" );

exit:

 return ( sts );
}
```

関連トピック

- 129 ページの「ESS_DBFILEINFO_T」

EssListDrillThruURLs

アクティブなデータベース・アウトライン内のドリルスルー URL 名をリストします。

1904 ページの「ドリルスルー URL の制限」を参照してください。

構文

```
ESS_FUNC_M EssListDrillThruURLs (
hCtx, &pCountOfUrls, &pUrls
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCountOfUrls	ESS_PUSHORT_T	ドリルスルー URL のカウント。
pUrls	ESS_PPDURLINFO_T	URL のリスト。

備考

呼出し側は、ESS_DT_STRUCT_URLINFO オプションを指定した EssFreeStructure を使用して ESS_DURLINFO_T 構造体の配列を割当て解除する必要があります。

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内のドリルスルー URL がリストされます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESS_PRIV_READ)を持っている必要があります。

- 呼出し側は EssSetActive を使用して、指定したデータベースをアクティブ・データベースとして選択しておく必要があります。

例

```
    static void DisplayUrlDefn (ESS_PDURLINFO_T pUrls )
{
 ESS_UINT_T    i;

 printf("\tUrlname     : %s\n", pUrls->cpURLName );
 if (pUrls->bIsLevel0)
  printf("\tUrl Is Level-0 slice : Yes\n");
 else
  printf("\tUrl Is Level-0 slice : No\n");

 printf("\tUrlXmlsize   : %i\n", pUrls->iURLXmlSize );
 printf("\tUrlXml    : %s\n", (ESS_STR_T) pUrls->cpURLXml);

 printf("\tNumber of drill region(s): %d\n", pUrls->iCountOfDrillRegions);
 for ( i = 0; i < pUrls->iCountOfDrillRegions; i++ )
 {
  printf("\t\tDrillRegion[%d]: %s\n", i, pUrls->cppDrillRegions[i] );
 }
 printf("\n");
}

ESS_STS_T sts = ESS_STS_NOERR;
ESS_USHORT_T usCountOfURLs, i;
ESS_PDURLINFO_T listOfURLs;
ESS_DURLINFO_T url;

/* Valid case*/

sts = EssListDrillThruURLs(hCtx, &usCountOfURLs, &listOfURLs);
printf("EssListDrillThruURLs sts: %ld\n",sts);
if(!sts)
 {
  printf("\tCount of URL: %d\n", usCountOfURLs);
  printf("\tList of URL(s):\n");
  for(i = 0; i < usCountOfURLs; i++)
  {
   DisplayUrlDefn (&listOfURLs[i]);
  }
 }
EssFreeStructure (hInst, ESS_DT_STRUCT_URLINFO, usCountOfURLs, listOfURLs);
```

EssListExtUsers

Shared Services を介して外部認証されるユーザーをリストします。

構文

```
    ESS_FUNC_M EssListExtUsers (hCtx, AppName, DbName, Protocol, Count,
```

```
ppUserList);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は、すべてのユーザーがリストされます。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション内のすべてのデータベースのユーザーがリストされます。
Protocol	ESS_STR_T	外部認証プロトコル: Shared Services モードの場合は CSS。プロトコルが指定されていない場合でも、この関数は、Shared Services を介して外部認証されるユーザーのリストを戻します。
Count	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス。
ppUserList	207 ページの「ESS_USERINFOEX_T」	割り当てられたユーザー情報構造体の配列を受け取るポインタのアドレス。戻されるユーザー情報構造体の AppName および DbName には NULL 値が含まれます。

備考

- AppName および DbName の両方が NULL でない場合、指定したアプリケーションとデータベースへのアクセス権を持つユーザーのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つユーザーのみがリストされます。AppName が NULL の場合、サーバー上のすべてのユーザーがリストされます。
- 戻される ESS_USERINFO_T 構造体の AppName および DbName フィールドには、NULL 値が含まれます。
- ppUserList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー数のカウントが pCount に、指定したアプリケーションおよびデータベースへのアクセス権を持つユーザーのリストが ppUserList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

関連トピック

- EssSetExtUser
- EssCreateExtUser
- EssGetExtUser

EssListFilters

データベースのすべてのフィルタをリストします。

構文

```
ESS_FUNC_M EssListFilters (
hCtx, AppName, DbName, Count, ppFilterList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pCount	ESS_PUSHORT_T	フィルタ名のカウントを受け取る変数のアドレス。
ppFilterList	ESS_PPFTRNAME_T	割り当てられたフィルタ名文字列の配列を受け取るポインタのアドレス。

備考

ppFilterList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、データベース内のフィルタのカウントが pCount に、フィルタ名の配列が ppFilterList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ListFilters (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_USHORT_T  Count = 0;
 ESS_USHORT_T  ind;
 ESS_PFTRNAME_T pFilterList = NULL;

 AppName = "Sample";
 DbName  = "Basic";

 sts = EssListFilters(hCtx, AppName, DbName,
    &Count, &pFilterList);

 if(!sts)
 {
  if(Count && pFilterList)
  {
printf ("\r\n-------Filter List-------\r\n\r\n");
```

```
    for (ind = 0; ind < Count; ind++)
printf("%s\r\n",pFilterList[ind]);
    EssFree (hInst, pFilterList);
   }
   else
printf ("\r\nFilter List is empty\r\n\r\n");
  }
  return (sts);
}
```

関連トピック

- EssGetFilter
- EssSetFilter

EssListFilterUsers

フィルタを使用するすべてのユーザーをリストします。

構文

```
ESS_FUNC_M EssListFilterUsers (
hCtx, dbName, AppName, UserCount, ppUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
dbName	ESS_STR_T	データベース名。
AppName	ESS_STR_T	アプリケーション名。
UserCount	ESS_PUSHORT_T	このフィルタを使用するユーザーのカウント。
ppUserInfo	ESS_PPUSERINFO_T（see 204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」）	ユーザー情報構造体の配列へのポインタ。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

関連トピック

- EssListFilters

EssListGroups

ある特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセス権を持つすべてのグループをリストします。

構文

```
ESS_FUNC_M EssListGroups (
```

```
    hCtx, AppName, DbName, pCount, ppGroupList
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は全グループがリストされます。
DbName	ESS_STR_T	データベース名。NULL の場合はアプリケーション内のすべてのデータベースのグループがリストされます。
pCount	ESS_PUSHORT_T	グループのカウントを受け取る変数のアドレス。
ppGroupList	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	割り当てられたグループ情報構造体の配列を受け取るポインタのアドレス。

備考

- AppName と DbName の両方が NULL でない場合、指定されたアプリケーションとデータベースへのアクセス権があるグループのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つグループのみがリストされます。AppName が NULL の場合、ログオンしているサーバー上のすべてのグループがリストされます。
- ppGroupList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、グループ数のカウントが pCount に、指定されたアプリケーションおよびデータベースへのアクセス権を持つグループのリストが ppGroupList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_ListGroups (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
       )
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_USHORT_T   Count;
 ESS_PGROUPINFO_T Groups = NULL;
 ESS_USHORT_T    ind;
 sts = EssListGroups (hCtx, NULL, NULL, &Count, &Groups);
 if (!sts)
 {
  if (Count && Groups)
  {
   printf ("\r\n-----Group List-----\r\n\r\n");
   for (ind = 0; ind < Count; ind++)
```

```
    printf ("Name->%s\r\n", Groups [ind].Name);
   EssFree (hInst, Groups);
  }
  else
   printf ("\r\nGroup List is Empty\r\n\r\n");
 }

 return (sts);
}
```

関連トピック

- EssListGroupsInfoEx
- EssGetGroup
- EssListUsers

EssListGroupsInfoEx

ある特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセス権を持つすべてのグループをリストします。EssListGroups に似ていますが、グループ・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

構文

```
ESS_FUNC_M EssListGroupsInfoEx (
hCtx,

AppName
,
DbName
,
pCount
,
ppGroupList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
AppName	ESS_STR_T	アプリケーション名(入力)。NULL の場合は全グループがリストされます。
DbName	ESS_STR_T	データベース名(入力)。NULL の場合は、アプリケーション内のすべてのデータベースのグループがリストされます。
pCount	ESS_PUSHORT_T	グループのカウントを受け取る変数のアドレス(出力)。
ppGroupList	ESS_PPGROUPINFOID_T	割り当てられたグループ情報構造体の配列を受け取るポインタのアドレス(出力)。グループ・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

- AppName と DbName の両方が NULL でない場合、指定されたアプリケーションとデータベースへのアクセス権があるグループのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つグループのみがリストされます。AppName が NULL の場合、ログオンしているサーバー上のすべてのグループがリストされます。
- ppGroupList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合、グループ数のカウントが pCount に、指定されたアプリケーションおよびデータベースへのアクセス権を持つグループのリストが ppGroupList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
void DisplayGroupsInfoEx(ESS_GROUPINFOID_T groupInfo)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_BOOL_T isDefined = ESS_TRUE;

 printf("\tUser Name: %s\n", groupInfo.Name);
 printf("\tProvider Name: %s\n", groupInfo.ProviderName);
 printf("\tIdentity: %s\n", groupInfo.connparam);
 printf("\tDescription: %s\n", groupInfo.Description);
 printf("\tEMail Identification: %s\n", groupInfo.EMailID);

 if (groupInfo.LockedOut)
  printf("\tLocked out: Yes\n");
 else
  printf("\tLocked out: No\n");

 if (groupInfo.PwdChgNow)
  printf("\tChange the password now: Yes\n");
 else
  printf("\tChange the password now: No\n");

 printf("\tPassword: %s\n", groupInfo.Password);
 printf("\tApplication: %s\n", groupInfo.AppName);
 printf("\tDatabase: %s\n", groupInfo.DbName);

 if (groupInfo.Login)
  printf("\tLogged in: Yes\n");
 else
  printf("\tLogged in: No\n");

 switch(groupInfo.Access)
 {
  case ESS_ACCESS_ADMIN:
```

```
 printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", groupInfo.Access);
 break;
case ESS_ACCESS_APPALL:
 printf("\tAccess: %d - ESS_ACCESS_APPALL\n", groupInfo.Access);
 break;
case ESS_ACCESS_DBALL:
 printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
 break;
case ESS_ACCESS_APPCREATE:
 printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", groupInfo.Access);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", groupInfo.Access);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", groupInfo.Access);
 break;
case ESS_ACCESS_CALC:
 printf("\tAccess: %d - ESS_ACCESS_CALC\n", groupInfo.Access);
 break;
case ESS_ACCESS_WRITE:
 printf("\tAccess: %d - ESS_ACCESS_WRITE\n", groupInfo.Access);
 break;
case ESS_ACCESS_READ:
 printf("\tAccess: %d - ESS_ACCESS_READ\n", groupInfo.Access);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", groupInfo.Access);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", groupInfo.Access);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", groupInfo.Access);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", groupInfo.Access);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", groupInfo.Access);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", groupInfo.Access);
 break;
case ESS_PRIV_DBLOAD:
 printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", groupInfo.Access);
 break;
case ESS_PRIV_CALC:
 printf("\tAccess: %d - ESS_ACCESS_DBALL\n", groupInfo.Access);
 break;
case ESS_PRIV_WRITE:
 printf("\tAccess: %d - ESS_PRIV_WRITE\n", groupInfo.Access);
 break;
case ESS_PRIV_READ:
```

```
   printf("\tAccess: %d - ESS_PRIV_READ\n", groupInfo.Access);
   break;
  case ESS_PRIV_NONE:
   printf("\tAccess: %d - ESS_PRIV_NONE\n", groupInfo.Access);
   break;
  default:
   printf("\tAccess: Unknown\n");
 }

 switch(groupInfo.MaxAccess)
 {
  case ESS_ACCESS_ADMIN:
   printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPALL:
   printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBALL:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_CALC:
   printf("\tMax Access: %d - ESS_ACCESS_CALC\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_WRITE:
   printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", groupInfo.MaxAccess);
   break;
  case ESS_ACCESS_READ:
   printf("\tMax Access: %d - ESS_ACCESS_READ\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_USERCREATE:
   printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPCREATE:
   printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPMANAGE:
   printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_APPLOAD:
   printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_DBCREATE:
   printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", groupInfo.MaxAccess);
   break;
  case ESS_PRIV_DBMANAGE:
```

```
    printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", groupInfo.MaxAccess);
    break;
   case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", groupInfo.MaxAccess);
    break;
   default:
    printf("\tMax Access: Unknown\n");
  }

  printf("\tPassword Expiration in Dates: %d\n",groupInfo.Expiration);
  printf("\tFailed Login Attempts Since Then: %d\n", groupInfo.FailCount);
  printf("\tLogin ID: %d\n", groupInfo.LoginId);
  printf("\tProtocol: %s\n", groupInfo.protocol);
  printf("\tConnection Parameter: %s\n", groupInfo.connparam);
  printf( "\n");

}

ESS_FUNC_M ESS_ListGroupsInfoEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{

  ESS_STS_T sts  = ESS_STS_NOERR;
  ESS_USHORT_T count, i;
  ESS_PGROUPINFOID_T pGroupList = ESS_NULL;

  sts = EssListGroupsInfoEx(hCtx, AppName, DbName, &count, &pGroupList);
  printf("EssListGroupsInfoEx sts: %ld\n", sts);
  if(!sts)
  {
   printf("\nNumber of group(s): %d\n", count);
   for(i = 0; i < count; i++)
   {
    DisplayGroupsInfoEx(pGroupList[i]);
   }
  }

 return (sts);
}
```

関連トピック

- EssGetGroupInfoEx
- EssListUsersInfoEx

EssListLocks

ある特定のアプリケーションとデータベースに接続しているすべてのユーザーを、それらのユーザーが現在ロックしているデータ・ブロックのカウントとともにリストします。

構文

```
ESS_FUNC_M EssListLocks (
hCtx, AppName, DbName, pCount, ppLockList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス。
ppLockList	154 ページの「ESS_LOCKINFO_T」	割り当てられたユーザー・ロック情報構造体の配列を受け取るポインタのアドレス。

備考

- この関数は、この関数が呼び出されたときにサーバーに接続していたユーザーのみがリストされる場合にスナップショットになります。
- ppLockList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、接続しているユーザー数のカウントが pCount に、ユーザー・ロック構造体のリストが ppLockList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ListLocks (ESS_HCTX_T  hCtx,
       ESS_HINST_T  hInst
      )
{
 ESS_FUNC_M     sts;
 ESS_USHORT_T   Count;
 ESS_PLOCKINFO_T  plockinfo = NULL;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";
```

```
 sts = EssListLocks (hCtx, AppName, DbName,
    &Count, &plockinfo);
 if (!sts)
 {
  if (Count && plockinfo)
    EssFree (hInst, plockinfo);
  else
    printf ("\r\nExclusive Lock List on %s:%s is empty\r\n\r\n", AppName, DbName);
 }
 return (sts);
}
```

関連トピック

- EssListLocksEx
- EssListConnections
- EssListUsers
- EssRemoveLocks

EssListLocksEx

ある特定のアプリケーションとデータベースに接続しているすべてのユーザーを、それらのユーザーが現在ロックしているデータ・ブロックのカウントとともにリストします。EssListLocks に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。

構文

```
ESS_FUNC_M EssListLocksEx (
hCtx
,
AppName
,
DbName
,
pCount
,
ppLockList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
AppName	ESS_STR_T	アプリケーション名(入力)。
DbName	ESS_STR_T	データベース名(入力)。
pCount	ESS_PUSHORT_T	ユーザー・カウントを受け取る変数のアドレス(出力)。
ppLockList	ESS_PPLOCKINFOEX_T	割り当てられたユーザー・ロック情報構造体の配列を受け取るポインタのアドレス(出力)。情報構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

- この関数はスナップショットです。この関数が呼び出されたときにサーバーに接続していたユーザーのみがリストされます。
- ppLockList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、接続しているユーザー数のカウントが pCount に、ユーザー・ロック構造体のリストが ppLockList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void DisplayLock(ESS_LOCKINFOEX_T lockinfo)
{
 ESS_STS_T sts = ESS_STS_NOERR;

 printf("\tUser Name: %s\n", lockinfo.UserName);
 printf("\tProvider Name: %s\n", lockinfo.ProviderName);
 printf("\tConnection Parameter: %s\n", lockinfo.connparam);
 printf("\tNumber of Locks: %d\n", lockinfo.nLocks);
 printf("\tTime: %ld\n", lockinfo.Time);
 printf("\tLoginId: %ld\n", lockinfo.LoginId);
 printf("\n");
}


ESS_FUNC_M ESS_ListLocksEx (ESS_HCTX_T hCtx, ESS_HINST_T  hInst)

{
 ESS_STS_T sts;
 ESS_USHORT_T count, i;
 ESS_PLOCKINFOEX_T plockinfo = NULL;
 ESS_ACCESS_T Access;


 sts = EssSetActive(hCtx, AppName, DbName, &Access);
 printf("EssSetActive sts: %ld\n",sts);

 sts = EssListLocksEx (hCtx, AppName, DbName, &count, &plockinfo);
 printf("EssListLocksEx sts: %ld\n", sts);
 if(!sts)
 {
  printf("\nNumber of lock info returned: %d\n", count);
  for(i = 0; i < count; i++)
  {
   DisplayLock(plockinfo[i]);
  }
 }
 return (sts);
```

```
}
```

関連トピック

- EssListConnectionsEx
- EssListUsersInfoEx

EssListLogins

現在のセッションのログイン・インスタンスのリストを戻します。

構文

```
ESS_FUNC_M EssListLogins (
hCtx, count, logins
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
count	ESS_PUSHORT_T	サーバーから戻されるログイン回数へのポインタ。
logins	ESS_PPCONNECTINFO_T	接続情報を含む ESS_CONNECTINFO_T 構造体の配列へのポインタ。

備考

同じユーザー名とサーバーに対してこの関数を複数回呼び出せます。API は指定されたサーバーへの各ログインに一意のコンテキスト・ハンドルを戻します。

戻り値

正常終了の場合は、ログイン情報と現在のログイン・カウントが戻されます。

アクセス

この関数を呼び出す前に、EssInit を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

関連トピック

- EssListLoginsEx
- EssAutoLogin
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssListLoginsEx

現在のセッションのログイン・インスタンスのリストを戻します。EssListLogins に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。

構文

```
ESS_FUNC_M EssListLoginsEx (
hCtx
,
count
,
logins
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
count	ESS_PUSHORT_T	サーバーから戻されるログイン回数へのポインタ(出力)。
logins	ESS_PPCONNECTINFOEX_T	接続情報を含む ESS_CONNECTINFOEX_T 構造体の配列へのポインタ(出力)。情報構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

備考

同じユーザー名とサーバーに対してこの関数を複数回呼び出せます。API は指定されたサーバーへの各ログインに一意のコンテキスト・ハンドルを戻します。

戻り値

正常終了の場合は、ログイン情報と現在のログイン・カウントが戻されます。

アクセス

この関数を呼び出す前に、EssInit を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

例

```
void DisplayLoginInfo(ESS_CONNECTINFOEX_T login)
{
 ESS_STS_T sts = ESS_STS_NOERR;

 printf("\tName: %s\n", login.Name);
 printf("\tApp Name: %s\n", login.AppName);
 printf("\tDb Name: %s\n", login.DbName);
 printf("\tLogin MachineName: %s\n", login.LoginMachine);
 printf("\tLogin Ip: %ld\n", login.LoginIP);
 printf("\tLast login time: %ld\n", login.LastLogin);
 printf("\n");
}


ESS_FUNC_M ESS_ListLoginsEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_HCTX_T hLocalCtx1;
```

```
 ESS_HCTX_T hLocalCtx2;
 ESS_USHORT_T count, i;
 ESS_PCONNECTINFOEX_T logins;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = ESS_NULL;
 ESS_ACCESS_T Access;


 sts = EssListLoginsEx(hCtx, &count, &logins);
 printf("EssListLogins sts: %ld\n", sts);
 if(!sts)
 {
  printf("\nConnection count(s): %d\n", count);
  for(i = 0; i < count; i++)
  {
   DisplayLoginInfo(logins[i]);
  }

  sts = EssFree (hInst, logins);
  printf("EssFree sts: %ld\n", sts);
 }



   return(sts);
}
```

関連トピック

- EssAutoLogin
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssListObjects

サーバーまたはローカル・クライアント上にある、指定したタイプのオブジェクトをリストします。

構文

```
ESS_FUNC_M EssListObjects (
hCtx, ObjType, AppName, DbName, pCount, ppObjList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。

パラメータ	データ型	説明
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(ビット OR（｜）で結合した複数のタイプ可)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリ内のオブジェクトがリストされます。
pCount	ESS_PUSHORT_T	適切なタイプのオブジェクトのカウントを受け取る変数のアドレス。
ppObjList	160 ページの「ESS_OBJINFO_T」	割り当てられたオブジェクト情報構造体の配列を受け取るポインタのアドレス。

備考

- ppObjList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- この関数では、戻されるオブジェクトの一貫性のある順序はオペレーティング・システムによって異なるため保証されません。

戻り値

正常終了の場合は、適切なタイプのオブジェクト数のカウントが pCount に、一致するオブジェクト構造体の配列が ppObjList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアプリケーションまたはデータベース(あるいはその両方)に対して(オブジェクト・タイプに応じて)適切なレベルのアクセス権限を持っている場合、サーバー・オブジェクトのみがリストされるので注意してください。

例

```
    ESS_FUNC_M
ESS_ListObjects (ESS_HCTX_T  hCtx,
        ESS_HINST_T hInst
        )
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_POBJINFO_T  pObject, pNextObject = NULL;
 ESS_SHORT_T    objType = 0;
 ESS_USHORT_T    objCnt;
 ESS_USHORT_T    objInd;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 Appname = "Sample";
 DbName  = "Basic";
 objType = ESS_OBJTYPE_OUTLINE;
 sts = EssListObjects (hCtx, objType, AppName,
    DbName, &objCnt, &pObject);
 if (!sts)
 {
  if (objCnt && pObject)
```

```
   {
    pNextObject = pObject;
    for (objInd = 0; objInd < objCnt; objInd++)
    {
      if (pNextObject)
      {
printf ("Name: %s \r\nUser: %s\r\nTime Stamp: %ld\r\n",
           pNextObject->Name,
           pNextObject->User,
           pNextObject->TimeStamp);

       pNextObject = pNextObject + 1;
      }
    }
    EssFree (hInst, pObject);
   }
   else
printf ("\r\nObject List is Empty\r\n\r\n");
  }
  return(sts);
}
```

関連トピック

- EssGetObject
- EssGetObjectInfo

EssListRequests

アクティブなセッションおよび要求に関する情報を戻します。

構文

```
ESS_FUNC_M EssListRequests (
hCtx, UserName, AppName, DbName, RequestCount, pRequestInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
RequestCount	ESS_PUSHORT_T	要求の数(出力)。
ppRequestInfoStruct	ESS_PPREQUESTINFO_T	要求のタイプ(出力)。

備考

- セッションとは、ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。

- 要求とは、ユーザーまたは他のプロセスが Essbase に対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。
- リストされた要求の中には終了済にもかかわらず、ネットワークの遅延によってアクティブとしてリストされたままのものもあります。
- この関数によって、UserName、AppName および DbName によって起動された要求およびセッションに関する情報が戻されます。これらのパラメータが NULL または空の場合、システム内のすべてのプロセスがリストされます。この関数は現行の要求数と、要求ごとに 1 つの ESS_REQUESTINFO_T 構造体を戻します。
- 戻される ppRequestInfoStruct は、EssFree を呼び出して解放する必要があります。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    #include <stdio.h>
#include <essapi.h>

ESS_FUNC_M ESS_ListRequest ()
{
  ESS_FUNC_M    sts   = ESS_STS_NOERR;
  ESS_STR_T   rString = NULL;
  ESS_HCTX_T  hCtx;
  ESS_USHORT_T Items;
  ESS_PAPPDB_T pAppsDbs = NULL;
  ESS_HINST_T  hInst ;
  ESS_ACCESS_T Access;
  ESS_USHORT_T numRequest;
  ESS_PREQUESTINFO_T requestInfo;

  ESS_INIT_T InitStruct =    /* Define init */
         /* structure */
  {
    ESS_API_VERSION,    /* Version of API */
    NULL,    /* user-defined message context */
    0,     /* max handles   */
    0L,     /* max buffer size */
    NULL, /* local path */
    /* The following parameters use defaults  */
    NULL,    /* message db path */
    NULL,    /* allocation function pointer */
    NULL,    /* reallocation function pointer */
    NULL,    /* free function pointer */
    NULL,  /* error handling function pointer */
    NULL,    /* path name of user-defined */
       /* Application help file */
    NULL,    /* Reserved for internal use. */
       /* Set to NULL */
  };
```

```
  EssInit (&InitStruct, &hInst);

  sts = EssLogin (hInst, "local", "admin", "password", &Items, &pAppsDbs, &hCtx);

  sts = EssListRequests( hCtx, NULL, NULL, NULL, &numRequest, &requestInfo);

  printf ( "Total requests on the server %d\n", numRequest );

  if ( !sts && requestInfo )
  {
    ESS_USHORT_T index = 0;

    while ( index < numRequest )
    {
      printf ( "login ID = %ul\n", requestInfo[index].LoginId );
      printf ( "user name = %s\n", requestInfo[index].UserName );
      printf ( "login machine = %s\n", requestInfo[index].LoginSourceMachine );
      printf ( "AppName = %s\n", requestInfo[index].AppName );
      printf ( "DbName = %s\n", requestInfo[index].DbName );
      printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
      printf ( "RequestString = %s\n", requestInfo[index].RequestString );
      printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
      printf ( "State = %d\n", requestInfo[index].State );
      printf ( "\n\n--------------------------------------\n\n",
requestInfo[index].State );

      sts = EssKillRequest (hCtx, &requestInfo[index] );

      index++;
    }

    EssFree ( hInst, requestInfo );
  }

  EssLogout (hCtx);
  EssTerm (hInst);
  return(sts);
}

void main()
{
  ESS_ListRequest ();
}
```

関連トピック

- EssListRequestsEx
- EssKillRequest
- 189 ページの「ESS_REQUESTINFO_T」
- 196 ページの「ESS_REQ_STATE_T」

EssListRequestsEx

アクティブなセッションおよび要求に関する情報を戻します。EssListRequests に似ていますが、ユーザー・ディレクトリにホストされているユーザーが含まれます。

構文

```
ESS_FUNC_M EssListRequestsEx (
hCtx
,
UserId
,
bIsIdentity
,
AppName
,
DbName
,
RequestCount
,
pRequestInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
UserId	ESS_STR_T	ユーザー名または ID (入力)。ID の場合は、ユーザー・ディレクトリのユーザーを識別する一意の ID 文字列が含まれます。
bIsIdentity	ESS_BOOL_T	ユーザー ID が使用されるか、ユーザー名が使用されるかを示します。TRUE の場合、UserId は一意の ID 属性です。FALSE の場合、UserId はユーザー名です。
AppName	ESS_STR_T	アプリケーション名(入力)。
DbName	ESS_STR_T	データベース名(入力)。
RequestCount	ESS_PUSHORT_T	要求の数(出力)。
pRequestInfo	ESS_PREQUESTINFOEX_T	要求のタイプ(出力)。

備考

- セッションとは、ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。
- 要求とは、ユーザーまたは他のプロセスが Essbase に対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。
- リストされた要求の中には終了済にもかかわらず、ネットワークの遅延によってアクティブとしてリストされたままのものもあります。
- この関数によって、UserID、AppName および DbName で指定されたプロセスによって起動された要求およびセッションに関する情報が戻されます。これ

らのパラメータが NULL または空の場合、システム内のすべてのプロセスがリストされます。この関数は現行の要求数と、要求ごとに 1 つの ESS_REQUESTINFOEX_T 構造体を戻します。

- 戻される ppRequestInfoStruct は、EssFree を呼び出して解放する必要があります。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void ListRequestsEx ()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_STR_T userId;
    ESS_BOOL_T bIsIdentity;
    ESS_USHORT_T numRequest;
    ESS_PREQUESTINFOEX_T requestInfo;
    ESS_USHORT_T index = 0;

    userId = "admin";
    bIsIdentity = ESS_FALSE;
    sts = EssListRequestsEx(hCtx, userId, bIsIdentity, "Sample", "Sample1",
&numRequest, &requestInfo);
    printf("\nEssListRequestsEx sts: %ld\n", sts);
    printf ( "Total requests on the server: %d\n", numRequest );
    if ( !sts && requestInfo )
    {
        while ( index < numRequest )
        {
            printf ( "login ID = %ul\n", requestInfo[index].LoginId );
            printf ( "user name = %s\n", requestInfo[index].UserName );
            printf ( "login machine = %s\n",
requestInfo[index].LoginSourceMachine );
            printf ( "AppName = %s\n", requestInfo[index].AppName );
            printf ( "DbName = %s\n", requestInfo[index].DbName );
            printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
            printf ( "RequestString = %s\n", requestInfo[index].RequestString );
            printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
            printf ( "State = %d\n", requestInfo[index].State );
            printf ( "\n\n-------------------------------------\n\n",
requestInfo[index].State );

            sts = EssKillRequestEx (hCtx, &requestInfo[index] );

            index++;
        }

        EssFree ( hInst, requestInfo );
```

```
    }

    userId = " native://nvid=f0ed2a6d7fb07688:5a342200:1265973105c:-7f46?USER ";
    bIsIdentity = ESS_TRUE;
    sts = EssListRequestsEx(hCtx, userId, bIsIdentity, "Sample", "Sample1",
&numRequest, &requestInfo);
    printf("\nEssListRequestsEx sts: %ld\n", sts);
    printf ( "Total requests on the server: %d\n", numRequest );
    if ( !sts && requestInfo )
    {
        while ( index < numRequest )
        {
            printf ( "login ID = %ul\n", requestInfo[index].LoginId );
            printf ( "user name = %s\n", requestInfo[index].UserName );
            printf ( "login machine = %s\n",
requestInfo[index].LoginSourceMachine );
            printf ( "AppName = %s\n", requestInfo[index].AppName );
            printf ( "DbName = %s\n", requestInfo[index].DbName );
            printf ( "DbRequestCode = %u\n", requestInfo[index].DbRequestCode );
            printf ( "RequestString = %s\n", requestInfo[index].RequestString );
            printf ( "TimeStarted = %ul\n", requestInfo[index].TimeStarted );
            printf ( "State = %d\n", requestInfo[index].State );
            printf ( "\n\n-------------------------------------\n\n",
requestInfo[index].State );

            sts = EssKillRequestEx (hCtx, &requestInfo[index] );

            index++;
        }

        EssFree ( hInst, requestInfo );
    }
}
```

関連トピック

- EssKillRequestEx

EssListSpoolFiles

データベースのすべてのトリガー・ログ・ファイルをリストします。

構文

```
ESS_FUNC_M EssListSpoolFiles (
hCtx, AppName, DbName, pCount, ppFileList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pCount	ESS_PUSHORT_T	スプール・ファイル名のカウントを受け取る変数のアドレス。
ppFileList	ESS_PPOBJINFO_T	割り当てられたスプール・ファイル名オブジェクトの配列を受け取るポインタのアドレス。

備考

この関数に対して割り当てられたメモリーは、EssFree を呼び出して解放する必要があります。

戻り値

正常終了の場合、データベース内のスプール・ファイルのカウントが pCount に、スプール・ファイル名の配列が ppFileList に戻されます。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssDisplayTriggers
- EssGetSpoolFile
- EssDeleteAllSplFiles
- EssDeleteSplFile
- EssMdxTrig

EssListSSMigrFailedGroups

Shared Services へ正常に移行しなかったグループを表示します。

構文

```
ESS_FUNC_M EssListSSMigrFailedGroups (
hCtx
,
count
,
pNativeUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
count	ESS_PUSHORT_T	移行できなかったグループのカウントを受け取る変数のアドレス。

パラメータ	データ型	説明
pNativeUserList	ESS_PPUSERNAME_T	移行できなかったグループの割り当てられた配列を受け取るポインタのアドレス。

備考

Essbase が Shared Services に移行していない場合、この関数はサポートされず、エラーが戻されます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_SS_ListSSMigrFailedGroups(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PUSERNAME_T  pNativeUserList = NULL;
 ESS_USHORT_T    Count = 0,
         index;

 sts = EssListSSMigrFailedGroups(hCtx, &Count, &pNativeUserList);

 if (!sts)
 {
  if (Count && pNativeUserList)
  {
   printf ("\n------- Group List -------\n\n");

   for (index = 0; index < Count; index++)
   {
    if (pNativeUserList[index])
     printf ("%s\n", pNativeUserList[index]);
   }

   EssFree(hInst, pNativeUserList);
  }
  else
   printf("\nGroup list is empty\n\n");
 }
 else
  printf("Failed to get Shared Services migration failed Groups list.\n");

 return (sts);
}
```

拡張された付録 B も参照してください

関連トピック

- EssListSSMigrFailedUsers

EssListSSMigrFailedUsers

Shared Services へ正常に移行しなかったユーザーを表示します。

構文

```
ESS_FUNC_M EssListSSMigrFailedUsers (
hCtx
,
count
,
pNativeUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
count	ESS_PUSHORT_T	移行できなかったユーザーのカウントを受け取る変数のアドレス。
pNativeUserList	ESS_PPUSERNAME_T	移行できなかったユーザーの割り当てられた配列を受け取るポインタのアドレス。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_SS_ListSSMigrFailedUsers(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PUSERNAME_T  pNativeUserList = NULL;
 ESS_USHORT_T    Count = 0,
         index;

 sts = EssListSSMigrFailedUsers(hCtx, &Count, &pNativeUserList);

 if (!sts)
 {
  if (Count && pNativeUserList)
  {
   printf ("\n------- User List -------\n\n");

   for (index = 0; index < Count; index++)
   {
    if (pNativeUserList[index])
     printf ("%s\n", pNativeUserList[index]);
   }

   EssFree(hInst, pNativeUserList);
  }
  else
   printf("\nUser list is empty\n\n");
 }
```

```
 else
  printf("Failed to get Shared Services migration failed Users list.\n");

 return (sts);
}
```

拡張された付録 B も参照してください

関連トピック

- EssListSSMigrFailedGroups

EssListTransactions

クライアント・バッファまたはカンマで区切られたファイルにトランザクション・メッセージが戻されます。カンマで区切られたファイルは、リレーショナル・データベースにエクスポートして、サードパーティ・ツールで処理できます。

構文

```
ESS_FUNC_M EssListTransactions(hCtx, TimeSrc, InpTime,
       ListOption, FileName, pCount, ppResults);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	この API を呼び出す前に、ログイン・コンテキストをアクティブに設定する必要があります。
TimeSrc	ESS_USHORT_T	トランザクションの表示開始時刻をどこで入手するのかを指定するオプション。
InpTime	ESS_TIME32_T	TimeSrc が ESS_TRLOG_TIMESPECIFIED の場合は、時刻を入力します。時刻は 1970 年 1 月 1 日以降に経過した秒数を ULONG 値で示します。
ListOption	ESS_USHORT_T	出力先を指定するオプション。110 ページの「リスト・オプション定数(C)」を参照してください
FileName	ESS_STR_T	ListOption がいずれかの LIST_TRANSACTIONS_*オプションの場合、サーバー・マシン上のこのファイルにコンテンツが書き込まれます: ● フル・パスを入力できます。 ● デフォルト: $ARBORPATH/app
pCount	ESS_PULONG_T	戻されたエントリの数
ppResults	ESS_PPTRANSACTION_ENTRY_T	ListOption が ESS_LIST_TRANSACTIONS_TOCLIENT の場合に戻されるエントリ

戻り値

- 0 - 正常終了の場合

- pCount には戻されるエントリ数が含まれています
- ppResults には、ListOptions が ESS_LIST_TRANSACTIONS_TOCLIENT の場合に戻されるエントリが含まれています

- エラー番号 - 失敗した場合

アクセス

リスト・トランザクションを呼び出す前に、アクティブな設定を使用してデータベースをアクティブにしておく必要があります。

呼出し元にはデータベースへの Essbase 管理者アクセス権限が必要です。

例

```
    void ListAndReplayTransactions()
{
    ESS_FUNC_M                sts = ESS_STS_NOERR;
    ESS_USHORT_T                TimeSrc;
    ESS_TIME32_T                timestamp = 0;
    ESS_USHORT_T                listOption;
    ESS_STR_T                    FileName = ESS_NULL;
    ESS_ULONG_T                Count = 0;
    ESS_PTRANSACTION_ENTRY_T    pResults;
    ESS_CHAR_T                listTime[ESS_TIMESIZE];
    ESS_TRANSACTION_REPLAY_INP_T    ReplayInp;
    ESS_PSEQID_T                pSeqIds = ESS_NULL;
    ESS_OBJDEF_T                Data;
    ESS_STR_T                    Script;
    ESS_SHORT_T                isAbortOnError;
    ESS_PMBRERR_T            pMbrErr = NULL;
    ESS_PROCSTATE_T        pState;

    /* Load data from server */
    Data.hCtx    = hCtx;
    Data.AppName  = AppName;
    Data.DbName   = DbName;
    Data.ObjType  = ESS_OBJTYPE_TEXT;
    Data.FileName  = "Calcdat";
    isAbortOnError = ESS_TRUE;
    sts = EssImport (hCtx, ESS_NULL, &Data,
                              &pMbrErr, NULL, isAbortOnError);
    printf("EssImport sts: %ld\r\n",sts);

    /* List and replay with a specified time */
    TimeSrc = 1;
    strcpy(listTime, "09/18/2007:00:00:00");
                          /* mm/dd/yyyy:hh:mm:ss */
    timestamp = adtGenericGetTime(listTime);
    listOption = ESS_LIST_TRANSACTIONS_TOCLIENT;
    sts =
    EssListTransactions(hCtx, TimeSrc,
                                                timestamp, listOption,
                                                FileName, &Count, &pResults);


        /* This function converts listTime to the number of
```

```
            seconds since January 1, 1970. */

    printf("EssListTransactions sts: %ld\r\n",sts);
    if (Count && pResults)
        PrintTransactionLog(Count, pResults);

    memset(&ReplayInp, 0, sizeof
                               (ESS_TRANSACTION_REPLAY_INP_T));
    ReplayInp.InpType = ESS_REPLAY_BASED_GIVENTIME;
    ReplayInp.value.InpTime = timestamp;
    sts = EssReplayTransactions (hCtx, AppName, DbName,
                                                    ReplayInp, pSeqIds);
    printf("EssReplayTransactions sts: %ld\r\n",sts);
    printf("\n\n");

    /* Run a calc*/
    Script = "CALC ALL;";
    sts = EssCalc(hCtx, ESS_TRUE, Script);
    printf("EssCalc sts: %ld\r\n",sts);
    if (!sts)
    {
        sts = EssGetProcessState (hCtx, &pState);
        while (!sts && (pState.State != ESS_STATE_DONE))
            sts = EssGetProcessState (hCtx, &pState);
    }

    /* List and replay with last replay time */
    TimeSrc = 2;
    timestamp = 0;
    sts =
    EssListTransactions(hCtx, TimeSrc,
                                          timestamp, listOption,
                                          FileName, &Count, &pResults);


        /* This function converts listTime to the number of
            seconds since January 1, 1970. */
    printf("EssListTransactions sts: %ld\r\n",sts);
    if (Count && pResults)
        PrintTransactionLog(Count, pResults);
    memset(&ReplayInp, 0, sizeof
                            (ESS_TRANSACTION_REPLAY_INP_T));
    ReplayInp.InpType = ESS_REPLAY_BASED_LASTREPLAYTIME;
    sts = EssReplayTransactions (hCtx, AppName,
                        DbName, ReplayInp, pSeqIds);
    printf("EssReplayTransactions sts: %ld\r\n",sts);

    if(pSeqIds)
        EssFree(hInst, pSeqIds);
    if(pResults)
        EssFree(hInst, pResults);
    if(pMbrErr)
        EssFree(hInst, pMbrErr);
}
```

関連トピック

- 197 ページの「ESS_SEQID_T」
- 144 ページの「ESS_DISKVOLUME_REPLACE_T」
- 198 ページの「ESS_TRANSACTION_ENTRY_T」
- 199 ページの「ESS_TRANSACTION_REPLAY_INP_T」
- 200 ページの「ESS_TRANSACTION_REQSPECIFIC_T」
- EssReplayTransactions

EssListUsers

特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。

構文

```
ESS_FUNC_M EssListUsers (
hCtx, AppName, DbName, pCount, ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は、すべてのユーザーがリストされます。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション内のすべてのデータベースのユーザーがリストされます。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス。
ppUserList	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	割り当てられたユーザー情報構造体の配列を受け取るポインタのアドレス。戻されるユーザー情報構造体の AppName および DbName には NULL 値が含まれます。

備考

- AppName および DbName の両方が NULL でない場合、指定したアプリケーションとデータベースへのアクセス権を持つユーザーのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つユーザーのみがリストされます。AppName が NULL の場合、サーバー上のすべてのユーザーがリストされます。
- 戻された ESS_USERINFO_T 構造体の「AppName」および「DbName」フィールドには、NULL 値が含まれます。
- ppUserList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー数のカウントが pCount に、指定したアプリケーションおよびデータベースへのアクセス権を持つユーザーのリストが ppUserList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_STS_T
ESS_ListUsers (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_STS_T    sts;
 ESS_USHORT_T   Count;
 ESS_PUSERINFO_T  Users = NULL;
 ESS_USHORT_T   ind;

 sts = EssListUsers (hCtx, NULL, NULL, &Count,
   &Users);
 if (!sts)
 {
  if (Count && Users)
  {
printf ("\r\n-------User List-------\r\n\r\n");
    for (ind = 0; ind < Count; ind++)
    {
printf ("Name->%s Application->%s database->%s\r\n",
    Users[ind].Name, Users[ind].AppName,
    Users[ind].DbName);
    }
    EssFree (hInst, Users);
  }
  else
printf ("\r\nUsers list is empty\r\n\r\n");
 }
 return (sts);
}
```

関連トピック

- EssListUsersInfoEx
- EssGetUser
- EssListConnections
- EssListGroups
- EssListLocks

EssListUsersEx

特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。

構文

```
     ESS_FUNC_M EssListUsersEx (
     hCtx, AppName, DbName, SecurityProvider, pCount,  ppUserList
```

```
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。NULL の場合は、すべてのユーザーがリストされます。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション内のすべてのデータベースのユーザーがリストされます。
SecurityProvider	ESS_STR_T	外部認証メカニズム名。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス。
ppUserList	207 ページの「ESS_USERINFOEX_T」	割り当てられたユーザー情報構造体の配列を受け取るポインタのアドレス。戻されるユーザー情報構造体の AppName および DbName には NULL 値が含まれます。

備考

- AppName および DbName の両方が NULL でない場合、指定したアプリケーションとデータベースへのアクセス権を持つユーザーのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つユーザーのみがリストされます。AppName が NULL の場合、サーバー上のすべてのユーザーがリストされます。
- 戻される ESS_USERINFO_T 構造体の AppName および DbName フィールドには、NULL 値が含まれます。
- ppUserList に対して割り当てられたメモリーは、`EssFree` を使用して解放する必要があります。

戻り値

正常終了の場合は、ユーザー数のカウントが pCount に、指定したアプリケーションおよびデータベースへのアクセス権を持つユーザーのリストが ppUserList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_STS_T
ESS_ListUsers (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_STS_T     sts;
 ESS_USHORT_T   Count;
 ESS_PUSERINFO_T  Users = NULL;
 ESS_USHORT_T   ind;

 sts = EssListUsersEx (hCtx, NULL, NULL, &Count,
    &Users);
```

```
 if (!sts)
 {
  if (Count && Users)
  {
printf ("\r\n-------User List-------\r\n\r\n");
    for (ind = 0; ind < Count; ind++)
    {
printf ("Name->%s Application->%s database->%s\r\n",
    Users[ind].Name, Users[ind].AppName,
    Users[ind].DbName);
    }
    EssFree (hInst, Users);
  }
  else
printf ("\r\nUsers list is empty\r\n\r\n");
 }
 return (sts);
}
```

関連トピック

- EssGetUser
- EssListConnections
- EssListGroups
- EssListLocks
- EssCreateExtUser
- EssGetUserEx
- EssSetUserEx
- 207 ページの「ESS_USERINFOEX_T」

EssListUsersInfoEx

特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。EssListUsers に似ていますが、ユーザー・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

構文

```
ESS_FUNC_M EssListUsersInfoEx (
hCtx
,
AppName
,
DbName
,
pCount
,
ppUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
AppName	ESS_STR_T	アプリケーション名(入力)。NULL の場合は、すべてのユーザーがリストされます。
DbName	ESS_STR_T	データベース名(入力)。NULL の場合は、アプリケーション内のすべてのデータベースのユーザーがリストされます。
pCount	ESS_PUSHORT_T	ユーザーのカウントを受け取る変数のアドレス(出力)。
ppUserList	ESS_PPUSERINFOID_T	割り当てられたユーザー情報構造体の配列を受け取るポインタのアドレス(出力)。ユーザー・リスト構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

戻り値

正常終了の場合は、ユーザー数のカウントが pCount に、指定したアプリケーションおよびデータベースへのアクセス権を持つユーザーのリストが ppUserList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
void DisplayUserInfoID2(ESS_USERINFOID_T userInfo)
{
 ESS_STS_T sts  = ESS_STS_NOERR;
 ESS_BOOL_T    isDefined = ESS_TRUE;

 printf("\tUser Name: %s\n", userInfo.Name);
 printf("\tProvider Name: %s\n", userInfo.ProviderName);
 printf("\tConnparam: %s\n", userInfo.connparam);
 printf("\tDescription: %s\n", userInfo.Description);
 printf("\tEMail Identification: %s\n", userInfo.EMailID);

 if (userInfo.LockedOut)
  printf("\tLocked out: Yes\n");
 else
  printf("\tLocked out: No\n");

 if (userInfo.PwdChgNow)
  printf("\tChange the password now: Yes\n");
 else
  printf("\tChange the password now: No\n");

 printf(  "\tConnected Application: %s\n", userInfo.AppName);
 printf(  "\tConnected Database: %s\n",  userInfo.DbName);

 if (userInfo.Login)
  printf("\tLogged in: Yes\n");
 else
  printf("\tLogged in: No\n");
```

```
switch(userInfo.Access)
{
 case ESS_ACCESS_ADMIN:
  printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPALL:
  printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBALL:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
  break;
 case ESS_ACCESS_APPMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBCREATE:
  printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userInfo.Access);
  break;
 case ESS_ACCESS_DBMANAGE:
  printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userInfo.Access);
  break;
 case ESS_ACCESS_CALC:
  printf("\tAccess: %d - ESS_ACCESS_CALC\n", userInfo.Access);
  break;
 case ESS_ACCESS_WRITE:
  printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userInfo.Access);
  break;
 case ESS_ACCESS_READ:
  printf("\tAccess: %d - ESS_ACCESS_READ\n", userInfo.Access);
  break;
 case ESS_PRIV_USERCREATE:
  printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userInfo.Access);
  break;
 case ESS_PRIV_APPCREATE:
  printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userInfo.Access);
  break;
 case ESS_PRIV_APPMANAGE:
  printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userInfo.Access);
  break;
 case ESS_PRIV_APPLOAD:
  printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userInfo.Access);
  break;
 case ESS_PRIV_DBCREATE:
  printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userInfo.Access);
  break;
 case ESS_PRIV_DBMANAGE:
  printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userInfo.Access);
  break;
 case ESS_PRIV_DBLOAD:
  printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userInfo.Access);
  break;
 case ESS_PRIV_CALC:
  printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userInfo.Access);
  break;
 case ESS_PRIV_WRITE:
```

```
   printf("\tAccess: %d - ESS_PRIV_WRITE\n", userInfo.Access);
   break;
  case ESS_PRIV_READ:
   printf("\tAccess: %d - ESS_PRIV_READ\n", userInfo.Access);
   break;
  case ESS_PRIV_NONE:
   printf("\tAccess: %d - ESS_PRIV_NONE\n", userInfo.Access);
   break;
  default:
   printf("\tAccess: Unknown\n");
}

switch(userInfo.MaxAccess)
{
  case ESS_ACCESS_ADMIN:
   printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPALL:
   printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBALL:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_APPMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBCREATE:
   printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_DBMANAGE:
   printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_CALC:
   printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_WRITE:
   printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userInfo.MaxAccess);
   break;
  case ESS_ACCESS_READ:
   printf("\tMax Access: %d - ESS_ACCESS_READ\n", userInfo.MaxAccess);
   break;
  case ESS_PRIV_USERCREATE:
   printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userInfo.MaxAccess);
   break;
  case ESS_PRIV_APPCREATE:
   printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userInfo.MaxAccess);
   break;
  case ESS_PRIV_APPMANAGE:
   printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userInfo.MaxAccess);
   break;
  case ESS_PRIV_APPLOAD:
   printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userInfo.MaxAccess);
   break;
  case ESS_PRIV_DBCREATE:
```

```
    printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_DBMANAGE:
    printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", userInfo.MaxAccess);
    break;
   case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", userInfo.MaxAccess);
    break;
   default:
    printf("\tMax Access: Unknown\n");
  }

  printf("\tPassword Expiration in Dates: %d\n",userInfo.Expiration);
  //EssSdCTime(NULL, userInfo.LastLogin, sizeof(time_string), time_string);
  //printf("\tLast Successful Login:      %s\n", time_string);
  printf("\tFailed Login Attempts Since Then: %d\n", userInfo.FailCount);
  printf("\tLogin ID: %ld\n", userInfo.LoginId);
  printf( "\n");
}

ESS_STS_T ESS_ListUsersInfo (ESS_HCTX_T  hCtx, ESS_HINST_T hInst)

{
  ESS_STS_T sts  = ESS_STS_NOERR;
  ESS_USHORT_T count, i;
  ESS_PUSERINFOID_T pUserList;


  sts = EssListUsersInfoEx(hCtx, AppName, "", &count, &pUserList);
  printf("EssListUsersInfoEx sts: %ld\n", sts);
  if(!sts)
  {
      printf("\tNumber of users: %d\n\n", count);
      for(i = 0; i < count; i++)
      {
          DisplayUserInfoID2(pUserList[i]);
      }
  }

  return (sts);
}
```

関連トピック

- EssCreateExtUser
- EssListGroupsInfoEx

EssListVariables

入力基準に従って、サーバー、アプリケーション、データベース・レベルの代替変数をリストします。

構文

```
ESS_FUNC_M EssListVariables (
hCtx, pCriteria, pNumVars, ppVarList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pCriteria	209 ページの「ESS_VARIABLE_T」	リストされた代替変数の説明を含む構造体を指すポインタ。 ● メンバー VarName および VarValue は無視されます。 ● Server メンバーは指定されていますが、AppName および DbName が空の場合は、関数によってサーバー・レベルの代替変数のみがリストされます。 ● Server および AppName メンバーは指定されていますが、DbName が空の場合は、指定したサーバーとアプリケーション・レベルの両方のすべての変数がリストされます。 ● Server、AppName および DbName という 3 つのメンバーすべてが指定されている場合は、指定した 3 つのすべてのレベルの変数がリストされます。 ● フィールドが空の場合は、そのフィールドは「無視」するよう処理されます。
pNumVars	ESS_PULONG_T	ppVarList パラメータに戻される変数の数を示す、符号なし long 値を指すポインタ。
ppVarList	209 ページの「ESS_VARIABLE_T」	代替変数構造体の配列へのポインタ。呼出し元は EssFree を呼び出して、この配列を解放する必要があります。

戻り値

成功の場合、ゼロが戻されます。

例

```
    /*
** ESS_ListVariables() lists the substitution variables using
** the API EssListVariables.
*/
ESS_FUNC_M
ESS_ListVariables (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_PVARIABLE_T  pVariables;
```

```
ESS_ULONG_T    ulCount, i;
ESS_VARIABLE_T  Variable;
printf("\n ****************************************");
printf("\n **** An example of using EssListVariables");
printf("\n ****************************************");
/*************************************************************/
/* List Variables at the level of the Server/App/Db       */
/* Variables under that specific server will be listed     */
/* Variables under that specific server/ App will be listed   */
/* Variables under that specific server/ App /DB will be listed */
/*************************************************************/
strcpy(Variable.VarName, "");  // ignored by EssListVariables
strcpy(Variable.Server, "local");
strcpy(Variable.AppName, "Sample");
strcpy(Variable.DbName, "Basic");
sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
if (sts == ESS_STS_NOERR)
{
 printf("\n--- Number of Substitution Variables at the Server, App and Db
  level is: %ld\n", ulCount);
 for (i = 0; i < ulCount; i++)
 {
  printf("Variable name   : %s\n",  pVariables[i].VarName);
  printf("Server name    : %s\n",  pVariables[i].Server);
  printf("Application name : %s\n",  pVariables[i].AppName);
  printf("Database name   : %s\n",  pVariables[i].DbName);
  printf("Variable value  : %s\n\n", pVariables[i].VarValue);
 }
}
/*************************************************************/
/* Variables under that specific Server will be listed     */
/* Variables under that specific Server/App will be listed   */
/*************************************************************/
if (sts == ESS_STS_NOERR)
{

 strcpy(Variable.VarName, "");  // ignored by EssListVariables
 strcpy(Variable.Server, "local");
 strcpy(Variable.AppName, "Sample");
 strcpy(Variable.DbName, "");
 sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
 if (sts == ESS_STS_NOERR)
 {
  printf("\n--- Number of Substitution Variables at the Server and App
   level is: %ld\n", ulCount);
  for (i = 0; i < ulCount; i++)
  {
   printf("Variable name   : %s\n",  pVariables[i].VarName);
   printf("Server name    : %s\n",  pVariables[i].Server);
   printf("Application name : %s\n",  pVariables[i].AppName);
   printf("Database name   : %s\n",  pVariables[i].DbName);
   printf("Variable value  : %s\n\n", pVariables[i].VarValue);
  }
 }
}
/*************************************************************/
/* List Variables at the level of the Server         */
```

```
 /*************************************************************/
 if (sts == ESS_STS_NOERR)
 {
  strcpy(Variable.VarName, "");  // ignored by EssListVariables
  strcpy(Variable.Server, "local");
  strcpy(Variable.AppName, "");
  strcpy(Variable.DbName, "");
  if (sts == ESS_STS_NOERR)
   sts = EssListVariables(hCtx, &Variable, &ulCount, &pVariables);
  if (sts == ESS_STS_NOERR)
  {
   printf("\n--- Number of Substitution Variables at the Server level is:
    %ld\n", ulCount);
   for (i = 0; i < ulCount; i++)
   {
    printf("Variable name   : %s\n",  pVariables[i].VarName);
    printf("Server name     : %s\n",  pVariables[i].Server);
    printf("Application name : %s\n",  pVariables[i].AppName);
    printf("Database name   : %s\n",  pVariables[i].DbName);
    printf("Variable value  : %s\n\n", pVariables[i].VarValue);
   }
  }
 }

 if (sts == ESS_STS_NOERR)
  printf("\n --> No Errors in EssListVariables\n\n\n");
 else
  printf("\n --> Error in EssListVariables number: %d\n\n\n", sts);

 return (sts);
} /* end ESS_ListVariables */
```

出力

```
    ***************************************
**** An example of using EssListVariables
***************************************
--- Number of Substitution Variables at the Server, App and Db level is: 3
Variable name   : QuarterName
Server name     : local
Application name : Sample
Database name   : Basic
Variable value  : Qtr2
Variable name   : MarketName
Server name     : local
Application name : Sample
Database name   :
Variable value  : East
Variable name   : MarketName
Server name     : local
Application name :
Database name   :
Variable value  : Market

--- Number of Substitution Variables at the Server and App level is: 2
```

```
Variable name   : MarketName
Server name     : local
Application name : Sample
Database name   :
Variable value  : East
Variable name   : MarketName
Server name     : local
Application name :
Database name   :
Variable value  : Market

--- Number of Substitution Variables at the Server level is: 1
Variable name   : MarketName
Server name     : local
Application name :
Database name   :
Variable value  : Market

 --> No Errors in EssListVariables
```

関連トピック

- 209 ページの「ESS_VARIABLE_T」
- EssCreateVariable
- EssDeleteVariable
- EssGetVariable

EssLoadAlias

構造化テキスト・ファイルからアクティブなデータベースの別名テーブルを作成し、永続的にロードします。

構文

```
ESS_FUNC_M EssLoadAlias (
hCtx, AliasName, FileName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AliasName	ESS_STR_T	ロードする別名テーブル名。
FileName	ESS_STR_T	サーバー上の構造化別名ファイルのフル・パス名。

備考

- この関数は AliasName がすでに存在する場合は正常に終了しません。既存のテーブルと同じ名前で別名テーブルをロードする前に、既存の別名テーブルを削除する必要があります。
- 別名テーブル・ファイルのフォーマットは、『Oracle Essbase データベース管理者ガイド』に記載されています。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_LoadAlias (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   TableName;
 ESS_STR_T   FileName;
 TableName = "NewAlias";
 FileName  = "NEW.ALT";
 sts = EssLoadAlias (hCtx, TableName, FileName);

 return (sts);
}
```

関連トピック

- EssListAliases
- EssSetActive

EssLoadApplication

サーバー上のアプリケーションを開始します。

構文

```
ESS_FUNC_M EssLoadApplication (
hCtx, AppName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	ロードするアプリケーションの名前。

備考

アプリケーションをロードするには、接続されているユーザーがアプリケーションに対してロード権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定されたアプリケーションに対してアプリケーションのロード/アンロード権限(ESS_PRIV_APPLOAD)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_LoadApp (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;
 AppName = "Sample";
 sts = EssLoadApplication (hCtx, AppName);
 return (sts);
}
```

関連トピック

- EssLoadDatabase
- EssUnloadApplication

EssLoadBufferInit

一時的なデータ・ロード・バッファを作成し、集約ストレージ・データベースへのデータ・ロード中にタプルの一時記憶領域を提供します。集約ストレージ・データベースにのみ適用されます。

構文

```
     ESS_FUNC_M EssLoadBufferInit (
     hCtx, AppName, DbName, ulBufferId, ulDuplicateAggregationMethod,
ulOptionFlags, ulSize
     );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	ロード・バッファを作成するアプリケーション名。
DbName	ESS_STR_T	ロード・バッファを作成するデータベース名。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID 番号(1 から 999,999 までの数値)。ID がすでに使用されている場合、操作は失敗します。

パラメータ	データ型	説明
ulDuplicateAggregationMethod	ESS_ULONG_T	バッファ内の同じセルに対する複数の値を組み合せる定数を次に示します: ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD: バッファが同じセルに対する複数の値を含んでいる場合に値を追加します。 #define ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD 0 ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ASSUME_EQUAL: 同じセルに対する複数の値が同一であることを確認します; 同一である場合、重複を無視します。同じセルに対する値が異なる場合は、エラー・メッセージを表示してデータ・ロードを停止します。 #define ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ASSUME_EQUAL 1 ● ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_USE_LAST: ロード・バッファに最後にロードされたセルの値を使用することで、重複するセルを組み合せます。このオプションは、比較的小規模な、最大 10,000 までのセル・データのロード用です。 #define ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_USE_LAST 2 use_last オプションでデータ・ロード・バッファを使用する場合、データ・ロードは重複した値がなくても著しく遅くなります。 **注意** use_last メソッドは、パフォーマンスに重大な影響を及ぼすため、大量のデータ・ロード向きではありません。データ・ロードが 100 万セルを超える場合、数値データを(型付きメジャー・データとは)別のデータ・ロード・プロセスに分けることを検討してください。この個別データ・ロードでは、かわりに add メソッドを使用できます。

パラメータ	データ型	説明
ulOptionFlags	ESS_ULONG_T	次のロード・バッファ・オプションのうち、1 つ以上: ● ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES: 入力データ・ストリーム内の#MISSING 値を無視します。 #define ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES 0x00000001 ● ESS_ASO_DATA_LOAD_BUFFER_IGNORE_ZERO_VALUES: 入力データ・ストリーム内のゼロを無視します。 #define ESS_ASO_DATA_LOAD_BUFFER_IGNORE_ZERO_VALUES 0x00000002 ● ESS_ASO_DATA_LOAD_BUFFER_WAIT_FOR_RESOURCES: ロード・バッファ操作を実行するには、リソースが使用可能になるまで、essbase.cfg の ASOLOADBUFFERWAIT 構成設定で指定された期間待機するよう、Essbase に指示します。 #define ESS_ASO_DATA_LOAD_BUFFER_WAIT_FOR_RESOURCES 0x00000004 ビット単位 OR(|)を使用して、ulOptions を複数指定します; 次に例を示します: ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES | ESS_ASO_DATA_LOAD_BUFFER_IGNORE_ZERO_VALUES
ulSize	ESS_ULONG_T	このロード・バッファが使用できるロード・バッファ・リソースのパーセンテージ。可能な値: 0-100。 値が 0 の場合、Essbase では、自ら決定したデフォルトのロード・バッファ・サイズが使用されます。 全ロード・バッファの合計サイズが 100 を超えると、操作は失敗します。

備考

複数のバッファが単一の集約ストレージ・データベースに存在できます;ただし、指定したロード・バッファを一度に使用できるのは、1 つのデータ・ロードのみです。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

例

```
    void TestBeginDataloadASO(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
```

```
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_BOOL_T    Store;
    ESS_BOOL_T    Unlock;
    ESS_BOOL_T    abortOnError;
    ESS_STR_T    loadString;
    ESS_OBJDEF_T rulesFile;
    ESS_PMBRERR_T pMbrErr;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T   ulDuplicateAggregationMethod;
    ESS_ULONG_T   ulOptionsFlags;
    ESS_ULONG_T   ulSize;
    ESS_ULONG_T   ulBufferCnt;
    ESS_ULONG_T   ulCommitType ;
    ESS_ULONG_T   ulActionType;
    ESS_ULONG_T   ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];

    /* EssLoadBufferInit */
    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
    ulBufferId = 201;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* EssBeginDataloadASO, EssSendString, EssEndDataload */
    Store  = ESS_TRUE;
    Unlock = ESS_FALSE;
    abortOnError = ESS_FALSE;
    loadString = "Mar Sale \"Curr Year\" \"Original Price\" \"017589\" \"13668\" Cash
\"No Promotion\" \"1 to 13 Years\" \"Under 20,000\" \"Digital Cameras\" 111";

    sts = EssBeginDataloadASO (hCtx, Store, Unlock, abortOnError, ESS_NULL,
ulBufferId);
    printf("EssBeginDataloadASO sts: %ld\n",sts);
    sts = EssSendString(hCtx, loadString);
    printf("EssSendString sts: %ld\n",sts);
    sts = EssEndDataload(hCtx, &pMbrErr);
    printf("EssEndDataload sts: %ld\n",sts);

    /* EssLoadBufferTerm */
    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\Commit data to main slice and destroy buffer:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);

}
```

関連トピック

- EssBeginDataloadASO
- EssSendString
- EssEndDataload
- EssLoadBufferTerm
- EssImportASO
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssListExistingLoadBuffers
- EssMergeDatabaseData

EssLoadBufferTerm

EssLoadBufferInit によって割り当てられた一時的なデータロード・メモリー・バッファを破棄し、データを集約ストレージ・データベースにロードします。オプションで、データを先にコミットすることも可能です。

集約ストレージ・データベースにのみ適用されます。

構文

```
     ESS_FUNC_M EssLoadBufferTerm (
     hCtx, AppName, DbName, ulBufferCnt, *ulBufferIdAry, ulCommitType,
ulActionType, ulOptions
     );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベースの名前。
ulBufferCnt	ESS_ULONG_T	リスト内のバッファ数。
*ulBufferIdAry	ESS_ULONG_T	この操作で影響を受けるバッファ ID の配列。

パラメータ	データ型	説明
ulCommitType	ESS_ULONG_T	次のいずれかの定数を使用して、バッファに格納されている値を、すでにデータベースに格納されている値と組み合せます: ● ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA: データベース内の既存のセル値を、ロード・バッファからの新しい値で置き換えます。データベース内のセルで、バッファに対応する値がないものは更新されません。 #define ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA 0 ● ESS_ASO_DATA_LOAD_BUFFER_ADD_DATA: 既存の値に新しい値を追加します。 #define ESS_ASO_DATA_LOAD_BUFFER_ADD_DATA 1 ● ESS_ASO_DATA_LOAD_BUFFER_SUBTRACT_DATA: 既存の値から新しい値を削除します。 #define ESS_ASO_DATA_LOAD_BUFFER_SUBTRACT_DATA 2 ● ESS_ASO_DATA_LOAD_BUFFER_OVERRIDE_ALL_DATA: データベース内の既存のデータ・セル(データベース内のセルで、ロード・バッファに対応する値がないものも含む)をすべて破棄し、1 回の操作でロード・バッファのコンテンツをロードします。 #define ESS_ASO_DATA_LOAD_BUFFER_OVERRIDE_ALL_DATA 3 すべてのデータを上書きするオプションを使用する際は、ulOptions 設定は無視されます。Essbase では、現在バッファに格納されているデータは、データベースのメイン・スライスに書き込まれます。 ● ESS_ASO_DATA_LOAD_BUFFER_OVERRIDE_INCREMENTAL_DATA: 現在任意の増分スライスに格納されているすべてのデータ・セルを破棄し、ロード・バッファのコンテンツをロードします。 #define ESS_ASO_DATA_LOAD_BUFFER_OVERRIDE_INCREMENTAL_DATA 4 増分データを上書きするオプションを使用する際に、ulOptions 設定がメイン・スライスである場合、Essbase では ulOptions 設定が無視され、現在バッファに格納されているデータは、データベースの新規スライスに書き込まれます。 複数のバッファをコミットする場合、この ulCommitType 設定やバッファ自体の構成方法とは関係なく、異なるバッファからの値は常に追加操作を使用して結合されます。 **注:** ulActionType 設定が中止の場合、ulCommitType 設定は無視されます。

パラメータ	データ型	説明
ulActionType	ESS_ULONG_T	次の定数のいずれかになります: ● ESS_ASO_DATA_LOAD_BUFFER_COMMIT: データがロード・バッファからデータベースにロードされます; その後、バッファが破棄されます。 #define ESS_ASO_DATA_LOAD_BUFFER_COMMIT 1 ● ESS_ASO_DATA_LOAD_BUFFER_ABORT: ロード・バッファが破棄されます。バッファ内のすべてのデータが失われます。 #define ESS_ASO_DATA_LOAD_BUFFER_ABORT 2 中止オプションを使用すると、ulCommitType および ulOptions 設定が無視されます
ulOptions	ESS_ULONG_T	次の定数のいずれかになります: ● ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE: 現在バッファに格納されているデータが、データベースのメイン・スライスに書き込まれます。 #define ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE 0 メイン・スライスへの追加オプションを使用する際に、ulCommitType が増分データを上書きするように設定されている場合、Essbase では ulOptions 設定が無視され、現在バッファに格納されているデータが、データベースの新規スライスに書き込まれます。 ● ESS_ASO_DATA_LOAD_INCR_TO_NEW_SLICE: 現在バッファに格納されているデータが、データベースの新規スライスに書き込まれます。この操作により、データのロード速度が速くなります。 #define ESS_ASO_DATA_LOAD_INCR_TO_NEW_SLICE 1 ● ESS_ASO_DATA_LOAD_INCR_TO_NEW_SLICE_LIGHTWEIGHT: 現在バッファに格納されているデータが、軽量操作として、データベースの新規スライスに書き込まれます。このオプションは、最大 1,000 のセルまでの同時に発生する小規模なデータ・ロード(グリッド・クライアントのデータ更新操作など)用です。 #define ESS_ASO_DATA_LOAD_INCR_TO_NEW_SLICE_LIGHTWEIGHT 2 **注:** ulCommitType がすべてのデータを上書きするように設定されている場合、ulOptions 設定は無視されます。Essbase では、現在バッファに格納されているデータは、データベースのメイン・スライスに書き込まれます。ulActionType 設定が中止の場合、ulOptions 設定は無視されます

備考

この関数は指定されたロード・バッファ(通常は単一のロード・バッファ)を破棄します。指定されたアクション・タイプが"コミット"の場合は、バッファの破棄前に、バッファ内に現在保管されているデータがデータベースに適用されます。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

例

```
    void TestBeginDataloadASO(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_BOOL_T    Store;
    ESS_BOOL_T    Unlock;
    ESS_BOOL_T    abortOnError;
    ESS_STR_T    loadString;
    ESS_OBJDEF_T rulesFile;
    ESS_PMBRERR_T pMbrErr;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T   ulDuplicateAggregationMethod;
    ESS_ULONG_T   ulOptionsFlags;
    ESS_ULONG_T   ulSize;
    ESS_ULONG_T   ulBufferCnt;
    ESS_ULONG_T   ulCommitType ;
    ESS_ULONG_T   ulActionType;
    ESS_ULONG_T   ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];

    /* EssLoadBufferInit */
    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
    ulBufferId = 201;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* EssBeginDataloadASO, EssSendString, EssEndDataload */
    Store  = ESS_TRUE;
    Unlock = ESS_FALSE;
    abortOnError = ESS_FALSE;
    loadString = "Mar Sale \"Curr Year\" \"Original Price\" \"017589\" \"13668\" Cash
\"No Promotion\" \"1 to 13 Years\" \"Under 20,000\" \"Digital Cameras\" 111";

    sts = EssBeginDataloadASO (hCtx, Store, Unlock, abortOnError, ESS_NULL,
ulBufferId);
    printf("EssBeginDataloadASO sts: %ld\n",sts);
    sts = EssSendString(hCtx, loadString);
    printf("EssSendString sts: %ld\n",sts);
    sts = EssEndDataload(hCtx, &pMbrErr);
    printf("EssEndDataload sts: %ld\n",sts);

    /* EssLoadBufferTerm */
```

```
    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\Commit data to main slice and destroy buffer:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);

}
```

関連トピック

- EssLoadBufferInit
- EssBeginDataloadASO
- EssSendString
- EssEndDataload
- EssImportASO
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssListExistingLoadBuffers
- EssMergeDatabaseData

EssLoadDatabase

アプリケーション内のデータベースをサーバー上で開始します。

構文

```
ESS_FUNC_M EssLoadDatabase (
hCtx, AppName, DbName
);
```

パラメータ　データ型　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	ロードするデータベースの名前。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースのロード/アンロード権限(ESS_PRIV_APPLOAD)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_LoadDb (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;

 AppName = "Sample";
 DbName  = "Basic";
 sts = EssLoadDatabase(hCtx, AppName, DbName);

 return (sts);
}
```

関連トピック

- EssLoadApplication
- EssUnloadDatabase

EssLocateIBH

データベース内の無効なブロック・ヘッダーを検索します。検索プロセスの最後に、サーバー・ベースの IBH ログ・ファイルが作成されます。後でこのファイルを EssFixIBH で使用して、エラーを修正できます。

構文

```
ESS_FUNC_M EssLocateIBH (
hCtx
,
dbName
);
```

パラメータ　データ型　　説明

hCtx;　　ESS_HCTX_T API コンテキスト・ハンドル。

dbName;　　ESS_STR_T　データベースの名前。

関連トピック

- EssFixIBH
- EssGetIBH

EssLockObject

サーバーまたはクライアントのオブジェクト・システム上のオブジェクトをロックして、他のユーザーが更新できないようにします。

構文

```
ESS_FUNC_M EssLockObject (
hCtx, ObjType, AppName, DbName, ObjName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	ロックするオブジェクトの名前。

備考

- オブジェクトをロックするには、そのオブジェクトが存在している必要があり、かつ他のユーザーによってロックされていてはいけません。
- この関数はオブジェクトを取得しません。オブジェクトを取得するには、EssGetObject を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、オブジェクトが含まれる指定されたアプリケーションまたはデータベースに対してアプリケーション・デザイン権限またはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_LockObject (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     ObjName;
 ESS_OBJTYPE_T   ObjType;

 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Basic";
 ObjType = ESS_OBJTYPE_OUTLINE;

 sts = EssLockObject (hCtx, ObjType, AppName,
```

```
    DbName,ObjName);
  if(!sts)
   printf("The Object \"%s\" is locked\r\n",
       ObjName);
  return (sts);
}
```

関連トピック

- EssGetObject
- EssGetObjectInfo
- EssListObjects
- EssPutObject
- EssUnlockObject

EssLogin

ユーザーを Essbase サーバーにログインさせます。この関数は通常、EssInit の呼出しを正しく実行した後で、かつコンテキスト・ハンドルの引数が必要な他の API の呼出しを実行する前に呼び出す必要があります。

構文

```
ESS_FUNC_M EssLogin (
hInstance, Server, UserName, Password, pDbCount, ppDbList, phCtx
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。

Server	ESS_STR_T	ネットワーク・サーバー名の文字列。
		サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
		`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
		保護モード(SSL)の場合、URL の構文は次のとおりです
		`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
		たとえば、
		`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

パラメータ	データ型	説明
UserName	ESS_STR_T	ユーザー名の文字列。
Password	ESS_STR_T	パスワード文字列。
pDbCount	ESS_PUSHORT_T	アクセス可能なアプリケーションとデータベースのカウントを受け取る変数のアドレス。
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース名構造体の配列を受け取るポインタのアドレス。
phCtx	ESS_PHCTX_T	Essbase サーバー・コンテキスト・ハンドルへのポインタ。

備考

- Microsoft Windows でプログラミングしている場合は、EssLogin のかわりに EssAutoLogin 関数の使用を検討する必要があります。
- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- 同じユーザー名とサーバーに対してこの関数を複数回呼び出せます。指定したサーバーに対してログインするたびに、API から一意のコンテキスト・ハンドルが戻されます。

戻り値

成功の場合、Essbase サーバー・コンテキスト・ハンドルが phCtx に戻され、他の API 関数への後続の呼出しで引数として使用できます。また、指定したユーザーがアクセス可能なデータベース数が pCount に戻され、アクセス可能なアプリケーションおよびデータベースのリストが ppDbList に戻されます。

アクセス

この関数を呼び出す前に、EssInit を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

例

```
    ESS_FUNC_M
ESS_Login (ESS_HINST_T  hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_HCTX_T  hCtx;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;
 ESS_PAPPDB_T pAppsDbs = NULL;
 ESS_STR_T   SvrName;
 ESS_STR_T   User;
 ESS_STR_T   Password;

 SvrName  = "POPLAR";
 User    = "Joseph";
 Password = "Password";

 sts = EssLogin (hInst, SvrName, User, Password,
```

```
    &Items, &pAppsDbs, &hCtx);
  if (!sts)
  {
   for (ind = 0; ind < Items; ind++)
   {
     if ((pAppsDbs+ind) != NULL)
     {
       if ((pAppsDbs[ind].AppName != NULL) &&
         (pAppsDbs[ind].DbName  != NULL))
       {
printf ("%s\r\n", pAppsDbs[ind].AppName);
printf ("%s\r\n", pAppsDbs[ind].DbName);
       }
     }
   }
  }
  return(sts);
}
```

関連トピック

- EssAutoLogin
- EssLoginAs
- EssLoginEx
- EssLoginExAs
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssLoginAs

別のユーザーとして Essbase サーバーにログインします。これによって、管理者はユーザーごとの許可を設定してレポートのスケジュールを作成できます。

この関数は通常、EssInit の呼出しを正しく実行した後で、かつコンテキスト・ハンドルの引数が必要な他の API の呼出しを実行する前に呼び出す必要があります。

構文

```
ESS_FUNC_M EssLoginAs (
hInstance, Server, UserName, Password, UserNameAs, pDbCount, ppDbList, phCtx
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。

パラメータ	データ型	説明

Server	ESS_STR_T	ネットワーク・サーバー名の文字列。
		サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
		`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
		保護モード(SSL)の場合、URL の構文は次のとおりです
		`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
		たとえば、
		`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

UserName	ESS_STR_T	ユーザー名の文字列。
Password	ESS_STR_T	パスワード文字列。
UserNameAs	ESS_STR_T	代行するユーザーのユーザー名文字列。
pDbCount	ESS_PUSHORT_T	アクセス可能なアプリケーションとデータベースのカウントを受け取る変数のアドレス。
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース名構造体の配列を受け取るポインタのアドレス。
phCtx	ESS_PHCTX_T	Essbase サーバー・コンテキスト・ハンドルへのポインタ。

備考

- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。
- 同じユーザー名とサーバーに対してこの関数を複数回呼び出せます。指定したサーバーに対してログインするたびに、API から一意のコンテキスト・ハンドルが戻されます。

戻り値

成功の場合、Essbase サーバー・コンテキスト・ハンドルが phCtx に戻され、他の API 関数への後続の呼出しで引数として使用できます。また、指定したユーザーがアクセス可能なデータベース数が pCount に戻され、アクセス可能なアプリケーションおよびデータベースのリストが ppDbList に戻されます。

アクセス

別のユーザーとしてログインするには、管理者である必要があります。

この関数を呼び出す前に、EssInit を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

関連トピック

- EssAutoLogin
- EssLoginExAs
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssLoginEx

ユーザー名とパスワードではなく、ユーザー認証トークンを使用して、ユーザーを Essbase サーバーにログインさせます。この関数は通常、EssInit の呼出しを正しく実行した後で、かつコンテキスト・ハンドルの引数が必要な他の API の呼出しを実行する前に呼び出す必要があります。

構文

```
ESS_FUNC_M EssLoginEx (
hInstance, Server, Token, pDbCount, ppDbList, phCtx
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。

Server	ESS_STR_T	ネットワーク・サーバー名の文字列。
		サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
		`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
		保護モード(SSL)の場合、URL の構文は次のとおりです
		`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
		たとえば、
		`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

Token	ESS_STR_T	認証されるユーザーのユーザー名とパスワードを表すトークン。
pDbCount	ESS_PUSHORT_T	アクセス可能なアプリケーションとデータベースのカウントを受け取る変数のアドレス。

パラメータ	データ型	説明
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース名構造体の配列を受け取るポインタのアドレス。
phCtx	ESS_PHCTX_T	Essbase サーバー・コンテキスト・ハンドルへのポインタ。

備考

- この関数が失敗した場合は、ユーザーのユーザー名とパスワードを確認するために、対応する EssLogin 関数が自動的に呼び出されます。
- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放する必要があります。

戻り値

成功の場合、Essbase サーバー・コンテキスト・ハンドルが phCtx に戻され、他の API 関数への後続の呼出しで引数として使用できます。また、指定したユーザーがアクセス可能なデータベース数が pCount に戻され、アクセス可能なアプリケーションおよびデータベースのリストが ppDbList に戻されます。

アクセス

この関数を呼び出す前に、EssInit を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

関連トピック

- EssLogin
- EssLoginAs
- EssLoginExAs
- EssAutoLogin
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssLoginExAs

管理者ユーザー名とパスワードではなくユーザー認証トークンを使用して、管理者を別のユーザーとして Essbase サーバーにログインさせます。この関数は通常、EssInit の呼出しを正しく実行した後で、かつコンテキスト・ハンドルの引数が必要な他の API の呼出しを実行する前に呼び出す必要があります。

別なユーザーとしてログインすることで、管理者はユーザーごとの許可を設定して、レポートのスケジュールを作成できます。

構文

```
ESS_FUNC_M EssLoginExAs (
hInstance, Server, Token, UserNameAs, pDbCount, ppDbList, phCtx
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。

Server	ESS_STR_T	ネットワーク・サーバー名の文字列。
		サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
		`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
		保護モード(SSL)の場合、URL の構文は次のとおりです
		`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
		たとえば、
		`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

Token	ESS_STR_T	認証されるユーザーのユーザー名とパスワードを表すトークン。
UserNameAs	ESS_STR_T	代行するユーザーのユーザー名文字列。
pDbCount	ESS_PUSHORT_T	アクセス可能なアプリケーションとデータベースのカウントを受け取る変数のアドレス。
ppDbList	120 ページの「ESS_APPDB_T」	割り当てられたアプリケーション/データベース名構造体の配列を受け取るポインタのアドレス。
phCtx	ESS_PHCTX_T	Essbase サーバー・コンテキスト・ハンドルへのポインタ。

備考

- この関数が失敗した場合は、ユーザーのユーザー名とパスワードを確認するために、対応する `EssLoginAs` 関数が自動的に呼び出されます。
- ppDbList に対して割り当てられたメモリーは、`EssFree` を使用して解放する必要があります。

戻り値

成功の場合、Essbase サーバー・コンテキスト・ハンドルが phCtx に戻され、他の API 関数への後続の呼出しで引数として使用できます。また、指定したユーザーがアクセス可能なデータベース数が pCount に戻され、アクセス可能なアプリケーションおよびデータベースのリストが ppDbList に戻されます。

アクセス

別のユーザーとしてログインするには、管理者である必要があります。

この関数を呼び出す前に、`EssInit` を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

関連トピック

- EssLogin
- EssLoginAs
- EssAutoLogin
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssLoginSetPassword

ユーザーをログインさせ、パスワードを変更します。パスワードが失効した場合、または次のログイン時に変更が必要な場合にこの関数を使用します。

構文

パラメータ	データ型	説明
HInstance	ESS_HINST_T	API インスタンス・ハンドル。

Server	ESS_STR_T	ネットワーク・サーバー名の文字列。
		サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
		`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
		保護モード(SSL)の場合、URL の構文は次のとおりです
		`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
		たとえば、
		`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

UserName	ESS_STR_T	ユーザー名。
Password	ESS_STR_T	旧パスワード。
NewPassword	ESS_STR_T	新パスワード。
pDbCount	ESS_PUSHORT_T	アクセス可能なデータベースの数。
ppDbList	ESS_PPAPPDB_T	アクセス可能なアプリケーションデータベース構造体の配列へのポインタのアドレス。
phCtx	ESS_PHCTX_T	コンテキスト・ハンドルへのポインタ。

備考

- この関数は、EssLogin を呼び出し、ステータス・コード 1051090(パスワードが期限切れ)または 1051093(すぐにパスワードを変更)を受け取った後に呼び出します。
- Microsoft Windows では、EssLoginSetPassword のかわりに EssAutoLogin を使用することを検討してください。
- ppDbList に対して割り当てられたメモリーは、EssFree を使用して解放します。

戻り値

成功の場合、この関数は次の値を戻します:

- hCtx では、コンテキスト・ハンドル。
- pDbCount に、ユーザーがアクセス可能なデータベースの数。
- ppDbList に、アクセス可能なアプリケーションデータベース構造体の配列へのポインタ。

アクセス

この関数を呼び出す前に、EssInit を呼び出して API を初期化し、有効なインスタンス・ハンドルを取得します。

例

```
    ESS_FUNC_M
ESS_LoginSetPassword (ESS_HINST_T  hInst)
{
 ESS_FUNC_M  sts = ESS_STS_NOERR;
 ESS_HCTX_T  hCtx;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;
 ESS_PAPPDB_T pAppsDbs = NULL;
 ESS_STR_T   SvrName;
 ESS_STR_T   User;
 ESS_STR_T   Password;
 ESS_STR_T   NewPassword;

 SvrName   = "POPLAR";
 User     = "Joseph";
 Password   = "Password";
 NewPassword = "NewPassword";

 sts = EssLoginSetPassword (hInst, SvrName, User, Password, NewPassword
             &Items, &pAppsDbs, &hCtx);
 if (!sts)
 {
  for (ind = 0; ind < Items; ind++)
  {
   if ((pAppsDbs+ind) != NULL)
   {
    if ((pAppsDbs[ind].AppName != NULL) &&
      (pAppsDbs[ind].DbName  != NULL))
    {
```

```
     printf ("%s\r\n", pAppsDbs[ind].AppName);
     printf ("%s\r\n", pAppsDbs[ind].DbName);
    }
   }
  }
  if (pAppsDbs)
  EssFree(hInst,pAppsDbs);
 }
 return(sts);
}
```

関連トピック

- EssAutoLogin
- EssInit
- EssListDatabases
- EssLogout
- EssSetActive

EssLogout

Essbase サーバーからユーザーをログアウトさせます。

構文

```
ESS_FUNC_M EssLogout (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	ログアウトする API コンテキスト・ハンドル。

備考

- この関数は指定されたコンテキスト・ハンドルが示すログインのみをログアウトさせます。その他のログインまたはコンテキストは、同じユーザー名を使用していても影響を受けません。
- この関数はログイン・コンテキストについてのみ使用してください。ローカル・コンテキストには、EssDeleteLocalContext 関数を使用します。

戻り値

なし。

アクセス

この関数を呼び出すには、呼出し元が、EssLogin または EssAutoLogin 関数を使用して正常にログインしている必要があります。

例

```
ESS_FUNC_M
```

```
ESS_Logout (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 sts = EssLogout (hCtx);
 return(sts);
}
```

関連トピック

- EssAutoLogin
- EssDeleteLocalContext
- EssGetActive
- EssLogin
- EssLogoutUser

EssLogoutUser

スーパーバイザまたはアプリケーション・デザイナが他のユーザーを Essbase サーバーから切断できるようにします。

構文

```
ESS_FUNC_M EssLogoutUser (
hCtx, LoginId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	ログアウトを強制するユーザーの API コンテキスト・ハンドル。
LoginId	ESS_LOGINID_T	ログアウトされるユーザーのログイン ID。

備考

- LoginId は、EssListConnections 関数によって戻されるユーザー情報構造体から取得できます。
- この関数は指定された LoginID が示すログインのみをログアウトさせます。その他のログインまたはコンテキストは影響を受けません。
- スーパーバイザは、hCtx にログインしたサーバーにログインしたユーザーをログアウトできます。アプリケーション・デザイナは、hCtx をアプリケーション・デザイナとして、アプリケーションに接続しているユーザーのみをログアウトできます。自分自身はログアウトできません。

戻り値

なし。

アクセス

この関数を呼び出すには、スーパーバイザまたはアプリケーション・デザイナ権限を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LogoutUser (ESS_HCTX_T hCtx,
ESS_HINST_T hInst)
{
ESS_FUNC_M sts = ESS_STS_NOERR;
ESS_USHORT_T usrcnt;
ESS_PUSERINFO_T users;
sts = EssListConnections(hCtx, &usrcnt,
&users);
if(!sts)
{
if(usrcnt > 0)
{
/*************************************
* Log out first user from the list *
*************************************/
sts = EssLogoutUser(hCtx, users[0].LoginId);
if(!sts)
EssFree(hInst, users);
}
}
return(sts);
}
```

関連トピック

- EssListConnections
- EssLogout

EssLogSize

Essbase サーバー・ログ・ファイル(essbase.log)のサイズ、またはアプリケーション・ログ・ファイル(appname.log)のサイズを戻します。

構文

```
ESS_FUNC_M EssLogSize (
hCtx, AgentLog, pszAppName, pulLogSize
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AgentLog	ESS_BOOL_T	TRUE の場合、Essbase サーバー・ログ・ファイル(essbase.log)のサイズが戻されます。FALSE の場合、アプリケーション・ログ・ファイル(appname.log)のサイズが戻されます。
pszAppName	ESS_STR_T	アプリケーション名。
pulLogSize	ESS_PULONG_T	戻されるログ・ファイルのサイズ。

備考

- メッセージ・ログを表示するには、EssGetLogFile を使用します。
- essbase.log および appname.log の場所は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するのに、呼出し元がアクセス権限を持っている必要はありません。

例

```
    ESS_FUNC_M ESS_LogSize (ESS_HCTX_T hCtx)
{
 ESS_STR_T   pszAppName = NULL;
 ESS_ULONG_T  ulLogSize = 0;
 ESS_FUNC_M   sts = ESS_STS_NOERR;

 pszAppName = "Sample";

 /*
  * Get the log file size for the "Sample" application.
  */
 sts = EssLogSize(hCtx, ESS_FALSE, pszAppName, &ulLogSize);

 return(sts);
}
```

関連トピック

- EssDeleteLogFile
- EssGetLogFile
- EssWriteToLogFile

EssLROAddObject

レポート・オブジェクトを Essbase データベースのデータ・セルにリンクします。

構文

```
      ESS_FUNC_M EssLROAddObject (
      hCtx, memCount, pMemComb, usOption,
pLRODesc
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
memCount	ESS_ULONG_T	pMemComb で指定したメンバー数。
pMemComb	ESS_PVOID_T	リンクされるデータ・セルを定義するメンバー名の配列。
usOption	ESS_USHORT_T	オブジェクトの保管先を指定するオプション。次のいずれかの値を使用します: ● ESS_STORE_OBJECT_API オブジェクトをサーバーに保管します。 ● ESS_NOSTORE_OBJECT_API オブジェクトをサーバーに保管しません。
pLRODesc	114 ページの「ESS_LRODESC_API_T」	オブジェクト記述構造体へのポインタ。

備考

- リンク・オブジェクトは、次のいずれかのタイプです:
  - Word 文書、Excel スプレッドシートまたはビットマップ・イメージなどのフラット・ファイル。
  - 最大 599 文字のテキストが含まれるセル・ノート。
  - URL へのリンク。
  - その他の Essbase データベースへのリンク(リンク・パーティション機能)。
- オブジェクトをサーバーに保管しないことを選択した場合(usOption)は、アプリケーションがそのオブジェクトのすべてのファイル管理タスクを担当します(つまり、オブジェクトは Essbase データベースに保管されないため、他のプログラムが担当する必要があります)。
- セル・ノートは、常にサーバーに保管されるため、usOption パラメータは無視されます。
- URL リンク・オブジェクトでの usOption パラメータは、常に ESS_NOSTORE_OBJECT_API である必要があります。
- EssLROAddObject は、現在ログインしているユーザー名をオブジェクトの"created by"ユーザー名として使用し、pLRODesc オブジェクト記述構造体で指定されたユーザー名は無視します。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_STS_T  ESS_LROAddObject (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
    ESS_STS_T           sts = ESS_STS_NOERR;
```

```
    ESS_PMBRNAME_NONUNI_T  pMemComb = NULL;
    ESS_LRODESC_API_T       lroDesc;
    ESS_USHORT_T      usOption = 0;
    ESS_ULONG_T       memCount;

    memset (&lroDesc, 0 , sizeof(ESS_LRODESC_API_T));

    lroDesc.usObjType = 0;   /* Creating a cell note */
    strcpy(lroDesc.lro.note, "The profit for Colas in the East based on actuals");
    usOption = ESS_NOSTORE_OBJECT_API;
    strcpy(lroDesc.userName, "user1");
    memCount = 5;

    sts = EssAlloc(hInst, memCount*sizeof(ESS_MBRNAME_NONUNI_T),
                  (ESS_PPVOID_T)&pMemComb);
    if (sts)
    {
        printf("could not allocate memory\n");
        return sts;
    }

    memset(pMemComb, 0, memCount*sizeof(ESS_MBRNAME_NONUNI_T));
    strcpy( pMemComb[0], "Profit");
    strcpy( pMemComb[1], "East");
    strcpy( pMemComb[2], "Actual");
    strcpy( pMemComb[3], "Colas");
    strcpy( pMemComb[4], "Year");

    sts = EssLROAddObject( hCtx, memCount, pMemComb, usOption, &lroDesc);

    if (sts)
    {
        printf( "Could not attach LRO\n");
    }
    EssFree(hInst, pMemComb);
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetObject
- EssLROUpdateObject
- EssLRODeleteObject

EssLRODeleteCellObjects

Essbase データベースの指定されたデータ・セルにリンクされているすべてのオブジェクトを削除します。セルにリンクされている特定のオブジェクトを削除するには、EssLRODeleteObject を使用します。

構文

```
ESS_FUNC_M EssLRODeleteCellObjects (
```

```
    hCtx, memCount, pMemComb, pulLROCount, pLRODescList
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
memCount	ESS_ULONG_T	pMemComb で指定されたメンバーの数。
pMemComb	ESS_PVOID_T	メンバー名の配列。
pulLROCount	ESS_ULONG_T	削除された LRO カタログ・エントリ数。
pLRODescList	114 ページの「ESS_LRODESC_API_T」	削除された LRO カタログ・エントリのリスト。

備考

- この関数では、指定されたセルにリンクされているすべてのオブジェクトがそれらのカタログ・エントリと一緒に削除されます。
- オブジェクトがサーバーに保管されていない場合は、セル・リンクのみ破棄され、ファイルは削除されません。
- 呼出し元は、pLRODescList に割り当てられたメモリーの解放に責任を持ちます。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LRO DeleteCellObjects (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
    ESS_LRODESC_API_T        plroDescList=NULL;
    ESS_PMBRNAME_NONUNI_T  pMemComb = NULL;
    ESS_ULONG_T       memCount;
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_ULONG_T       ulLroCount;
    memCount = 5;
    sts = EssAlloc(hInst, memCount*sizeof(ESS_MBRNAME_NONUNI_T),
           (ESS_PPVOID_T)&pMemComb);
    if(sts)
    {
        printf("Could not allocate memory \n");
        return sts;
    }
    memset(pMemComb, 0, memCount*sizeof(ESS_MBRNAME_NONUNI_T));
    strcpy( pMemComb[0], "Profit");
    strcpy( pMemComb[1], "East");
    strcpy( pMemComb[2], "Actual");
    strcpy( pMemComb[3], "Colas");
```

```
    strcpy( pMemComb[4], "Year");
    sts = EssLRODeleteCellObjects(hCtx, memCount, pMemComb, &ulLroCount,
&plroDescList);
    if (sts)
    {
        printf ("Could not delete cell objects. \n");
    }
    EssFree( hInst, pMemComb);
    if (plroDescList)
        EssFree(hInst, plroDescList);
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROAddObject
- EssLRODeleteObject
- EssLROPurgeObjects

EssLRODeleteObject

Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを削除します。セルにリンクされたすべてのオブジェクトを削除するには、EssLRODeleteCellObjects を使用します。

構文

```
ESS_FUNC_M EssLRODeleteObject (
hCtx, plinkId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
plinkId	114 ページの「ESS_LROHANDLE_API_T」	オブジェクト識別構造体に対するポインタ。

備考

- 指定されたオブジェクトは削除され、カタログ・リストからも除外されます。
- オブジェクトがサーバーに保管されていない場合は、セル・リンクのみ破棄され、ファイルは削除されません。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_FUNC_M  Ess_LRO DeleteObject (ESS_HCTX_T hCtx)
{
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_LROHANDLE_API_T   linkId;
    memset(&linkId, 0, sizeof(ESS_LROHANDLE_API_T));
    linkId.hObject = 26;
    linkId.cellKey.cellOffset = 282;
    linkId.cellKey.blkOffset = 113;
    linkId.cellKey.segment = 0;
    sts = EssLRODeleteObject(hCtx, &linkId);
    if (sts)
    {
        printf("Could not delete object\n");
    }
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROAddObject
- EssLRODeleteCellObjects
- EssLROPurgeObjects

EssLROGetCatalog

Essbase データベース内の指定したデータ・セルについて、LRO カタログ・エントリのリストを取得します。

構文

```
ESS_FUNC_M EssLROGetCatalog (
hCtx, memCount, pMemComb, pulLROCount, ppLRODescList)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
memCount	ESS_ULONG_T	pMemComb で指定されたメンバーの数。
pMemComb	ESS_PMBRNAMECOMB_T	メンバー名の配列。
pulLROCount	ESS_ULONG_T *	呼出し元に戻される LRO カタログ・エントリの数。
pLRODescList	114 ページの「ESS_LRODESC_API_T」	呼出し元に戻される LRO カタログ・エントリのリストへのポインタのアドレス。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LRO GetCatalog (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
    ESS_PMBRNAME_NONUNI_T  pMemComb = NULL;
    ESS_PLRODESC_API_T       plroDescList=NULL;
    ESS_USHORT_T      usOption = 0;
    ESS_ULONG_T       memCount;
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_ULONG_T       ulLroCount;
    memCount = 5;
    sts = EssAlloc(hInst, memCount*sizeof(ESS_MBRNAME_NONUNI_T),
          (ESS_PPVOID_T)&pMemComb);
    if(sts)
    {
        printf("Could not allocate memory \n");
        return sts;
    }
    memset(pMemComb, 0, memCount*sizeof(ESS_MBRNAME_NONUNI_T));
    strcpy( pMemComb[0], "Profit");
    strcpy( pMemComb[1], "East");
    strcpy( pMemComb[2], "Actual");
    strcpy( pMemComb[3], "Colas");
    strcpy( pMemComb[4], "Year");
    sts = EssLROGetCatalog(hCtx, memCount, pMemComb, &ulLroCount, &plroDescList);
    if (sts)
    {
        printf ("Could not get the catalog \n");
    }
    EssFree(hInst, pMemComb);
    if(plroDescList)
    {
        EssFree(hInst, pMemComb);
    }
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetCatalogBatch
- EssLROAddObject
- EssLROUpdateObject
- EssLROGetObject
- EssLRODeleteObject

EssLROGetCatalogBatch

Essbase データベース内の指定した複数のデータ・セルについて、LRO カタログ・エントリのリストを取得します。

構文

```
ESS_FUNC_M EssLROGetCatalogBatch (
hCtx, memCount, pMemComb, cellCount, pulLROCount, ppLRODescList)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
memCount	ESS_ULONG_T *	pMemComb でセルごとに指定された「メンバー数」の配列。
pMemComb	ESS_PMBRNAMECOMB_T *	「メンバー名」の組合せの配列。配列自体の各要素は、セルごとのメンバー名の配列です。
cellCount	ESS_ULONG_T	LRO セルのカウント。
pulLROCount	ESS_ULONG_T *	呼出し元に戻された「LRO カタログ・エントリ数」の配列。配列内の各要素は、入力セルの LRO カタログ・エントリの数に対応しています。
pLRODescList	114 ページの「ESS_LRODESC_API_T」	呼出し元に戻される LRO カタログ・エントリのリストへのポインタのアドレス。

備考

この関数を使用するには、初期化構造体 ESS_INIT_T の MaxBuffer フィールドを 0xFFFFFFFF バイトに設定して、プログラムを初期化します。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して読取り権限 (ESS_PRIV_READ)を持っている必要があります。

例

```
    /*
* ESS_GetLinkedObjectCatalogBatch() -- Gets a list of LRO description for a list of
given data cell
* From the Database Sample.Basic, it will fetch LROs for the following Cells.
* 1) "Jan", "Sales", "100-10", "New York", "Actual"
* 2) "Feb", "COGS", "200-10", "Utah", "Budget"
* 3) "Mar", "Payroll", "300-10", "Texas", "Variance"
*/
ESS_STS_T ESS_GetLinkedObjectCatalogBatch(ESS_HINST_T hInst, ESS_HCTX_T hCtx)
{
 ESS_STS_T       status = 0;
 ESS_UINT_T       memberLength = ESS_MBRNAMELEN_NONUNI;
```

```
 ESS_PMBRNAME_NONUNI_T  *ppMemComb=NULL;
 ESS_PMBRNAME_NONUNI_T  pMemComb = NULL;
 ESS_ULONG_T      *pulLroCount= NULL;
 ESS_PLRODESC_API_T   pLroDescList = NULL;
 ESS_PLRODESC_API_T   *ppLroDescList = NULL;
 ESS_ULONG_T      cellCount = 3;      /* Number of cells for which to retrieve LROs
*/
 ESS_ULONG_T      mbrsCount[3] = {5, 5, 5}; /* Number of members in combinations for
each cell */

 ESS_ULONG_T      i,j,k,offset;
 ESS_CHAR_T      *pMember = NULL;
 ESS_CHAR_T       response;

 status = EssAlloc(hInst, cellCount * sizeof(ESS_PMBRNAMECOMB_T),
(ESS_PPVOID_T)&ppMemComb);
 if (status)
  goto exit;

 /* Member combination for Cell # 1 */
 status = EssAlloc(hInst, mbrsCount[0] * memberLength,
(ESS_PPVOID_T)&(ppMemComb[0]));
 if (status)
  goto exit;
 pMemComb = ppMemComb[0];
 memset(pMemComb, 0, mbrsCount[0]* memberLength);
 strcpy((pMemComb)[0], "Jan");
 strcpy((pMemComb)[1], "Sales");
 strcpy((pMemComb)[2], "100-10");
 strcpy((pMemComb)[3], "New York");
 strcpy((pMemComb)[4], "Actual");

 /* Member combination for Cell # 2 */
 status = EssAlloc(hInst, mbrsCount[1] * memberLength,
(ESS_PPVOID_T)&(ppMemComb[1]));
 if (status)
  goto exit;
 pMemComb = ppMemComb[1];
 memset(pMemComb, 0, mbrsCount[1] * memberLength);
 strcpy((pMemComb)[0], "Feb");
 strcpy((pMemComb)[1], "COGS");
 strcpy((pMemComb)[2], "200-10");
 strcpy((pMemComb)[3], "Utah");
 strcpy((pMemComb)[4], "Budget");

 /* Member combination for Cell # 3 */
 status = EssAlloc(hInst, mbrsCount[2] * memberLength,
(ESS_PPVOID_T)&(ppMemComb[2]));
 if (status)
  goto exit;
 pMemComb = ppMemComb[2];
 memset(pMemComb, 0, mbrsCount[2] * memberLength);
 strcpy((pMemComb)[0], "Mar");
 strcpy((pMemComb)[1], "Payroll");
 strcpy((pMemComb)[2], "300-10");
 strcpy((pMemComb)[3], "Texas");
 strcpy((pMemComb)[4], "Variance");
```

```
 /* Will hold inormation about how many LROs fetched for each Cell */
 status = EssAlloc(hInst, cellCount * sizeof(ESS_ULONG_T),
(ESS_PPVOID_T)&pulLroCount);
 if (status)
  goto exit;
 memset(pulLroCount, 0, cellCount * sizeof(ESS_ULONG_T));

 ppLroDescList = &pLroDescList;

 status = EssLROGetCatalogBatch(hCtx, mbrsCount, ppMemComb, cellCount, pulLroCount,
ppLroDescList);
 if (status)
  goto exit;


 for (k=0, offset=0; k<cellCount; k++)
 {
  ESS_LRODESC_API_T *pLroDesc = &pLroDescList[offset];
  for (i=0; i<pulLroCount[k]; i++, offset++)
  {
   printf("***** information for linked object ********\n");
   printf("Object type - %2d\n", (pLroDesc+i)->usObjType);
   printf("Link Id : \n");
   printf("  Object handle  -  %d \n", (pLroDesc+i)->linkId.hObject);
   printf("  Cell offset   -  %d \n", (pLroDesc+i)->linkId.cellKey.cellOffset);
   printf("  Block offset  -  %lf \n", (pLroDesc+i)->linkId.cellKey.blkOffset);
   printf("  Segment     -  %lf \n", (pLroDesc+i)->linkId.cellKey.segment);
   if ((pLroDesc+i)->usObjType > 0)
   {
    printf("Object name - %s\n",(pLroDesc+i)->lro.lroInfo.objName);
    printf("Object description - %s\n",(pLroDesc+i)->lro.lroInfo.objDesc);
   }
   else
   {
    printf("Cell notes - %s\n",(pLroDesc+i)->lro.note);
   }
   printf("User name - %s\n", (pLroDesc+i)->userName);
   printf("Security Access Level - %d\n", (pLroDesc+i)->accessLevel);
   if ((pLroDesc+i)->pMemComb)
   {
    printf("Member Name : \n");
    pMember = (ESS_CHAR_T *) (pLroDesc+i)->pMemComb;
    for (j=0; j < (pLroDesc+i)->memCount; j++)
    {
     printf("    %s\n",pMember);
     pMember += memberLength;
    }
    EssFree(hInst, (pLroDesc+i)->pMemComb);
   }
   printf("\n");
  }
 }
 printf("********* complete **********\n");

exit:
 if (status)
```

```
   printf("Fail Getting Catalog Information.\n");

  if (ppMemComb)
  {
   for(i=0; i<cellCount; i++)
   {
    EssFree(hInst, ppMemComb[i]);
   }
   EssFree(hInst, ppMemComb);
  }

  if (pLroDescList)
   EssFree(hInst, pLroDescList);

  if (pulLroCount)
   EssFree(hInst, pulLroCount);

  return(status);

}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetCatalog
- EssLROAddObject
- EssLROUpdateObject
- EssLROGetObject
- EssLRODeleteObject

EssLROGetObject

Essbase データベース内のデータ・セルにリンクされているオブジェクトを取得します。

構文

```
      ESS_FUNC_M EssLROGetObject (
      hCtx, plinkId, targetFile, usOption,
pRetLRODesc
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
plinkId ；	114 ページの「ESS_LROHANDLE_API_T」	オブジェクト識別構造体に対するポインタ。
targetFile	ESS_STR_T	オブジェクトが取得されるターゲット・ファイル名。

パラメータ	データ型	説明
usOption	ESS_USHORT_T	オブジェクトとカタログ・エントリの一方またはその両方を取得するかどうかを指定するオプション。次のいずれかを使用します: ● ESS_LRO_OBJ_API: オブジェクトのみを取得します。 ● ESS_LRO_CATALOG_API: カタログ・エントリのみ取得します。 ● ESS_LRO_BOTH_API: オブジェクトとカタログ・エントリを取得します。
pRetLRODesc	114 ページの「ESS_LRODESC_API_T」	オブジェクト記述構造体へのポインタ。

備考

セル・ノートはオブジェクトのカタログ・エントリの一部です。セル・ノートを取得するには、usOption パラメータの ESS_LRO_CATALOG_API を使用します。リンクされているノートが構造体 114 ページの「ESS_LRODESC_API_T」に含まれます。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LRO GetObject (ESS_HCTX_T hCtx)
{
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_LROHANDLE_API_T   linkId;
    ESS_LRODESC_API_T    lroDesc;
    ESS_USHORT_T      usOption = 2;  /* Default is catalog */
    ESS_CHAR_T       targetFile[ESS_ONAMELEN_API];
    memset(&lroDesc, 0, sizeof(ESS_LRODESC_API_T));
    memset(&linkId, 0, sizeof(ESS_LROHANDLE_API_T));
    /* Linked object is a LRO. (Windows Application) */
    linkId.hObject = 4;
    linkId.cellKey.cellOffset = 136;
    linkId.cellKey.blkOffset = 113.0;
    linkId.cellKey.segment = 0.0;
    usOption = ESS_LRO_BOTH_API ;  /* Get the catalog and the object */
    strcpy ( targetFile , "c:\\temp\\lrofile");
    sts = EssLROGetObject(hCtx, &linkId, targetFile, usOption, &lroDesc);
    if (sts)
    {
        printf("Could not get object\n");
    }
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROAddObject
- EssLROUpdateObject
- EssLRODeleteObject

EssLROListObjects

指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトのリストを取得します。

構文

```
ESS_FUNC_M EssLROListObjects (
hCtx, userName, listDate, pulLROCount, pLRODescList)
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
userName	ESS_CHAR_T	ユーザー名。指定した場合は、指定したユーザーが最後に変更したすべてのオブジェクトのリストが戻されます。
listDate	ESS_TIME_T	変更日。指定した場合は、特定日以前に変更されたすべてのオブジェクトのリストが戻されます。時刻は 1970 年 1 月 1 日以降に経過した秒数を ULONG 値で示します。
pulLROCount	ESS_ULONG_T *	戻される LRO カタログ・エントリの数。
pLRODescList;	114 ページの「ESS_LRODESC_API_T」	戻される LRO カタログ・エントリへのポインタのアドレス。

備考

- userName および listDate パラメータの両方を指定した場合、両方の基準に合致するオブジェクトがリストされます。
- 呼出し元は、pLRODescList に割り当てられたメモリーの解放に責任を持ちます。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LRO ListObjects (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
```

```
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_LRODESC_API_T    plroDescList=NULL;
    ESS_ULONG_T       ulLroCount;
    ESS_CHAR_T       userName[ESS_USERNAMELEN];
    ESS_CHAR_T       listDate[ESS_DATESIZE];
    ESS_CHAR_T       buf[ESS_DATESIZE];
    ESS_TIME_T       timestamp;
    struct tm       *pTmStruct, time_str;
    strcpy( userName, "user1");
    strcpy( listDate, "09/05/1997");

    time(&timestamp);
    pTmStruct = localtime((ESS_PLONG_T)&timestamp);
    memset(&time_str, 0, sizeof(struct tm));
    strncpy (buf, (const char *)&listDate[8], 2);
    time_str.tm_year = atoi(buf);
    strncpy(buf, listDate, 2);
    time_str.tm_mon = atoi(buf)-1;
    strncpy(buf, (const char *)&listDate[3], 2);
    time_str.tm_mday = atoi(buf);
    time_str.tm_hour = 0;
    time_str.tm_min = 0;
    time_str.tm_sec = 1;
    time_str.tm_isdst = -1;
    if ((time_str.tm_mon != pTmStruct->tm_mon) ||
        (time_str.tm_year != pTmStruct->tm_year) ||
        (time_str.tm_mday != pTmStruct->tm_mday))
    {
        time_str.tm_mday++;
        timestamp = mktime(&time_str);
    }
    sts = EssLROListObjects(hCtx, userName, timestamp, &ulLroCount, &plroDescList);
    if(sts)
    {
        printf("Could not list linked objects. \n");
    }
    if (plroDescList)
        EssFree(hInst, plroDescList);
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetCatalog
- EssLROPurgeObjects

EssLROPurgeObjects

指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトを削除します。

構文

```
ESS_FUNC_M EssLROPurgeObjects (
hCtx, userName, purgeDate, pulLROCount, pLRODescList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
userName	ESS_STR_T	ユーザー名を指すポインタ。指定した場合、特定のユーザーが最後に変更したすべてのオブジェクトが削除されます。
purgeDate	ESS_TIME_T	変更日。指定した場合は、特定日以前に変更されたすべてのオブジェクトのリストが戻されます。日付は 1970 年 1 月 1 日以降に経過した秒数を ULONG 値で表します。
pulLROCount	ESS_ULONG_T	削除された LRO カタログ・エントリの数。
pLRODescList	114 ページの「ESS_LRODESC_API_T」	削除された LRO カタログ・エントリのリストへのポインタのアドレス。

備考

- userName および purgeDate パラメータの両方を指定した場合、両方の基準に合致するオブジェクトが削除されます。
- 呼出し元は、pLRODescList に割り当てられたメモリーの解放に責任を持ちます。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対してデザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_LRO PurgeObjects (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
    ESS_FUNC_M       sts = ESS_STS_NOERR;
    ESS_LRODESC_API_T    plroDescList=NULL;
    ESS_ULONG_T       ulLroCount;
    ESS_CHAR_T       userName[ESS_USERNAMELEN];
    ESS_CHAR_T       purgeDate[ESS_DATESIZE];
    ESS_CHAR_T       buf[ESS_DATESIZE];
    ESS_TIME_T       timestamp;
    struct tm       *pTmStruct, time_str;
    strcpy( userName, "user1");
    strcpy( purgeDate, "09/05/1997");
    time(&timestamp);
    pTmStruct = localtime((ESS_PLONG_T)&timestamp);
    memset(&time_str, 0, sizeof(struct tm));
    strncpy (buf, (const char *)&purgeDate[8], 2);
```

```
    time_str.tm_year = atoi(buf);
    strncpy(buf, listDate, 2);
    time_str.tm_mon = atoi(buf)-1;
    strncpy(buf, (const char *)&purgeDate[3], 2);
    time_str.tm_mday = atoi(buf);
    time_str.tm_hour = 0;
    time_str.tm_min = 0;
    time_str.tm_sec = 1;
    time_str.tm_isdst = -1;
    if ((time_str.tm_mon != pTmStruct->tm_mon) ||
        (time_str.tm_year != pTmStruct->tm_year) ||
        (time_str.tm_mday != pTmStruct->tm_mday))
    {
        time_str.tm_mday++;
        timestamp = mktime(&time_str);
    }
    sts = EssLROPurgeObjects(hCtx, userName, timestamp, &ulLroCount, &plroDescList);
    if(sts)
    {
        printf("Could not purge linked objects. \n");
    }
    if (plroDescList)
        EssFree(hInst, plroDescList);
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetCatalog
- EssLRODeleteObject
- EssLRODeleteCellObjects

EssLROUpdateObject

サーバーに LRO の更新済バージョンを保管します。

構文

```
ESS_FUNC_M EssLROUpdateObject (
hCtx, plinkId, usOption, pLRODesc
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
plinkId	114 ページの「ESS_LROHANDLE_API_T」	オブジェクト識別構造体に対するポインタ。

パラメータ	データ型	説明
usOption	ESS_USHORT_T	オブジェクトとカタログ・エントリの一方またはその両方を保管するかどうかを指定するオプション。次のいずれかを使用します: ● ESS_LRO_OBJ_API オブジェクトのみ保管します。 ● ESS_LRO_CATALOG_API カタログ・エントリのみ保管します。 ● ESS_LRO_BOTH_API オブジェクトとカタログ・エントリを保管します。
pLRODesc	114 ページの「ESS_LRODESC_API_T」	オブジェクト記述構造体へのポインタ。

備考

- リンク・オブジェクトは、次のいずれかのタイプです:
  - Word 文書、Excel スプレッドシートまたはビットマップ・イメージなどのフラット・ファイル。
  - 最大 599 文字のテキストが含まれるセル・ノート。
  - 別の Essbase データベースへのリンク(リンク・パーティション機能)。
- セル・ノートはオブジェクトのカタログ・エントリの一部です。セル・ノートを保管するには、ESS_LRO_CATALOG_API を usOption パラメータに使用します。リンクされているノートが構造体 114 ページの「ESS_LRODESC_API_T」に含まれます。
- オブジェクトを最後に変更したユーザー名と変更日も更新されます。

戻り値

正常終了の場合は、ESS_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_STS_T ESS_LRO UpdateObject (ESS_HCTX_T hCtx)
{
    ESS_STS_T        sts = ESS_STS_NOERR;
    ESS_LROHANDLE_API_T   linkId;
    ESS_LRODESC_API_T    lroDesc;
    ESS_USHORT_T      usOption = 2;  /* Default is catalog */

    memset (&linkId, 0, sizeof(ESS_LROHANDLE_API_T));
    memset (&lroDesc, 0, sizeof(ESS_LRODESC_API_T));

    linkId.hObject = 25;
    linkId.cellKey.cellOffset = 149;
    linkId.cellKey.blkOffset = 113.0;
    linkId.cellKey.segment = 0.0;
```

```
    /* Linked object is a LRO. (Windows Application) */
    lroDesc.usObjType = 1;

    /* Update both object and catalog */
    usOption = ESS_LRO_BOTH_API;
    strcpy (lroDesc.lro.lroInfo.objName, "e:\\lro\\lroex.c");
    strcpy (lroDesc.lro.lroInfo.objDesc, "My C file");

    strcpy (lroDesc.userName, "user1");
    lroDesc.linkId.hObject = linkId.hObject;

    sts = EssLROUpdateObject(hCtx, &linkId, usOption, &lroDesc);
    if (sts)
    {
        printf("Could not update linked object.\n");
    }
    return sts;
}
```

関連トピック

- 113 ページの「LRO の定数と構造体の定数(C)」
- EssLROGetObject
- EssLROAddObject
- EssLRODeleteObject

EssMdxTrig

MDX ステートメントで指定された操作に基づいてトリガーを操作します。MDX は特定のトリガーを作成、置換、削除、使用可能または使用不可にできます。

構文

```
ESS_FUNC_M EssMdxTrig (
hCtx, AppName, DbName, mdxStatement
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
mdxStatement	ESS_STR_T	特定のトリガーを作成、置換、削除、使用可能または使用不可にすることを指定する MDX ステートメント。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssDisplayTriggers
- EssListSpoolFiles
- EssGetSpoolFile
- EssDeleteSplFile
- EssDeleteAllSplFiles

EssMergeDatabaseData

2 つ以上のデータ・スライスを 1 つのデータ・スライスにマージします。オプションで、プライマリ・データベース・スライスを除外できます。

この関数は、集約ストレージ・データベースにのみ適用されます。

構文

```
ESS_FUNC_M EssMergeDatabaseData (
hCtx, AppName, DbName, ulOptions
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	NULL を使用します。関数は常に現在選択されているデータベースに適用されます。
DbName	ESS_STR_T	NULL を使用します。関数は常に現在選択されているデータベースに適用されます。
ulOptions	ESS_ULONG_T	次の定数のいずれかになります: • #define ESS_MERGE_DATABASE_DATA_ALL 1: すべてのデータ・スライスを 1 つにマージします。 • #define ESS_MERGE_DATABASE_DATA_INCREMENTAL 2: すべての増分スライスを 1 つにマージにしますが、プライマリ・スライスとはマージしません。実行後、スライスは 2 つになります。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

例

```
    void TestMergeDatabaseData(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
    ESS_STS_T        sts = ESS_STS_NOERR;
    ESS_SHORT_T       isAbortOnError;
    ESS_OBJDEF_T   Rules;
    ESS_OBJDEF_T   Data;
    ESS_PMBRERR_T  pMbrErr = NULL;
    ESS_PMBRUSER_T pMbrUser = NULL;
    ESS_ULONG_T       ulBufferId;
    ESS_ULONG_T       ulDuplicateAggregationMethod;
```

```
    ESS_ULONG_T      ulOptionsFlags;
    ESS_ULONG_T      ulSize;
    ESS_ULONG_T      ulBufferCnt;
    ESS_ULONG_T      ulCommitType ;
    ESS_ULONG_T      ulActionType;
    ESS_ULONG_T      ulOptions;
    ESS_ULONG_T      ulBufferIdAry[1];
    ESS_ULONG_T      options;

    printf("\nCreate the buffer:\n");
    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
    ulBufferId = 1;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* Server object */
    Rules.hCtx   = hCtx;
    Rules.AppName  = AppName;
    Rules.DbName  = DbName;
    Rules.ObjType  = ESS_OBJTYPE_RULES;
    Rules.FileName = "ddldinaq";
    Data.hCtx    = hCtx;
    Data.AppName  = AppName;
    Data.DbName   = DbName;
    Data.ObjType  = ESS_OBJTYPE_TEXT;
    Data.FileName  = "ddldinaq_slice1a";
    isAbortOnError = ESS_TRUE;

    printf("\nLoad into buffer:\n");
    sts = EssImportASO (hCtx, &Rules, &Data, &pMbrErr, pMbrUser, isAbortOnError,
ulBufferId);
    printf("EssImportASO sts: %ld\n",sts);
    if(pMbrErr)
       EssFreeMbrErr(hCtx, pMbrErr);

    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\Create a new slice:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_NEW_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);

    options = ESS_MERGE_DATABASE_DATA_ALL;
    printf("\nMerge all data into one slice:\n");
    sts = EssMergeDatabaseData(hCtx, AppName, DbName, options);
    printf("EssMergeDatabaseData sts: %ld\n",sts);
}
```

関連トピック

- EssLoadBufferInit
- EssBeginDataloadASO
- EssSendString
- EssEndDataload
- EssLoadBufferTerm
- EssImportASO
- EssUpdateFileASO
- EssUpdateFileUTF8ASO
- EssListExistingLoadBuffers

EssPartialDataClear

アクティブな集約ストレージ・データベースで十分に定義された対称的な領域で指定されたデータを消去します。領域から選択的にデータを消去する方法が 2 つあります:

- 物理的方法。指定した地域の中の入力セルが集約ストレージ・データベースから物理的に消去されます。物理的にデータを消去するプロセスが完了するには、消去されるデータのサイズではなく、入力データのサイズに比例した時間がかかります。
- 論理的方法。指定した地域内の入力セルが、消去対象のセルの値をゼロにする負の相殺値を持つ新しいデータ・スライスに書き込まれます。論理的にデータを消去するプロセスが完了するには、消去されるデータのサイズに比例した時間がかかります。

構文

```
ESS_FUNC_M EssPartialDataClear (
hCtx
, RegionSpec, bPhysical);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(ログイン済)
RegionSpec	ESS_STR_T	領域定義(有効な MDX セットの指定) 領域は対称的である必要があります。領域の任意の次元メンバーは、保管されるメンバーである必要があります。物理的にデータを消去するとき、領域のメンバーは、プライマリ階層および代替階層からの上位レベルのメンバーにできます。(領域が、代替階層からの上位レベルのメンバーを含んでいる場合、パフォーマンスの低下につながる可能性があります。)論理的にデータを消去するとき、領域のメンバーは、プライマリ階層からのみ上位レベル・メンバーにできます。メンバーは、動的メンバー(暗黙または明示の MDX 式を持つメンバー)や、属性次元からのメンバーではいけません。
bPhysical	ESS_BOOL_T	TRUE の場合、物理消去領域操作を使用した、領域のデータの消去を指定します。FALSE の場合または指定しない場合、データは、論理消去領域操作を使用して消去されます。

備考

- 呼出し元は、データを消去するためにデータベース・マネージャまたは管理者権限を持っている必要があります。

戻り値

この関数の戻り値は、正常に完了した場合は 0 です。それ以外の場合は、エラー・コードが戻されます。

アクセス

この関数は、集約ストレージ・データベースにのみ適用されます。

例

```
    ESS_FUNC_M
TestPartialDataClear(ESS_HCTX_T hCtx)
{
  ESS_STS_T    sts;
  ESS_STR_T    regionSpec="{Feb}";

/* Perform a logical clear of February data */
  sts = EssPartialDataClear(hCtx, regionSpec, ESS_FALSE);
  return(sts);
}
```

EssPartitionApplyOtlChangeFile

EssPartitionApplyOtlChangeFileEx に置き換えられましたが、このフォーマットは下位互換性のために維持されています。詳細は、EssPartitionApplyOtlChangeFileEx を参照してください。

構文

```
ESS_FUNC_M EssPartitionApplyOtlChangeFile (
hCtx, usFileName, ppszFileName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
usFileName	ESS_USHORT_T	アウトライン変更ファイルの数。
ppszFileName	ESS_PSTR_T	ファイル名の配列。配列サイズは usFileName で定義します。

EssPartitionApplyOtlChangeFileEx

ソース上でのターゲット・アウトラインに対するアウトライン変更ファイル(*.CHG)に適応されます。この関数は、EssPartitionGetOtlChanges とともにバッ

チ処理で使用するために設計されており、変更ファイルのリストを指定できます。この関数にはフィルタを使用できます。

アプリケーションとデータベースのペアでメタデータ方向とタイプが同じパーティションが複数ある場合は、この関数を EssPartitionApplyOtlChangeFile のかわりに使用します。

構文

```
ESS_FUNC_M EssPartitionApplyOtlChangeFileEx (
hCtx, usFileName, ppszFileName, usDataDirectionType
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
usFileName	ESS_USHORT_T	アウトライン変更ファイルの数。
ppszFileName	ESS_PSTR_T	ファイル名の配列。配列サイズは usFileName で定義します
usDataDirectionType	ESS_USHORT_T	次の方向タイプ定数のいずれか: #define ESS_PARTITION_DATA_SOURCE 0x0001 #define ESS_PARTITION_DATA_TARGET 0x0002

備考

EssPartitionGetOtlChanges は、変更ファイルの名前を戻します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナ権限が必要です。

例

```
    ESS_FUNC_M ESS_PartitionApplyOtlChangeFileEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M     sts;
 ESS_STR_T     hostname, appname, dbname;
 ESS_USHORT_T    usType, uscnt, dataFlowDir, *dataFlowDirs = ESS_NULL;
 ESS_ULONG_T    uldimfilter=0,ulmbrfilter=0,ulmbrattrfilter=0;
 ESS_PARTOTL_QUERY_T  MetaQuery;
 ESS_PARTOTL_CHG_FILE_T MetaChangeFile;
 ESS_PPART_INFO_T  partitionp  = NULL;

 memset(&MetaQuery, 0, sizeof(ESS_PARTOTL_QUERY_T));

 hostname = "local";
 appname = "app1";
 dbname = "src1";
 usType = ESS_PARTITION_OP_LINKED;
```

```
 dataFlowDir = ESS_PARTITION_DATA_SOURCE;
 uldimfilter   = ESS_DIMCHG_ALL;
 ulmbrfilter   = ESS_PARTITION_OTLMBR_ALL;
 ulmbrattrfilter = ESS_PARTITION_OTLPARTITION_OTLMBRATTR_ALL;
 MetaQuery.HostDatabase.pszHostName  = hostname;
 MetaQuery.HostDatabase.pszAppName   = appname;
 MetaQuery.HostDatabase.pszDbName    = dbname;
 MetaQuery.usOperationType       = usType;
 MetaQuery. usDataDirectionType    = dataFlowDir;
 MetaQuery.MetaFilter.TimeStamp    = 0;
 MetaQuery.MetaFilter.ulDimFilter   = uldimfilter;
 MetaQuery.MetaFilter.ulMbrFilter   = ulmbrfilter;
 MetaQuery.MetaFilter.ulMbrAttrFilter = ulmbrattrfilter;

 sts = EssPartitionGetOtlChanges(hCtx, &MetaQuery, &MetaChangeFile);

  if (!sts)
  sts = EssAlloc(hInst, MetaChangeFile.usFileNum *sizeof(ESS_USHORT_T),
&dataFlowDirs);

  if (!sts)
    for (uscnt=0;uscnt< MetaChangeFile.usFileNum;uscnt++)
      dataFlowDirs[uscnt] = dataFlowDir;


  if (!sts)
  {

     sts = EssPartitionApplyOtlChangeFile
    (hCtx, MetaChangeFile.usFileNum, MetaChangeFile.ppszFileName);

    printf("EssPartitionApplyOtlChangeFile  sts: %ld\n",sts);

  }
 if(&MetaChangeFile)  EssFree(hInst,&MetaChangeFile);

 if(&dataFlowDirs)  EssFree(hInst, &dataFlowDirs);

 return(sts);

}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile

- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionApplyOtlChangeRecs

ターゲット・アウトラインへのアウトライン変更に適用されます。この関数は、EssPartitionGetOtlChanges を呼び出した後に、EssPartitionReadOtlChangeFile を使用して対話的に使用するように設計されています。EssPartitionReadOtlChangeFile によって戻される変更ファイルを編集して拒否フラグを設定できます。拒否フラグは ESS_PARTOTL_SELECT_APPLY_T から参照した 173 ページの「ESS_PARTOTL_MBR_RSRVD_API_T」で設定されます。

構文

```
ESS_FUNC_M EssPartitionApplyOtlChangeRecs (
hCtx, pApplyRecords
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pApplyRecords	180 ページの「ESS_PARTOTL_SELECT_APPLY_T」	適用するレコード。

備考

- 変更レコード間には依存関係が存在する場合があります。
- レコードを拒否すると、別のレコードを適用するときに失敗することがあります。たとえば、「A を追加」と「AA を A の子として追加」という 2 つのレコードがあるとします。最初のレコードを拒否して 2 つ目のレコードを受け入れると、適用時にエラーが発生します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M  Ess_PartitionApplyOtlChangeRecs (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M          sts = ESS_STS_NOERR;
```

```
 ESS_PARTOTL_SELECT_APPLY_T  ApplyRecords;
 ESS_STR_T          chgfilename;
 ESS_TIME_T          time = 0;
 ESS_PARTOTL_CHANGE_API_T   OtlChg;
 ESS_ULONG_T         uldimfilter=0,ulmbrfilter=0,ulmbrattrfilter=0;
 ESS_PARTOTL_SELECT_CHG_T   SelectMetaRecords;
 ESS_PARTOTL_READ_T      MetaChangeRead;

 memset(&ApplyRecords,    0, sizeof(ESS_PARTOTL_SELECT_APPLY_T));
 memset(&SelectMetaRecords, 0, sizeof(ESS_PARTOTL_SELECT_CHG_T));
 memset(&MetaChangeRead,   0, sizeof(ESS_PARTOTL_READ_T));

 chgfilename = "C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\app1\\trg1\
\ess00001.chg";
 uldimfilter    = ESS_DIMCHG_ALL;
 ulmbrfilter    = ESS_PARTITION_OTLMBR_ALL;
 ulmbrattrfilter  = ESS_PARTITION_OTLPARTITION_OTLMBRATTR_ALL;

 SelectMetaRecords.pszFileName        = chgfilename;
 SelectMetaRecords.QueryFilter.TimeStamp    = time;
 SelectMetaRecords.QueryFilter.ulDimFilter   = uldimfilter;
 SelectMetaRecords.QueryFilter.ulMbrFilter   = ulmbrfilter;
 SelectMetaRecords.QueryFilter.ulMbrAttrFilter = ulmbrattrfilter;
 MetaChangeRead.pOtlChg = &OtlChg;
 sts = EssPartitionReadOtlChangeFile (hCtx, &SelectMetaRecords, &MetaChangeRead);
 printf("\tEssPartitionReadOtlChangeFile  sts: %ld\n",sts);
 if (!sts)
 {

    ApplyRecords.pszFileName = chgfilename;
    ApplyRecords.pOtlChg = MetaChangeRead.pOtlChg;
    ApplyRecords.SourceTime = MetaChangeRead.SourceTime;
  sts = EssPartitionApplyOtlChangeRecs(hCtx, &ApplyRecords);
    printf("EssPartitionApplyOtlChangeRecs  sts: %ld\n",sts);

 }
 sts = EssPartitionFreeOtlChanges(hCtx);
 return(sts);

}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile

- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionCloseDefFile

共有パーティションの定義ファイルを閉じます。

構文

```
ESS_FUNC_M EssPartitionCloseDefFile (
hCtx, iFileHandle
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	ネット・コンテキスト。
iFileHandle	ESS_INT_T	終了対象のファイル・ハンドル。

備考

この関数は、一連の定義操作の一部として使用します。

1. EssPartitionOpenDefFile を使用して、既存の定義ファイルを開きます。
2. EssPartitionNewDefFile を使用して、新規定義ファイルを作成し開きます。
3. EssPartitionReadDefFile を使用して定義ファイルを読み取ります。または、EssPartitionWriteDefFile を使用して定義ファイルに書き込みます。
4. EssPartitionCloseDefFile で閉じます。
5. EssPartitionFreeDefCtx でメモリーを解放します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

使用例については、EssPartitionNewDefFile を参照してください

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges

- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionFreeDefCtx

共有パーティションのコンテキスト構造体の下に割り当てられたメモリーを動的に解放します。

構文

```
ESS_FUNC_M EssPartitionFreeDefCtx (
hCtx, pDdbCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pDdbCtx	161 ページの「ESS_PART_T」	共有パーティションのコンテキストへのポインタ。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

例

使用例は、EssPartitionNewDefFile を参照してください。

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList

- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionFreeOtlChanges

EssPartitionReadOtlChangeFile ルーチンによって割り当てられたメモリーを解放します。このルーチンは、アウトライン変更レコードを処理した後に呼び出します。

構文

```
ESS_FUNC_M EssPartitionFreeOtlChanges (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

使用例は、EssPartitionReadOtlChangeFile を参照してください。

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile

- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionGetAreaCellCount

指定したスライス文字列内のセルの数を戻します。

構文

```
ESS_FUNC_M EssPartitionGetAreaCellCount (
hCtx, pszSlice, pdCount
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pszSlice	ESS_STR_T	確認対象の入力スライス定義。
pdCount	ESS_PDOUBLE_T	ここにセル数が戻されます。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M ESS_PartitionGetAreaCellCount(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_DOUBLE_T pdCount;
 ESS_STR_T   pszSlice;

 pszSlice = "@IDESC(East)";

 sts = EssPartitionGetAreaCellCount(hCtx, pszSlice, &pdCount);
 if (!sts)
 {  printf("EssPartitionGetAreaCellCount  sts: %ld\n",sts);
   printf("\tArea cell count = %g \n",pdCount);    }

 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile

- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionGetAreaLev0CellCount

指定されたスライス文字列内の次元のレベル 0 結合であるセル数を戻します。これは、複製パーティションのターゲットが集約ストレージ・キューブの場合に便利です。

構文

```
ESS_FUNC_M EssPartitionGetAreaLev0CellCount (
hCtx, pszSlice, pdCount
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pszSlice	ESS_STR_T	確認対象の入力スライス定義。
pdCount	ESS_PDOUBLE_T	ここにセル数が戻されます。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M ESS_PartitionGetAreaLev0CellCount(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_DOUBLE_T pdCount;
 ESS_STR_T   pszSlice;

 pszSlice = "@IDESC(East)";
```

```
 sts = EssPartitionGetAreaLev0CellCount(hCtx, pszSlice, &pdCount);
 if (!sts)
 {  printf("EssPartitionGetAreaLev0CellCount  sts: %ld\n",sts);
   printf("\tArea cell count = %g \n",pdCount);    }

 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionGetList

現在選択されているデータベースが関与しているパーティション定義のリストを戻します。

構文

```
ESS_FUNC_M EssPartitionGetList (
hCtx, pSelectPartition, pusCount, ppPartition
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pSelectPartition	181 ページの「ESS_PARTSLCT_T」	パーティションの選択条件。
pusCount	ESS_PUSHORT_T	戻されるパーティションの数。

パラメータ	データ型	説明
ppPartition	162 ページの「ESS_PART_INFO_T」	パーティション情報構造体において割り当てられた配列へのポインタ。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M ESS_PartitionGetList(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M      sts    = ESS_STS_NOERR;
 ESS_USHORT_T     op_types  = 0;
 ESS_USHORT_T     dir_types = 0;
 ESS_USHORT_T     meta_dir_types = 0;
 ESS_USHORT_T     count, i;
 ESS_PPART_INFO_T   partitionp  = NULL;
 ESS_PARTSLCT_T    SelectPartition;
 memset(&Selectpartition, 0, sizeof(ESS_PARTSLCT_T));

 op_types =  ESS_PARTITION_OP_REPLICATED |
       ESS_PARTITION_OP_LINKED |
       ESS_PARTITION_OP_TRANSPARENT;

 dir_types = ESS_PARTITION_DATA_SOURCE | ESS_PARTITION_DATA_TARGET;

 meta_dir_types = ESS_PARTITION_OTL_SOURCE | ESS_PARTITION_OTL_TARGET;


 SelectPartition.usOperationTypes = op_types;
 SelectPartition.usDirectionTypes = dir_types;
 SelectPartition.usMetaDirectionTypes = meta_dir_types;

 sts = EssPartitionGetList(hCtx, &SelectPartition, &count, &Partitionp);
 printf("EssPartitionGetList  sts:  %ld\n",sts);
 if (!sts)
 {
 printf("\n# Partitions matching input criteria: %d\n\n", (int)count);

 for (i = 0; i < count; i++)
 {
  ESS_PART_INFO_T *info = &partitionp[i];

  printf("%2d: %s %s %s: Host=%s App=%s Db=%s\n", i+1,
      info->OperationType==ESS_PARTITION_OP_REPLICATED ? "Replication" :
      info->OperationType==ESS_PARTITION_OP_LINKED ? "Link" :
      info->OperationType==ESS_PARTITION_OP_TRANSPARENT ? "Transparent" : "Unknown",
      info->DataDirection==ESS_PARTITION_DATA_SOURCE ? "Source" :
      info->DataDirection==ESS_PARTITION_DATA_TARGET ? "Target" :"Unknown",
      info->MetaDirection==ESS_PARTITION_OTL_SOURCE ? "(Outline Change Source)":
      info->MetaDirection==ESS_PARTITION_OTL_TARGET ? "(Outline Change
Target)" :"(Unknown Outline Change Type)",
      info->SvrName, info->AppName, info->DbName);
  printf("   Outline last changed: %s\n",
       info->LastMetaUpdateTime==0 ? "Never\n"
```

```
                    : ctime(&info->LastMetaUpdateTime));
  if (info->OperationType==ESS_PARTITION_OP_REPLICATED &&
    info->DataDirection==ESS_PARTITION_DATA_TARGET)
  {
   printf("   Last replicated: %s   %s\n",
       info->LastRefreshTime==0 ? "Never\n"
                  : ctime(&info->LastRefreshTime),
       info->PartitionUpdatable ? "Locally updatable" :
                 "Not locally updatable");
  }
  else if (info->OperationType==ESS_PARTITION_OP_REPLICATED &&
      info->DataDirection==ESS_PARTITION_DATA_SOURCE)
  {
   printf("   Last updated: %s   %s\n\n",
       info->LastUpdateTime==0 ? "Never\n"
                  : ctime(&info->LastUpdateTime),
       info->IncrRefreshAllowed ? "Incrementally replicatable" :
                   "Not incrementally replicatable");
  }
 }#/* end for */
 }#/* end if */
 if (partitionp) EssFree(hInst, partitionp);
 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionGetOtlChanges

アウトライン変更をソース・サーバーの.CHG ファイルから読み込み、ターゲット・サーバーの.CHG ファイルに書き込みます。この関数は、

EssPartitionApplyOtlChangeFile とともにバッチ処理で使用するか、または EssPartitionReadOtlChangeFile と EssPartitionApplyOtlChangeRecs を組み合せて対話的に使用するように設計されています。

構文

```
ESS_FUNC_M EssPartitionGetOtlChanges (
hCtx, pQuery, pChangeFile
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pQuery	178 ページの「ESS_PARTOTL_QUERY_T」	変更クエリー条件。
pChangeFile	169 ページの「ESS_PARTOTL_CHG_FILE_T」	呼出し元が割り当てた変更ファイル情報構造体。

備考

pChangeFile 内の変更ファイル名文字列を解放するには、EssPartitionFreeOtlChanges を呼び出します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M  Ess_PartitionGetOtlChanges(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M       sts;
 ESS_STR_T       hostname, appname, dbname;
 ESS_USHORT_T      usType, dataFlowDir;
 ESS_ULONG_T      uldimfilter=0,ulmbrfilter=0,ulmbrattrfilter=0;
 ESS_PARTOTL_QUERY_T   MetaQuery;
 ESS_PARTOTL_CHG_FILE_T MetaChangeFile;
 ESS_PPART_INFO_T    partitionp  = NULL;

 memset(&MetaQuery, 0, sizeof(ESS_PARTOTL_QUERY_T));

 hostname = "local";
 appname = "app1";
 dbname = "src1";
 usType = ESS_PARTITION_OP_LINKED;
 dataFlowDir = ESS_PARTITION_DATA_SOURCE;
 uldimfilter   = ESS_PARTITION_OTLDIM_ALL;
 ulmbrfilter   = ESS_PARTITION_OTLMBR_ALL;
 ulmbrattrfilter = ESS_PARTITION_OTLMBRATTR_ALL;
 MetaQuery.HostDatabase.pszHostName  = hostname;
 MetaQuery.HostDatabase.pszAppName   = appname;
 MetaQuery.HostDatabase.pszDbName    = dbname;
```

```
 MetaQuery.usOperationType      = usType;
 MetaQuery. usDataDirectionType    = dataFlowDir;
 MetaQuery.MetaFilter.TimeStamp    = 0;
 MetaQuery.MetaFilter.ulDimFilter   = uldimfilter;
 MetaQuery.MetaFilter.ulMbrFilter   = ulmbrfilter;
 MetaQuery.MetaFilter.ulMbrAttrFilter = ulmbrattrfilter;

 sts = EssPartitionGetOtlChanges(hCtx, &MetaQuery, &MetaChangeFile);
 printf("EssPartitionGetOtlChanges  sts: %ld\n",sts);
 if (!sts) {
 printf("\tNumber of meta change file found: %d\n",MetaChangeFile.usFileNum);
 printf("\tName of meta change file found: %s\n",MetaChangeFile.ppszFileName[0]);
  }
 if(&MetaChangeFile)  EssFree(hInst,&MetaChangeFile);

 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionGetReplCells

複製パーティションで識別されているすべてのデータ・セルを、ソース・データベースから選択したターゲット・データベースに複製します。

構文

```
ESS_FUNC_M EssPartitionGetReplCells (
hCtx, pReplicatePartition
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pReplicatePartition	164 ページの「ESS_PART_REPL_T」	パーティション情報。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M  Ess_PartitionGetReplCells(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M  sts;
 ESS_PART_REPL_T       ReplicatePartition;
 ESS_PART_CONNECT_INFO_T    HostDatabase;

 memset(&ReplicatePartition, 0, sizeof(ESS_PART_REPL_T));
 memset(&HostDatabase, 0, sizeof(ESS_PART_CONNECT_INFO_T));

 ReplicatePartition.pHostDatabase = &HostDatabase;


 ReplicatePartition.lPartitionCount = -1;
 ReplicatePartition.bUpdatedOnly = ESS_FALSE;

 sts = EssPartitionGetReplCells(hCtx, &ReplicatePartition);
 printf("EssPartitionGetReplCells  sts: %ld\n",sts);

 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile

- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionNewDefFile

設定された入力パラメータに基づいて、新しい共有パーティションの定義ファイルを作成して開きます。

構文

```
ESS_FUNC_M EssPartitionNewDefFile (
hCtx, pszFileName, pHostDatabase, piFileHandle, ppDdbCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API ネットワーク・コンテキスト。
pszFileName	ESS_STR_T	作成するファイルの名前(フル・パス)。
pHostDatabase	161 ページの「ESS_PART_CONNECT_INFO_T」	ホスト・データベースを識別します。
piFileHandle	ESS_PINT_T	作成されたファイルへのハンドル。
ppDdbCtx	161 ページの「ESS_PART_T」	初期化された配布コンテキスト。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M ESS_PartitionNewDefFile(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = 0;
 ESS_INT_T   iFileHandle;
 ESS_STR_T   pszFileName;
 ESS_PART_T   *pDdbCtx;
 ESS_STR_T   hostname, appname, dbname;
 ESS_PART_CONNECT_INFO_T  HostDatabase;
 pszFileName = "C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\app1\\trg1\
\trg1.ddb";
 hostname = "local";
 appname = "app1";
 dbname = "dbname";
 HostDatabase.pszHostName  = hostname;
 HostDatabase.pszAppName   = appname;
 HostDatabase.pszDbName    = dbname;

 sts = EssPartitionNewDefFile(hCtx,pszFileName,&HostDatabase,&iFileHandle,&pDdbCtx);
 printf("EssPartitionNewDefFile  sts: %ld\n",sts);
```

```
 if (!sts)
 {

   /* ...
     ... process definition file information
     ...
   */
     sts = EssPartitionWriteDefFile(hCtx,iFileHandle,pDdbCtx);

   printf("\tEssPartitionWriteDefFile  sts: %ld\n",sts);

     sts = EssPartitionCloseDefFile(hCtx,iFileHandle);

   printf("\tEssPartitionCloseDefFile  sts: %ld\n",sts);

     sts = EssPartitionFreeDefCtx(hCtx,pDdbCtx);

   printf("\tEssPartitionFreeDefCtx  sts: %ld\n",sts);

 }
 return (sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionOpenDefFile

既存の共有パーティションの定義ファイルを開きます。

構文

```
ESS_FUNC_M EssPartitionOpenDefFile (
hCtx, pszFileName, piFileHandle, ppDdbCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pszFileName	ESS_STR_T	作成するファイルの名前(完全パス)。
piFileHandle	ESS_PINT_T	作成されたファイルへのハンドル。
ppDdbCtx	161 ページの「ESS_PART_T」	初期化された配布コンテキスト。

備考

この関数は、一連の定義操作の一部として使用します。

1. EssPartitionOpenDefFile を使用して、既存の定義ファイルを開きます。
2. EssPartitionNewDefFile を使用して、新規定義ファイルを作成し開きます。
3. EssPartitionReadDefFile を使用して定義ファイルを読み取ります。または、EssPartitionWriteDefFile を使用して定義ファイルに書き込みます。
4. EssPartitionCloseDefFile で閉じます。
5. EssPartitionFreeDefCtx でメモリーを解放します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M ESS_PartitionOpenDefFile(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = 0;
 ESS_INT_T    iFileHandle;
 ESS_STR_T    pszFileName;
 ESS_PART_T    DdbCtx, *pDdbCtx;
 pszFileName = "C:\\Hyperion\\products\\Essbase\\EssbaseServer\\app\\app1\\trg1\
\trg1.ddb";
 sts = EssPartitionOpenDefFile(hCtx,pszFileName,&iFileHandle,&pDdbCtx);
 printf("EssPartitionOpenDefFile  sts: %ld\n",sts);

 if (!sts)
 {
    sts = EssPartitionReadDefFile(hCtx,iFileHandle,&DdbCtx);
    printf("\tEssPartitionReadDefFile  sts: %ld\n",sts);

   /* ...
    ... process definition file information
    ...
   */
    sts = EssPartitionCloseDefFile(hCtx,iFileHandle);
```

```
   printf("\tEssPartitionCloseDefFile  sts: %ld\n",sts);

     sts = EssPartitionFreeDefCtx(hCtx,pDdbCtx);

   printf("\tEssPartitionFreeDefCtx  sts: %ld\n",sts);

  }
  return (sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionPurgeOtlChangeFile

TimeStamp パラメータで指定した時刻より前に行われた変更を削除します。

構文

```
ESS_FUNC_M EssPartitionPurgeOtlChangeFile (
hCtx, pPartition, TimeStamp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pPartition	162 ページの「ESS_PART_DEFINED_T」	パーティション定義。

パラメータ	データ型	説明
TimeStamp	ESS_TIME_T	この時刻より前のすべての変更レコードを削除します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

例

```
    ESS_FUNC_M  ESS_PartitionPurgeOtlChangeFile(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts;
 ESS_STR_T     hostname, appname, dbname;
 ESS_USHORT_T    usType,usdir;
 ESS_PART_DEFINED_T Partition;
 memset(&Partition, 0, sizeof(ESS_PART_DEFINED_T));

 hostname = "local";
 appname = "App1";
 dbname = "Src1";
 usType = ESS_PARTITION_OP_LINKED;
 usdir = ESS_PARTITION_DATA_TARGET;
 Partition.usType = usType;
 Partition.usDirection = usdir;
 Partition.HostDatabase.pszHostName = hostname;
 Partition.HostDatabase.pszAppName = appname;
 Partition.HostDatabase.pszDbName = dbname;
 sts = EssPartitionPurgeOtlChangeFile (hCtx, &Partition, 0);
 printf("EssPartitionPurgeOtlChangeFile  sts:  %ld\n",sts);
 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime

- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionPutReplCells

複製パーティションで識別されているすべてのデータ・セルを、選択したソース・データベースからターゲット・データベースに複製します。

構文

```
ESS_FUNC_M EssPartitionPutReplCells (
hCtx, pReplicatePartition
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pReplicatePartition	ESS_PPART_REPL_T	パーティション情報。

備考

このルーチンは、削除後ファイルが空になれば削除します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M  Ess_PartitionPutReplCells(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M  sts;
 ESS_PART_REPL_T ReplicatePartition;
 ESS_PART_CONNECT_INFO_T    HostDatabase;

 memset(&ReplicatePartition, 0, sizeof(ESS_PART_REPL_T));
 memset(&HostDatabase, 0, sizeof(ESS_PART_CONNECT_INFO_T));

 ReplicatePartition.pHostDatabase = &HostDatabase;

 ReplicatePartition.lPartitionCount = -1;
 ReplicatePartition.bUpdatedOnly = ESS_FALSE;

 sts = EssPartitionPutReplCells(hCtx, &ReplicatePartition);
 printf("EssPartitionPutReplCells  sts: %ld\n",sts);

 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionReadDefFile

パーティション定義ファイルをメモリーに読み取ります。

構文

```
ESS_FUNC_M EssPartitionReadDefFile (
hCtx, iFileHandle, pDdbCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
iFileHandle	ESS_INT_T	パーティション定義ファイルへのハンドル。
pDdbCtx	161 ページの「ESS_PART_T」	入力される分散データベース・コンテキスト。

備考

この関数は、一連の定義操作の一部として使用します。

1. EssPartitionOpenDefFile を使用して、既存の定義ファイルを開きます。
2. EssPartitionNewDefFile を使用して、新規定義ファイルを作成し開きます。
3. この関数を使用して定義ファイルを読み取ります。または、EssPartitionWriteDefFile を使用して定義ファイルに書き込みます。
4. EssPartitionCloseDefFile で閉じます。
5. EssPartitionFreeDefCtx でメモリーを解放します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

使用例については、EssPartitionOpenDefFile を参照してください。

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionReadOtlChangeFile

ターゲット・データベース上の変更ファイル(*.CHG)から、変更をメモリーに読み込みます。この関数は、EssPartitionGetOtlChanges を呼び出した後に、EssPartitionApplyOtlChangeRecs を使用して対話的に使用するように設計されています。この関数にはフィルタを使用できます。

構文

```
ESS_FUNC_M EssPartitionReadOtlChangeFile (
hCtx, pSelectMetaRecords, pMetaChangeRead
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pSelectMetaRecords	181 ページの「ESS_PARTOTL_SELECT_CHG_T」	読み取るレコードの選択条件。

パラメータ	データ型	説明
pMetaChangeRead	ESS_PREAD_T	ファイルから読み取られたメタ変更レコードへのポインタ。

備考

このルーチンでは pMetaChangeRead に時刻が戻されます。ターゲット・データベースのタイム・スタンプを更新するために EssPartitionApplyOtlChangeRecs に渡すタイム・スタンプと同じです。また、適用されたレコードをパージするために使用する EssPartitionPurgeOtlChangeFile のタイム・スタンプとも同じです。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M ESS_PartitionReadOtlChangeFile(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_FUNC_M     sts;
 ESS_STR_T     chgfilename;
 ESS_TIME_T     time;
 ESS_PARTOTL_CHANGE_API_T  OtlChg;
 ESS_ULONG_T    uldimfilter=0,ulmbrfilter=0,ulmbrattrfilter=0;
 ESS_PARTOTL_SELECT_CHG_T SelectMetaRecords;
 ESS_PARTOTL_READ_T     MetaChangeRead;

 memset(&OtlChg, 0, sizeof(ESS_PARTOTL_CHANGE_API_T));
 memset(&SelectMetaRecords, 0, sizeof(ESS_PARTOTL_SELECT_CHG_T));
 memset(&MetaChangeRead, 0, sizeof(ESS_PARTOTL_READ_T));

 chgfilename = "d:\\essbase5\\app\\app1\\trg1\\ess00001.chg";
 time = 0;

 uldimfilter   = ESS_DIMCHG_ALL;
 ulmbrfilter   = ESS_PARTITION_OTLMBR_ALL;
 ulmbrattrfilter = ESS_PARTITION_OTLMBRATTR_ALL;
 SelectMetaRecords.pszFileName      = chgfilename;
 SelectMetaRecords.QueryFilter.TimeStamp  = time;
 SelectMetaRecords.QueryFilter.ulDimFilter = uldimfilter;
 SelectMetaRecords.QueryFilter.ulMbrFilter = ulmbrfilter;
 SelectMetaRecords.QueryFilter.ulMbrAttrFilter = ulmbrattrfilter;

 MetaChangeRead.pOtlChg = &OtlChg;
 sts = EssPartitionReadOtlChangeFile (hCtx, &SelectMetaRecords, &MetaChangeRead);
 printf("EssPartitionReadOtlChangeFile  sts:  %ld\n",sts);
 sts = EssPartitionFreeOtlChanges(hCtx);
 printf("\tEssPartitionFreeOtlChanges  sts:  %ld\n",sts);

 return(sts);
```

```
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionReplaceDefFile

新しい共有パーティション・ファイルが送信済で、このデータベースの既存のファイルが置換されることをサーバーに通知します。

構文

```
ESS_FUNC_M EssPartitionReplaceDefFile (
hCtx
);
```

パラメータ　データ型　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M  Ess_PartitionReplaceDefFile(ESS_HCTX_T hCtx)
{
```

```
 ESS_FUNC_M   sts = ESS_STS_NOERR;

 sts = EssPartitionReplaceDefFile(hCtx);
 printf("EssPartitionReplaceDefFile  sts: %ld\n",sts);
 return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionResetOtlChangeTime

ソース・パーティションから「最終変更」時刻を取得し、宛先パーティションの「最終メタ変更」時刻として割り当てます。

構文

```
     ESS_FUNC_M EssPartitionResetOtlChangeTime
(
     hCtx, pSourcePartition, pDestinationPartition
     );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pSourcePartition	162 ページの「ESS_PART_DEFINED_T」	新規時刻のパーティション。
pDestinationPartition	162 ページの「ESS_PART_DEFINED_T」	時刻がリセットされるパーティション。

備考

- ソース・パーティションはタイム・スタンプを提供するパーティションを参照し、ターゲット・パーティションはタイム・スタンプを受け取るパーティションを参照します。
- ソース・パーティションは、データ・ソース・パーティションまたはアウトライン・ソース・パーティションである必要はありません。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャ権限が必要です。

例

```
    ESS_FUNC_M ESS_PartitionResetOtlChangeTime(ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M          sts;
 ESS_PART_DEFINED_T SourcePartition, TargetPartition;
 memset(&SourcePartition, 0, sizeof(ESS_PART_DEFINED_T));
 memset(&TargetPartition, 0, sizeof(ESS_PART_DEFINED_T));

 SourcePartition.HostDatabase.pszHostName = "local";
 SourcePartition.HostDatabase.pszAppName = "App1";
 SourcePartition.HostDatabase.pszDbName = "Src1";
 SourcePartition.usType = ESS_PARTITION_OP_LINKED;
 SourcePartition.usDirection = ESS_PARTITION_DATA_SOURCE;

 TargetPartition.HostDatabase.pszHostName = "local";
 TargetPartition.HostDatabase.pszAppName = "App1";
 TargetPartition.HostDatabase.pszDbName = "Trg1";
 TargetPartition.usType = ESS_PARTITION_OP_LINKED;
 TargetPartition.usDirection = ESS_PARTITION_DATA_TARGET;


 sts = EssPartitionResetOtlChangeTime (hCtx, &SourcePartition, &TargetPartition);
 printf("EssPartitionResetOtlChangeTime  sts: %ld\n",sts);
 return(sts);

}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges

- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionValidateDefinition

リモート・サーバー上の pRemoteDDBFilename にあるパーティション定義に対して、これに対応する(ESS_PPARTSLCT_VALIDATE_T で指定した)ローカル・パーティション定義を確認します。

構文

```
      ESS_FUNC_M EssPartitionValidateDefinition (
      hCtx, pSelectVerify,
pulInvalidComponent, ppInvalidComponent, pRemoteDDBFileName
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pSelectVerify	182 ページの「ESS_PARTSLCT_VALIDATE_T」	確認するパーティションに関する説明。
pulInvalidComponent	ESS_PULONG_T	検証の実行結果におけるエラーおよび警告の数。
ppInvalidComponent	164 ページの「ESS_PARTDEF_INVALID_T」	検証の実行結果におけるエラーおよび警告のリスト。
pRemoteDDBFileName	ESS_STR_T	リモート・サーバーのパーティション定義ファイル名。

備考

- pulInvalidComponent が 0 でないとき、無効なコンポーネントを解放するには、関数 EssFree を呼び出します。
- リモート・パーティション定義ファイルは、ローカルまたはリモート・ホスト上に保存できます。パーティション定義ファイルがローカルの場合、pRemoteDDBFileName には拡張子付きのファイル名を含むフル・パスを指定する必要があります。パーティション定義ファイルがリモートの場合は、pRemoteDDBFileName には拡張子を除くファイル名を指定する必要があります(拡張子は.DDB であると想定)。

- システム上にあるパーティション定義ファイルをサーバーによって検索する場合は、次の規則に従います:
  - pSelectVerify->pszFileName = DbName の場合のサーバーの検索対象は、DbName.**DDN** になります。
  - pSelectVerify->pszFileName != DbName の場合のサーバーの検索対象は、pSelectVerify->pszFileName.**DDB** になります。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_STS_T  ESS_PartitionValidateDefinition(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T    sts = 0;
 ESS_PARTSLCT_VALIDATE_T  SelectVerify;
 ESS_PARTDEF_INVALID_T   *pInvalidComponent;
 ESS_ULONG_T        ulInvalidComponentCount = 0;
 ESS_STR_T         pRemoteDDBFileName = "src";

 /* assume, logged into target database  */

 memset(&SelectVerify, 0, sizeof(ESS_PARTSLCT_VALIDATE_T));
 SelectVerify.usLoc             = ESS_FILE_SERVER;
 SelectVerify.pszFileName          = "trg";
 SelectVerify.Part.usType          = ESS_PARTITION_OP_REPLICATED;
 SelectVerify.Part.usDirection       = ESS_PARTITION_DATA_TARGET;
 SelectVerify.Part.HostDatabase.pszHostName = "Local"
 SelectVerify.Part.HostDatabase.pszAppName  = "PartSrc";
 SelectVerify.Part.HostDatabase.pszDbName   = "Src";

sts = EssPartitionValidateDefinition (hCtx, &SelectVerify,
   &ulInvalidComponentCount, &pInvalidComponent, pRemoteDDBFileName);

 if (ulInvalidComponentCount > 0)
   printf("Validation resulted in warnings and errors.\n");
 else
   printf ("Partition is valid.\n");

 if (pInvalidComponent)
   EssFree(hInst, pInvalidComponent);

return(sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs

- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateLocal
- EssPartitionWriteDefFile

EssPartitionValidateLocal

ESS_HCTX_T で指定したデータベースに関連付けられているすべてのパーティション定義を確認します。

構文

```
ESS_FUNC_M EssPartitionValidateLocal (
hCtx, pusValidateResult
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pusValidateResult	ESS_PUSHORT_T	パーティション検証の結果。

備考

pusValidateResult は、次のいずれかの値になります:

- ESS_DDB_VERIFY_ERROR (検証でエラーが発生した場合)
- ESS_DDB_VERIFY_FAIL (検証が失敗した場合)
- ESS_DDB_VERIFY_NOERR (すべてのパーティションが有効である場合)
- ESS_DDB_VERIFY_WARNING (検証で警告が生成された場合)

戻り値

関数が正常に終了した場合はゼロが戻され、関数が正常に終了しなかった場合はエラー・コードが戻されます。関数がパーティション定義のないデータベース上で実行された場合は、ゼロが戻されます。

アクセス

この関数を呼び出すには、データベース・デザイナのアクセス権限が必要です。

例

```
    ESS_FUNC_M ESS_PartitionValidateLocal(ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_USHORT_T usValidateRes = (ESS_USHORT_T)ESS_DDB_VERIFY_NOERR;

 sts = EssPartitionValidateLocal(hCtx, &usValidateRes);

 if (!sts)
 {
  switch (usValidateRes)
  {
  case ESS_DDB_VERIFY_WARNING:
   printf("Validation resulted in warning(s) - see server log for details\n");
   break;
  case ESS_DDB_VERIFY_ERROR:
   printf("Validation resulted in error(s) - see server log for details\n");
   break;
  default:
   printf("\nPartition(s) validated\n");
   break;
  }
 }
 else
 {
  printf("Call to EssPartitionValidateLocal() failed.\n");
 }
 return (sts);
}
```

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges
- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile

- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionWriteDefFile

EssPartitionWriteDefFile

共有パーティション定義ファイルの現在のメモリー・バージョンをディスクに書き込みます。

構文

```
ESS_FUNC_M EssPartitionWriteDefFile (
hCtx, iFileHandle, TpDdbCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
iFileHandle	ESS_INT_T	共有パーティション定義ファイルへのハンドル。
pDdbCtx	161 ページの「ESS_PART_T」	書き出す値。

備考

この関数は、一連の定義操作の一部として使用します。

1. EssPartitionOpenDefFile を使用して、既存の定義ファイルを開きます。
2. EssPartitionNewDefFile を使用して、新規定義ファイルを作成し開きます。
3. この関数を使用して定義ファイルに書き込むか、EssPartitionReadDefFile を使用して定義ファイルを読み取ります。
4. EssPartitionCloseDefFile で閉じます。
5. EssPartitionFreeDefCtx でメモリーを解放します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

例

使用例は、EssPartitionNewDefFile を参照してください。

関連トピック

- 115 ページの「パーティションの定数および構造体の定義(C)」
- EssPartitionApplyOtlChangeFile
- EssPartitionApplyOtlChangeRecs
- EssPartitionCloseDefFile
- EssPartitionFreeDefCtx
- EssPartitionFreeOtlChanges

- EssPartitionGetAreaCellCount
- EssPartitionGetList
- EssPartitionGetOtlChanges
- EssPartitionGetReplCells
- EssPartitionNewDefFile
- EssPartitionOpenDefFile
- EssPartitionPurgeOtlChangeFile
- EssPartitionPutReplCells
- EssPartitionReadDefFile
- EssPartitionReadOtlChangeFile
- EssPartitionReplaceDefFile
- EssPartitionResetOtlChangeTime
- EssPartitionValidateDefinition
- EssPartitionValidateLocal

EssPerformAllocationASO

集約ストレージ・データベースに対する割当てを実行または確認します。

構文

```
ESS_FUNC_M EssPerformAllocationASO (
hCtx, verifyOnly, errorList, allocStruct
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
verifyOnly	ESS_BOOL_T	割当てを実行せずに割当てパラメータを確認することを示すフラグ。これを ESS_TRUE に設定すると、割当てパラメータの確認のみが行われます。ESS_FALSE の場合は、割当てが確認され、実行されます。
errorList	183 ページの「ESS_PERF_ALLOC_ERROR_T」**	API 関数によって割り当てられ、戻されるエラー構造体のリンク・リストへのポインタ。これにより、クライアントは警告およびエラー・メッセージの詳細を保持します。この引数は 0 にできません。リンク・リストはクライアントによって解放される必要があります。
allocStruct	184 ページの「ESS_PERF_ALLOC_T」*	割当てパラメータを指定する構造体。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    void HandleErrors(ESS_HINST_T hInst, ESS_PERF_ALLOC_ERROR_T **pErrorList)
{
 if (pErrorList)
 {
```

```
   ESS_PERF_ALLOC_ERROR_T *errorList = *pErrorList;
   ESS_PERF_ALLOC_ERROR_T *nextError;

   while (errorList)
   {
    printf("Error number %ld occurred\n", errorList->messageNumber);
    if (errorList->argument != ESS_PERF_ALLOC_ARG_NA)
     printf("  in argument %d\n", errorList->argument);
    if (errorList->lineNumber)
     printf("  on line %ld\n", errorList->lineNumber);
    if (errorList->token[0] != '\0')
     printf("  on token %s\n", errorList->token);

    nextError = errorList->nextError;
    ESS_STS_T sts = EssFree (hInst, errorList);
    printf("\nEssFree sts for errorList %ld\n",sts);
    errorList = nextError;
   }

   *pErrorList = NULL;
  }
}

void ESS_GLAllocation()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_BOOL_T verifyOnly;
 ESS_PERF_ALLOC_ERROR_T *errorList = ESS_NULL;
 ESS_PERF_ALLOC_T *allocStruct;

 sts = EssAlloc (hInst, sizeof(ESS_PERF_ALLOC_T), (ESS_PPVOID_T)&allocStruct);
 printf("EssAlloc sts for allocStruct: %ld\n", sts);

 verifyOnly = ESS_FALSE;
 errorList = ESS_NULL;
 allocStruct->pov = "[[Account]]@[1100]]].Children";
 allocStruct->amount = "100";
 allocStruct->amountContext = "";
 allocStruct->amountTimeSpan = "";
 allocStruct->target = "([Allocated], [041509GR PL2], [11], [[All Department
Values]].[000]]], [0000], [Base], [USD], [Total])";
 allocStruct->targetTimeSpan = "{[Feb-08]}";
 allocStruct->targetTimeSpanOption = ESS_ASO_ALLOCATION_TIMESPAN_DIVIDEAMT;
 allocStruct->offset = "([Mar-08], [041509GR PL2], [11], [[All Department Values]].
[000]]], [0000], [Base], [USD], [Total], [291], [Allocated])";
 allocStruct->debitMember = "[Beginning Balance Dr]";
 allocStruct->creditMember = "[Beginning Balance Cr]";
 allocStruct->range = "DESCENDANTS([Accessories], [Product].Levels(0))";
 allocStruct->excludedRange = "";
 allocStruct->basis = "([041509GR PL2], [11], [[All Department Values]].[000]]],
[0000], [Base], [USD], [Total], [Beginning Balance Cr], [4140], [Actual])";
 allocStruct->basisTimeSpan = "{[Feb-08]}";
 allocStruct->basisTimeSpanOption = ESS_ASO_ALLOCATION_TIMESPAN_COMBINEBASIS;
 allocStruct->allocationMethod = ESS_ASO_ALLOCATION_METHOD_SHARE;
 allocStruct->spreadSkipOption = 0;
 allocStruct->zeroAmountOption = ESS_ASO_ALLOCATION_ZEROAMT_DEFAULT;
 allocStruct->zeroBasisOption = ESS_ASO_ALLOCATION_ZEROBASIS_NEXTAMT;
```

```
 allocStruct->negativeBasisOption = ESS_ASO_ALLOCATION_NEGBASIS_DEFAULT;
 allocStruct->roundMethod = ESS_ASO_ALLOCATION_ROUND_NONE;
 allocStruct->roundDigits = "";
 allocStruct->roundToLocation = "";
 allocStruct->groupID = 0;
 allocStruct->ruleID = 0;

 sts = EssPerformAllocationAso(hCtx, verifyOnly, &errorList, allocStruct);
 printf("EssPerformAllocationAso sts: %ld\n",sts);

 HandleErrors(hInst, &errorList);
 if(allocStruct)
 {
  sts = EssFree (hInst, allocStruct);
       printf("EssFree sts for allocStruct %ld\n",sts);
      }
}
```

EssPerformCustomCalcASO

集約ストレージ・データベース上でカスタム計算を実行または検証します。

構文

```
ESS_FUNC_M EssPerformCustomCalcASO (
hCtx, verifyOnly, errorList, calcStruct
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
verifyOnly	ESS_BOOL_T	計算を実行せずに検証するかどうかを示すフラグ。ESS_TRUE に設定されている場合、計算は検証のみ行われます。ESS_FALSE に設定されている場合、計算は検証および実行されます。
errorList	183 ページの「ESS_PERF_ALLOC_ERROR_T」**	カスタム計算に関するエラー情報を含む API によって移入されて戻されるエラー構造体のリンク・リストへのポインタ。この引数は 0 にできません。リンク・リストはクライアントによって解放される必要があります。
calcStruct	187 ページの「ESS_PERF_CUSTCALC_T」*	クライアントが割り当てたカスタム計算構造体およびパラメータへのポインタ。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    void HandleErrors(ESS_HINST_T hInst, ESS_PERF_ALLOC_ERROR_T **pErrorList)
{
 if (pErrorList)
 {
```

```
   ESS_PERF_ALLOC_ERROR_T *errorList = *pErrorList;
   ESS_PERF_ALLOC_ERROR_T *nextError;

   while (errorList)
   {
    printf("Error number %ld occurred\n", errorList->messageNumber);
    if (errorList->argument != ESS_PERF_ALLOC_ARG_NA)
     printf("  in argument %d\n", errorList->argument);
    if (errorList->lineNumber)
     printf("  on line %ld\n", errorList->lineNumber);
    if (errorList->token[0] != '\0')
     printf("  on token %s\n", errorList->token);

    nextError = errorList->nextError;
    ESS_STS_T sts = EssFree (hInst, errorList);
    printf("\nEssFree sts for errorList %ld\n",sts);
    errorList = nextError;
   }

   *pErrorList = NULL;
  }
}

void ESS_GLCustomCalc()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_BOOL_T verifyOnly;
 ESS_PERF_ALLOC_ERROR_T *errorList = ESS_NULL;
 ESS_PERF_CUSTCALC_T *calcStruct;

 sts = EssAlloc (hInst, sizeof(ESS_PERF_CUSTCALC_T), (ESS_PPVOID_T)&calcStruct);
 printf("EssAlloc sts for calcStruct: %ld\n", sts);

 sts = EssAlloc (hInst, sizeof(ESS_PERF_CUSTCALC_T), (ESS_PPVOID_T)&calcStruct);
 printf("EssAlloc sts: %ld\n", sts);

 verifyOnly = ESS_FALSE;
 errorList = ESS_NULL;
 calcStruct->pov = "{[1120], [1130]}";
 calcStruct->script = "[Jan-96] := ([Feb-08], [041509GR PL2], [00], [[All Department
Values]].[000]]], [0000], [[All Product Values]].[000]]], [Actual], [Beginning
Balance Dr], [BASE], [USD], [Total]);";
 calcStruct->target = "([041509GR PL2], [00], [[All Department Values]].[000]]],
[0000], [[All Product Values]].[000]]], [Actual], [BASE], [USD], [Total])";
 calcStruct->debitMember = "[Beginning Balance Dr]";
 calcStruct->creditMember = "[Beginning Balance Cr]";
 calcStruct->offset = "";
 calcStruct->sourceRegion = "{([Feb-08], [041509GR PL2], [00], [[All Department
Values]].[000]]], [0000], [[All Product Values]].[000]]], [Actual], [Beginning
Balance Dr], [BASE], [USD], [Total])}";
 calcStruct->groupID = 0;
 calcStruct->ruleID = 0;

 sts = EssPerformCustomCalcAso(hCtx, verifyOnly, &errorList, calcStruct);
 printf("EssPerformCustomCalcAso sts: %ld\n",sts);

 HandleErrors(hInst, &errorList);
```

```
 if(calcStruct)
 {
  sts = EssFree (hInst, calcStruct);
  printf("EssFree sts for allocStruct %ld\n",sts);
 }
}
```

EssPutObject

ローカル・ファイルからサーバーまたはクライアントのオブジェクト・システムにオブジェクトをコピーし、オプションでロック解除します。

構文

```
ESS_FUNC_M EssPutObject (
hCtx, ObjType, AppName, DbName, ObjName, LocalName, Unlock
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	追加するオブジェクト名。
LocalName	ESS_STR_T	クライアント上のローカル・ソース・ファイルのフル・パス名。
Unlock	ESS_BOOL_T	オブジェクトのロック解除を制御するフラグ。TRUE の場合、サーバー・オブジェクトのロックが解除され、他のユーザーによる更新が許可されます。

備考

- サーバー上に存在しているオブジェクトを配置するには、呼出し元によって事前にロックされている必要があります。オブジェクトがサーバー上になければ、新規作成されます。

戻り値

正常終了の場合は、オブジェクトが LocalName で指定したローカル・ファイルからサーバーにコピーされます。

アクセス

この関数を使用するには、呼出し元はオブジェクトが含まれている指定したアプリケーション、データベースのいずれか、またはその両方に対して、適切なレベルのアクセス権(オブジェクト・タイプにより異なる)を持っている必要がありま

す。オブジェクトのロックを解除するには(unlock フラグが TRUE)、呼出し元はオブジェクトを含む指定のアプリケーションまたはデータベースに対するアプリケーション、またはデータベースのデザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)が必要です。

例

```
    ESS_FUNC_M
ESS_PutObject (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     ObjName;
 ESS_OBJTYPE_T   ObjType;
 ESS_STR_T     LocalName;
 ESS_BOOL_T     UnLock;

 AppName  = "Sample";
 DbName   = "Basic";
 ObjName  = "Basic1";
 ObjType  = ESS_OBJTYPE_OUTLINE;
 LocalName = "C:\\Hyperion\\products\\Essbase\\EssbaseClient\\Test.otl";
 UnLock   = ESS_TRUE;

 sts = EssPutObject (hCtx, ObjType, AppName,
    DbName, ObjName, LocalName, UnLock);
 return (sts);
}
```

関連トピック

- EssGetObject
- EssLockObject
- EssUnlockObject

EssQueryDatabaseMembers

レポートスタイルのクエリーを実行して、選択したデータベース・メンバーの情報をリストします。

構文

```
ESS_FUNC_M EssQueryDatabaseMembers (
hCtx, mbrQuery
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
mbrQuery	ESS_STR_T	メンバー・クエリー文字列。クエリー文字列は、レポート指定に類似のコマンドです。有効なクエリー文字列は、注意を参照してください。

備考

- NULL 文字列が戻されるまで EssGetString を呼び出して、このクエリーから戻されたメンバー情報を読み取る必要があります。
- この関数では、属性メンバー・ロング名がサポートされます。
- レポート指定の詳細は、『Oracle Essbase テクニカル・リファレンス』を参照してください。
- この関数は、ブール値 bSpanRelPart が EssSetSpanRelationalPartition によって設定されている場合に、リレーショナル・パーティションとして保管されているメンバーについての情報を戻すことができます。この関数は、メンバー名、別名(リレーショナル・メンバーのメンバー名と同じ)および次元/世代番号に基づいた、メンバーのソートをサポートします。他のオプションについては、リレーショナル・メンバーは同一に扱われ、メンバーのリストの最下位に表示されます。

  メンバー選択文字列にはリレーショナル・ストアでサポートされないものもあります。この関数は、次のメンバー選択文字列についてのリレーショナル情報を戻すことができます:

  - ALLINSAMEDIM
  - DIMTOP
  - CHILDRENOF
  - DESCENDANTSOF
  - PARENTOF
  - ANCESTORSOF
  - ALLSIBILINGSOF
- メンバー・クエリー文字列は、選択文字列、オプションのソート・コマンドおよび後続のオプション出力コマンドから構成されます。次の形式を使用します:

```
mbrQuery ==: <selectionstring> [<sortcommand> [<outputcommand>] ]
```

- メンバー<selectionstring>に有効な値は次のとおりです:

```
<CHILDRENOF  -- returns ICHILDRENOF
<ALLINSAMEDIM
<DIMTOP
<OFSAMEGENERATION
<ONSAMELEVELAS
<ANCESTORSOF -- returns IANCESTORSOF
<PARENTOF
<DESCENDANTSOF -- returns IDESCENDANTSOF
<ALLSIBLINGSOF
<LSIBLINGOF
```

- <sortcommand>に有効な値は次のとおりです:

```
<SORTASCENDING
```

```
<SORTDESCENDING
<SORTNONE
<SORTMBRNAMES
<SORTALTNAMES
<SORTMBRNUMBERS
<SORTDIMNUMBERS
<SORTLEVELNUMBERS
<SORTGENERATION
```

- <outputcommand>の形式は次のとおりです:

```
<outputcommand> ==: Item [separator] | FORMAT {<item> <separator> }
```

  - メンバー情報に関する 1 アイテムのリストを取得するには、次の出力コマンドを使用します:

```
  <outputcommand> ==: <MBRNAMES |
<ALTNAMES |
<MBRNUMBERS |
<DIMNUMBERS |
<LEVELNUMBERS |
<GENERATIONS |
<CALCSTRINGS |
<UCALCS |
<TABSEPARATED |
<SPACESEPARATED |
<COMMASEPARATED |
<NEWLINESEPARATED |
<ATTRIBUTES
```

  - メンバーの複数の情報アイテムのリストを取得するには、フォーマット指定句を使用します。リストしたいアイテム、順序、区切り文字を指定します。フォーマット指定句の構文は次のとおりです:

```
<FORMAT <item> [<separator>] {<item> [<separator>]}
```

    <item>に有効な値は次のとおりです:

```
MBRNAMES
ALTNAMES
MBRNUMBERS
DIMNUMBERS
LEVELNUMBERS
GENERATIONS
CALCSTRINGS
UCALCS
ATTRIBUTES
```

    ATTRIBUTES は、属性の数と、それに続く属性名のタブ区切りリストとしてリストされます。

<separator>に有効な値は次のとおりです:

```
            TABSEPARATED
SPACESEPARATED
COMMASEPARATED
NEWLINESEPARATED
```

区切り文字を指定しない場合のデフォルトは TABSEPARATED です。

- 以下にスクリプト例を示します:

```
        login "local" "user1" "password" "" ""
select "attr" "attr"
GetMembers "<NEWLINESEPARATED
<FORMAT {
MBRNAMES    SPACESEPARATED ALTNAMES   TABSEPARATED
MBRNUMBERS   SPACESEPARATED DIMNUMBERS  TABSEPARATED
LEVELNUMBERS  SPACESEPARATED GENERATIONS TABSEPARATED
CALCSTRINGS   SPACESEPARATED UCALCS    TABSEPARATED
DIMTYPES    SPACESEPARATED STATUSES   TABSEPARATED
ATTRIBUTES
}
<DESCENDANTS Product "
```

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_STS_T
ESS_GetMembers (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
       )
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_STR_T   mString = NULL;

 sts = EssQueryDatabaseMembers (hCtx,
    "<ALLINSAMEDIM Year");

 if (!sts)
  sts = EssGetString (hCtx, &mString);

 while ((!sts) && (mString != NULL))
 {
  printf ("%s\r\n", mString);
  EssFree (hInst, mString);
```

```
  sts = EssGetString (hCtx, &mString);
 }

 return(sts);
}
```

関連トピック

- EssCheckMemberName
- EssGetMemberInfo
- EssSetActive

EssRealloc

定義されたメモリー割当ての仕組みを使用して、以前に割り当てられたメモリー・ブロックを異なるサイズに再割当てします。

構文

```
ESS_FUNC_M EssRealloc (
hInstance, Size, ppBlock
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。
Size	ESS_SIZE_T	再割当てするメモリー・ブロックの新規サイズ。
ppBlock	ESS_PPVOID_T	以前に割り当てられたメモリー・ブロックへのポインタのアドレス。このポインタは、再割当てされたメモリー・ブロックを参照するように更新されます。

備考

- この関数は、EssInit 関数に渡されたユーザー指定のメモリー管理関数を使用して、以前に割り当てられたメモリーを再割当てします。この種の関数が提供されていない場合は、デフォルトのメモリー再割当て関数(プラットフォームによって異なる)が使用されます。
- この呼出しでは、EssAlloc 関数で割り当てられたメモリーのみ再割当てする必要があります。また、この関数で再割当てされたメモリーは、必ず EssFree 関数で解放する必要があります。
- 一般に、サイズが 0 のブロックの再割当ての結果はプラットフォームやコンパイラに依存するので、このような再割当てはお薦めしません。

戻り値

正常終了の場合、再割当てされたメモリー・ブロックへのポインタが ppBlock に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_VOID_T
ESS_Realloc (ESS_HINST_T hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_SIZE_T   Size;
 ESS_PVOID_T  pBlock = NULL;

 /* Allocate memory */
 Size = 10;
 sts = EssAlloc(hInst, Size, &pBlock);
 if(sts)
  printf("Cannot allocate memory\r\n");

 /* Reallocate memory */
 Size = 20;
 if(!sts)
 {
  sts = EssRealloc(hInst, Size, &pBlock);
  if(sts)
   printf("Cannot reallocate memory\r\n");
 }

 if(pBlock)
  EssFree(hInst, pBlock);
}
```

関連トピック

- EssAlloc
- EssFree
- EssInit

EssRemoveAlias

アクティブなデータベースから別名テーブルを完全に削除します。

構文

```
ESS_FUNC_M  EssRemoveAlias (
hCtx, AliasName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AliasName	ESS_STR_T	削除する別名テーブルの名前。

備考

- この関数は、アクティブな別名テーブルまたはデフォルトの別名テーブルを削除できません。

- EssListConnections を呼び出して、別名テーブルを削除しようとしているデータベースが他のユーザーによって使用されていないことを確認します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESS_PRIV_READ)を持っていて、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    ESS_FUNC_M
ESS_RemoveAlias (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AliasName;
 AliasName = "NewAlias";
 sts = EssRemoveAlias(hCtx, AliasName);
 if(!sts)
  printf("The %s is removed.\r\n",AliasName);

 return (sts);
}
```

関連トピック

- EssClearAliases
- EssListAliases
- EssSetActive

EssRemoveLocks

現在ユーザーがロックしているデータベースのデータ・ブロックのロックをすべて解除します。

構文

```
ESS_FUNC_M EssRemoveLocks (
hCtx, AppName, DbName, LoginId
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。

パラメータ	データ型	説明
LoginId	ESS_LOGINID_T	解除するロックを保持しているユーザー・ログインの ID。

備考

- 必要とされる LoginId は、EssListLocks 関数によって戻されるユーザー・ロック情報構造体から取得できます。
- LoginId で指定したユーザーが現在ログインしている場合、この関数はそのユーザーの接続を終了します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RemoveLocks (ESS_HCTX_T    hCtx,
        ESS_HINST_T    hInst
        )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;

 ESS_USHORT_T   Count;
 ESS_PLOCKINFO_T plockinfo = NULL;
 ESS_PLOCKINFO_T plinfo;
 ESS_USHORT_T   ind;
 ESS_SHORT_T    Item;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";
for (ind = 0; ind < Count; ind++)
   {
    plinfo = plockinfo + ind;
printf ("%-2d  %-15s  %-12ld  %-5d    %ld\r\n",
    ind, plinfo->UserName, plinfo->LoginId,
    plinfo->nLocks, plinfo->Time);
   }
   printf ("\r\n");
   /**********************************
   * Chooser Lock List Item to Remove *
   **********************************/
   Item = 1;
 }
 else
 {
printf ("\r\nExclusive Lock List on %s:%s is empty\r\n\r\n",
AppName, DbName);
   goto exit;
 }
```

```
 if (!sts)
 {
  if ((Item >= 0) && (Item < Count))
  {
   plinfo = plockinfo + Item;
   sts = EssRemoveLocks (hCtx, AppName,
      DbName, plinfo->LoginId);
  }
 }
exit:
 if (plockinfo)
  EssFree (hInst, plockinfo);
return (sts);
}
```

関連トピック

- EssListLocks

EssReplayTransactions

指定したトランザクションを実行(再実行)します。

- デフォルトでは、この関数は最後に復元されたバックアップ時刻または最後に再実行した要求時刻のいずれか新しい時刻以降のすべてを再実行します。
- この関数は復元後の要求を再実行しません。これは、復元コマンドの使用方法として、トランザクションを再実行して、新しいトランザクションを開く方法が推奨されているからです。
- pSeqIds オプションを使用して、再実行を強制できます。

構文

```
ESS_FUNC_M EssReplayTransactions(hCtx, AppName,
                                 DbName, ReplayDat, pSeqIds);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	ログイン・コンテキスト。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
ReplayDat	ESS_TRANSACTION_REPLAY_INP_T	入力パラメータの再実行。
pSeqIds	ESS_PSEQID_T	入力タイプがシーケンス ID の場合は、シーケンス ID 範囲の配列

戻り値

- 0 - 正常終了の場合
  - pSeqIds にはシーケンス ID の範囲が含まれています
- エラー番号 - 失敗した場合

アクセス

呼出し元は、データベースへの管理者アクセス権を持っている必要があります。

例

```
    void ListAndReplayTransactions()
{
    ESS_FUNC_M                 sts = ESS_STS_NOERR;
    ESS_USHORT_T                 TimeSrc;
    ESS_TIME32_T                 timestamp = 0;
    ESS_USHORT_T                 listOption;
    ESS_STR_T                     FileName = ESS_NULL;
    ESS_ULONG_T                 Count = 0;
    ESS_PTRANSACTION_ENTRY_T    pResults;
    ESS_CHAR_T                 listTime[ESS_TIMESIZE];
    ESS_TRANSACTION_REPLAY_INP_T    ReplayDat;
    ESS_PSEQID_T                 pSeqIds = ESS_NULL;
    ESS_OBJDEF_T                 Data;
    ESS_STR_T                     Script;
    ESS_SHORT_T                isAbortOnError;
    ESS_PMBRERR_T            pMbrErr = NULL;
    ESS_PROCSTATE_T        pState;

    /* Load data from server */
    Data.hCtx    = hCtx;
    Data.AppName  = AppName;
    Data.DbName   = DbName;
    Data.ObjType  = ESS_OBJTYPE_TEXT;
    Data.FileName  = "Calcdat";
    isAbortOnError = ESS_TRUE;
    sts = EssImport (hCtx, ESS_NULL, &Data,
                               &pMbrErr, NULL, isAbortOnError);
    printf("EssImport sts: %ld\r\n",sts);

    /* List and replay with a specified time */
    TimeSrc = 1;
    strcpy(listTime, "09/18/2007:00:00:00");
                          /* mm/dd/yyyy:hh:mm:ss */
    timestamp = adtGenericGetTime(listTime);
    listOption = ESS_LIST_TRANSACTIONS_TOCLIENT;
    sts = EssListTransactions(hCtx, TimeSrc,
                                              timestamp, listOption,
                                              FileName, &Count, &pResults);
        /* This function converts listTime to the number of
            seconds since January 1, 1970. */
    printf("EssListTransactions sts: %ld\r\n",sts);
    if (Count && pResults)
        PrintTransactionLog(Count, pResults);

    memset(&ReplayDat, 0, sizeof
                               (ESS_TRANSACTION_REPLAY_INP_T));
    ReplayDat.InpType = ESS_REPLAY_BASED_GIVENTIME;
    ReplayDat.value.InpTime = timestamp;
    sts =
    EssReplayTransactions (hCtx, AppName, DbName,
                                                    ReplayDat, pSeqIds);
```

```
    printf("EssReplayTransactions sts: %ld\r\n",sts);
    printf("\n\n");

    /* Run a calc*/
    Script = "CALC ALL;";
    sts = EssCalc(hCtx, ESS_TRUE, Script);
    printf("EssCalc sts: %ld\r\n",sts);
    if (!sts)
    {
        sts = EssGetProcessState (hCtx, &pState);
        while (!sts && (pState.State != ESS_STATE_DONE))
            sts = EssGetProcessState (hCtx, &pState);
    }

    /* List and replay with last replay time */
    TimeSrc = 2;
    timestamp = 0;
    sts = EssListTransactions(hCtx, TimeSrc,
                                          timestamp, listOption,
                                          FileName, &Count, &pResults);
        /* This function converts listTime to the number of
            seconds since January 1, 1970. */
    printf("EssListTransactions sts: %ld\r\n",sts);
    if (Count && pResults)
        PrintTransactionLog(Count, pResults);
    memset(&ReplayDat, 0, sizeof
                           (ESS_TRANSACTION_REPLAY_INP_T));
    ReplayDat.InpType = ESS_REPLAY_BASED_LASTREPLAYTIME;
    sts =
    EssReplayTransactions (hCtx, AppName,
                         DbName, ReplayDat, pSeqIds);


    printf("EssReplayTransactions sts: %ld\r\n",sts);

    if(pSeqIds)
        EssFree(hInst, pSeqIds);
    if(pResults)
        EssFree(hInst, pResults);
    if(pMbrErr)
        EssFree(hInst, pMbrErr);
}
```

Using SeqIds

シーケンス ID 配列を使用して再実行する場合、シーケンス ID の範囲を指定します。

- num_seq_id_range に範囲カウントを入力します。
- num_seq_id_range の後に ESS_SEQID_T の配列、タイプ・データ構造体を入力します。配列内の要素数は num_seq_id_range と一致している必要があります。

- seq_id_upper_start および seq_id_upper_end フィールドは予約済で、0 を入力しておく必要があります。
- seq_id_start および seq_id_end フィールドには、範囲の開始と終了の値を入力する必要があります。
- シーケンス ID が 1 つしかない場合は、開始と終了の値としてその ID を指定します。

例 1: 1-5、8-10、12-16 の範囲を 6、7、および 11 をスキップして再実行する場合:

```
    num_seq_id_range = 3
seqid_array[0].seq_id_start = 1
seqid_array[0].seq_id_end = 5
seqid_array[0].seq_id_start_upper = 0
seqid_array[0].seq_id_end_upper = 0
seqid_array[1].seq_id_start = 8
seqid_array[1].seq_id_end = 10
seqid_array[1].seq_id_start_upper = 0
seqid_array[1].seq_id_end_upper = 0
seqid_array[2].seq_id_start = 12
seqid_array[2].seq_id_end = 16
seqid_array[2].seq_id_start_upper = 0
seqid_array[2].seq_id_end_upper = 0
```

例 2: 3-7 の範囲のみを再実行する場合は、num_seq_id_range = 1:

```
    seqid_array[0].seq_id_start = 3
seqid_array[0].seq_id_end = 7
seqid_array[0].seq_id_start_upper = 0
seqid_array[0].seq_id_end_upper = 0
```

例 3: トランザクション ID の 5 を再実行する場合:

```
    num_seq_id_range = 1
seqid_array[0].seq_id_start = 5
seqid_array[0].seq_id_end = 5
seqid_array[0].seq_id_start_upper = 0
seqid_array[0].seq_id_end_upper = 0
```

関連トピック

- 197 ページの「ESS_SEQID_T」
- 144 ページの「ESS_DISKVOLUME_REPLACE_T」
- 198 ページの「ESS_TRANSACTION_ENTRY_T」
- 199 ページの「ESS_TRANSACTION_REPLAY_INP_T」
- 200 ページの「ESS_TRANSACTION_REQSPECIFIC_T」
- EssListTransactions

EssRenameApplication

クライアント上またはサーバー上で、既存のアプリケーションの名前を変更します。アプリケーションがサーバー上で実行されている場合、最初に停止されます。

構文

```
ESS_FUNC_M  EssRenameApplication (
hCtx, OldName, NewName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
OldName	ESS_STR_T	名前を変更する既存のアプリケーション名。
NewName	ESS_STR_T	アプリケーションの新しい名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

クライアント・アプリケーションの名前を変更すると、ローカル・アプリケーションのディレクトリ名も変更されます。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESS_PRIV_APPCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RenameApp (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M  sts = ESS_STS_NOERR;
 ESS_STR_T  OldName;
 ESS_STR_T  NewName;

 OldName = "Sample";
 NewName = "Sample2";
 sts = EssRenameApplication(hCtx, OldName,
    NewName);

 return (sts);
}
```

関連トピック

- EssRenameDatabase
- EssRenameObject

EssRenameDatabase

クライアントまたはサーバー上で、アプリケーション内の既存のデータベースの名前を変更します。データベースがサーバー上で実行されている場合、最初に停止されます。

構文

```
ESS_FUNC_M  EssRenameDatabase (
hCtx, AppName, OldName, NewName
);
```

パラメータ　データ型　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
OldName	ESS_STR_T	名前を変更する既存のデータベース名。
NewName	ESS_STR_T	データベースの新しい名前。1903 ページの「データベース名の制限」を参照してください。

備考

クライアント・データベースの名前を変更すると、ローカル・データベースのディレクトリ名も変更されます。

戻り値

なし。

アクセス

サーバー・データベースの場合は、呼出し元がデータベースの作成/削除/編集権限(ESS_PRIV_DBCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RenameDatabase (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M  sts;
 ESS_STR_T  AppName;
 ESS_STR_T  OldName;
 ESS_STR_T  NewName;

 AppName = "Sample";
 OldName = "Basic";
 NewName = "Basic2";

 sts = EssRenameDatabase(hCtx, AppName, OldName,
    NewName);

 return(sts);
}
```

関連トピック

- EssRenameApplication
- EssRenameObject

EssRenameFilter

既存のフィルタの名前を変更します。

構文

```
ESS_FUNC_M EssRenameFilter (
hCtx, AppName, DbName, OldName, NewName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
OldName	ESS_STR_T	名前を変更する既存のフィルタの古い名前。
NewName	ESS_STR_T	フィルタの新しい名前。1904 ページの「フィルタ名の制限」を参照してください。

備考

古いフィルタ名が存在し、新しいフィルタ名が存在していないことが必要です。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RenameFilter (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    OldName;
 ESS_STR_T    NewName;

 AppName = "Sample";
 DbName  = "Basic";
 OldName = "Test";
 NewName = "NewTest";
```

```
  sts = EssRenameFilter(hCtx, AppName, DbName,
     OldName, NewName);
  return (sts);
}
```

関連トピック

- EssCopyFilter
- EssDeleteFilter
- EssListFilters

EssRenameGroup

既存のグループの名前を変更します。

構文

```
ESS_FUNC_M  EssRenameGroup (
hCtx, OldName, NewName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
OldName	ESS_STR_T	名前を変更する既存のグループの古い名前。
NewName	ESS_STR_T	グループの新しい名前。1904 ページの「グループ名の制限」を参照してください。

備考

- 指定された新規グループ名が存在していることが必要です。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RenameGroup (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   OldName;
 ESS_STR_T   NewName;
 OldName = "PowerUsers";
 NewName = "PowerGroup";

 sts = EssRenameGroup (hCtx, OldName, NewName);
```

```
 return (sts);
}
```

関連トピック

- EssCreateGroup
- EssDeleteGroup
- EssListGroups

EssRenameObject

サーバーまたはクライアント・オブジェクト・システム上の既存のオブジェクトの名前を変更します。

構文

```
      ESS_FUNC_M EssRenameObject (
      hCtx, ObjType, AppName, DbName,
OldName, NewName
      );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
OldName	ESS_STR_T	名前を変更するオブジェクトの古い名前。
NewName	ESS_STR_T	オブジェクトの新しい名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- オブジェクト名を変更するには、そのオブジェクトがロックされていないことと、新しいオブジェクトが存在していないことが必要です。
- アウトライン・オブジェクトおよび LRO オブジェクトの名前は、変更できません。
- 関連付けられているアウトラインを含めてデータベース名を変更するには、EssRenameDatabase を使用します。
- 異なるアプリケーションやデータベースではオブジェクトの名前を変更できません。別のアプリケーションやデータベースにオブジェクトをコピーするには、EssCopyObject を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、オブジェクトが含まれる指定されたアプリケーションまたはデータベースに対してアプリケーション・デザイン権限またはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_STS_T
ESS_RenameObject (ESS_HCTX_T hCtx)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     OldName;
 ESS_STR_T     NewName;
 ESS_OBJTYPE_T   ObjType;

 AppName   = "Sample";
 DbName    = "Basic";
 OldName   = "Test";
 NewName   = "NewTest";
 ObjType   = ESS_OBJTYPE_TEXT;

 sts = EssRenameObject(hCtx, ObjType, AppName,
    DbName, OldName, NewName);

 if(!sts)
  printf("The Object is renamed.\r\n");

 return (sts);
}
```

関連トピック

- EssCopyObject
- EssCreateObject
- EssDeleteObject
- EssListObjects

EssRenameUser

既存のユーザー名を変更します。

構文

```
ESS_FUNC_M EssRenameUser (
hCtx, OldName, NewName
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
OldName	ESS_STR_T	名前を変更する既存のユーザーの古い名前。
NewName	ESS_STR_T	ユーザーの新しい名前。1904 ページの「ユーザー名の制限」を参照してください。

備考

指定された新規ユーザー名が存在していない必要があります。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_RenameUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   OldName;
 ESS_STR_T   NewName;

 OldName = "Jim Smith";
 NewName = "Tom Smith";

 sts = EssRenameUser (hCtx, OldName, NewName);

 return (sts);
}
```

関連トピック

- EssCreateUser
- EssDeleteUser
- EssListUsers

EssReport

レポート指定を単一文字列としてアクティブなデータベースに送信します。この関数は、EssBeginReport を呼び出し、次に EssSendString を呼び出し、最後に EssEndReport を呼び出すのと同じです。レポート・データは出力することも、レポート指定の確認のみ行いエラーがあれば戻させることもできます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。

構文

```
ESS_FUNC_M  EssReport (
hCtx, Output, Lock, RptSpec
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Output	ESS_BOOL_T	データの出力を制御します。TRUE の場合は、指定したレポートに従ってサーバーから出力されます。FALSE の場合は、データは出力されません。
Lock	ESS_BOOL_T	ブロックのロックを制御します。TRUE の場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSE の場合は、ブロックのロックは行われません。
RptSpec	ESS_STR_T	単一の文字列としてのレポート指定(64KB 未満である必要があります)。

備考

- レポート指定の文字列の長さは、64KB 未満である必要があります。
- この関数によってデータが出力される場合(Output フラグが TRUE)、戻されるデータは、NULL が戻されるまで EssGetString を呼び出して読み取る必要があります。
- この関数によってブロックがロックされる場合(Lock フラグが TRUE)、呼出し元はロックされたブロックのロック解除を行う必要があります(たとえば、Unlock フラグを TRUE に設定して EssUpdate を呼び出します)。
- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。呼出し元がアクセス権を持っていないすべてのメンバーは、不明として戻されます。

例

```
    ESS_FUNC_M
ESS_ReportLine (ESS_HCTX_T    hCtx,
       ESS_HINST_T  hInst
       )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T   rString;

 sts = EssReport (hCtx, ESS_TRUE, ESS_FALSE,
    "<Desc Year !");
```

```
 /******************
  * Get the report *
  ******************/

 if (!sts)
   sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
  EssFree (hInst, rString);
   sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return (sts);
}
```

関連トピック

- EssBeginReport
- EssEndReport
- EssGetString
- EssReportFile
- EssUpdate

EssReportFile

ファイルからアクティブなデータベースへレポート指定を送信します。レポート・データを出力できます。または、レポート指定の確認のみも可能です。エラーがあれば戻されます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。

構文

```
ESS_FUNC_M  EssReportFile (
hDestCtx, hSrcCtx, AppName, DbName, FileName, Output, Lock
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	レポート・ファイルの場所に対する API コンテキスト・ハンドル。レポート・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。レポート・ファイルがクライアント(ローカル)にある場合、ローカルのコンテキストは EssCreateLocalContext で作成する必要があります。
AppName	ESS_STR_T	レポート・ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	レポート・ファイルの場所のデータベース名。

パラメータ	データ型	説明
FileName	ESS_STR_T	レポート指定ファイル名。拡張子は.repであることがわかっているため、ファイル拡張子を指定する必要はありません。
Output	ESS_BOOL_T	データの出力を制御します。TRUEの場合は、指定したレポートに従ってサーバーから出力されます。FALSEの場合は、データは出力されません。
Lock	ESS_BOOL_T	ブロックのロックを制御します。TRUEの場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSEの場合は、ブロックのロックは行われません。

備考

- この関数によってデータが出力される場合(Outputフラグが TRUE)は、戻されるデータは EssGetString を呼び出して読み取ることができます。
- この関数によってブロックがロックされる場合(Lockフラグが TRUE)、呼出し元はロックされたブロックのロック解除を行う必要があります(たとえば、Unlockフラグを TRUE に設定して EssUpdate を呼び出します)。
- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ReportFile (ESS_HCTX_T  hCtx,
        ESS_HINST_T hInst
      )
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_STR_T   rString;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;

 hSrcCtx = hCtx;
 AppName = "Sample";
 DbName  = "Basic";
 FileName = "Test";

 sts = EssReportFile (hCtx, hSrcCtx, AppName,
    DbName, FileName, ESS_TRUE, ESS_FALSE);
 /* Get the report */
 if (!sts)
  sts = EssGetString (hCtx, &rString);
```

```
 while ((!sts) && (rString != NULL))
 {
  printf ("%s", rString);
  EssFree (hInst, rString);
  sts = EssGetString (hCtx,&rString);
 }
 return(sts);
}
```

関連トピック

- EssBeginReport
- EssGetString
- EssReport
- EssUpdateFile

EssReRegisterApplication

1 つまたはすべての Essbase アプリケーションを Shared Services アプリケーションとして再確立します。

構文

```
ESS_FUNC_M EssReRegisterApplication (
hCtx
,
AppName
,
AllApps
);
```

パラメータ データ型 説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	再登録するアプリケーション名。
AllApps	ESS_BOOL_T	ESS_TRUE の場合は、すべてのアプリケーションが再登録されます。それ以外の場合は、名前付きのアプリケーションのみが再登録されます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者、アプリケーション・マネージャ、またはデータベース・マネージャである必要があります。呼出し元に十分な権限がないアプリケーションの場合は、警告メッセージが表示され、アプリケーションの再登録がスキップされます。

例

```
ESS_FUNC_M ESS_SS_ReRegisterApplication(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
```

```
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_BOOL_T    allApps;
 ESS_STR_T     appName = ESS_NULL;

 sts = EssAlloc(hInst, sizeof(ESS_APPNAME_T), &appName);
 if(sts)
  return (sts);
 memset(appName, 0, sizeof(ESS_APPNAME_T));
 strcpy( appName, "Sample");

 /* Do you want All applications re-registered?
  * Enter ESS_TRUE for Yes
  *    ESS_FALSE for No
  **/
 allApps = ESS_FALSE; /* Re-registering only 1 application */

 sts = EssReRegisterApplication(hCtx, appName, allApps);

 if (sts)
  printf("Failed to Re-register Application %s.\n", appName);

 if (appName)
  EssFree(hInst, appName);

 return (sts);
}
```

拡張された付録 B も参照してください

EssResetDatabase

アクティブなデータベース内のロード済データをすべて消去し、アウトラインを空にリセットします。

構文

```
ESS_FUNC_M EssResetDatabase (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

- この関数を使用して削除されたデータおよびアウトライン・リセットは復元できません。注意して使用してください!
- この関数呼出しは非同期です。この呼出しを行った後、データベースのリセット操作の完了を示すステータスが戻されるまで、EssGetProcessState を呼び出す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースへの書込み権限(ESS_PRIV_WRITE)を持ち、EssSetActive を使用してそのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
     ESS_FUNC_M
 ESS_ResetDb (ESS_HCTX_T   hCtx)
 {
  ESS_FUNC_M    sts = ESS_STS_NOERR;
  ESS_PROCSTATE_T pState;
  sts = EssResetDatabase(hCtx);

  if (!sts)
  {
   sts = EssGetProcessState (hCtx, &pState);
   while(!sts && (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
  }
  return (sts);
 }
```

関連トピック

- EssResetPerfStats
- EssDumpPerfStats

EssResetPerfStats

パフォーマンス統計テーブルの値をゼロにリセットします。

構文

パラメータ	データ型	説明
hCtx;	ESS_HCTX_T	API コンテキスト・ハンドル。
persistence;	ESS_ULONG_T	リセットされるテーブルのセットのパーシスタンスを示す次の値のいずれかになります: ● 0: 短期間のテーブルのみをリセットします ● 1: 短期間および中期間のテーブルをリセットします ● 2: 短期間、中期間および長期間のテーブルをリセットします ● 3: パフォーマンス統計情報の収集を使用不可にします ● 4: パフォーマンス統計情報の収集を使用可能にします

パラメータ	データ型	説明
scope;	ESS_ULONG_T	リセットされるテーブルのセットの範囲を示す次の値のいずれかになります: ● 1: スレッドベースのテーブルのみをリセットします ● 2: データベースベースのテーブルのみをリセットします ● 4: サーバーベースのテーブルのみをリセットします ● 7: すべてのテーブルをリセットします

備考

- 統計収集を使用可能(persistence の値が 4)または使用不可(persistence の値が 3)にしても、統計はリセットされません。
- パフォーマンス統計テーブルの詳細は、Oracle Essbase テクニカル・リファレンスの MaxL のパフォーマンス統計に関するトピックを参照してください。

戻り値

成功の場合、0 が戻されます。

アクセス

この関数を使用するには、スーパーバイザ・アクセス権が必要です。

例

```
    /* This function resets all short term tables */

ESS_STS_T ESSResetPerfStats(ESS_HCTX_T *context)
{
 ESS_STS_T  sts     = ESS_STS_NOERR;
 ESS_ULONG_T persistence = 0;
 ESS_ULONG_T scope    = 7;

 sts = EssResetPerfStats(context, persistence, scope);

 return sts;
}
```

関連トピック

- EssDumpPerfStats
- EssGetStatBufSize

EssResetUser

ユーザーのセキュリティ構造体を最初の状態にリセットします。

構文

```
      ESS_FUNC_M EssResetUser() (
     hCtx
```

```
,
UserName
);
```

パラメータ　データ型　　説明

hCtx;　　ESS_HCTX_T API コンテキスト・ハンドル。

UserName;　ESS_STR_T　ユーザー名。

備考

次のユーザー・セキュリティ・パラメータは、初期状態にリセットされます:

- LockedOut
- PwdChgNow
- Failcount
- LastLogin
- LastPwdChg
- Expiration

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ResetUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T  UserName = "William";

 sts = EssResetUser (hCtx, UserName);
 return (sts);
}
```

関連トピック

- EssInit
- EssLogin
- EssLogout

EssRestore

使用されなくなりました。

この関数は、Essbase の以前のリリースとの互換性のためにのみ保持されています。現行の Essbase アーカイブについては、EssArchiveBegin および EssArchiveEnd を参照してください。この関数は、エラー・メッセージ ESS_STS_OBSOLETE を戻します。

関連項目

EssArchiveBegin

EssArchiveEnd

EssArchive

EssSetActive

EssRestoreDatabase

指定したバックアップ・アーカイブ・ファイルからデータベースを復元します。

構文

```
ESS_FUNC_M EssRestoreDatabase (hCtx, AppName, DbName, BackupFileName,
bForceDiffName, Count, ReplaceVol);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	ログイン・コンテキスト。
AppName	ESS_STR_T	アプリケーション名。 **注：** データベース・レベルでのみ動作します。AppName パラメータはアプリケーションを指定し、存在するデータベースにアクセスできるようにします。
DbName	ESS_STR_T	データベース名。
BackupFileName	ESS_STR_T	アーカイブ・データの読込み元となるバックアップ・ファイルのフル・パス。次の例のようにフル・パスを指定します: `c:\hyperion\Test.arc`

パラメータ	データ型	説明
bForceDiffName	ESS_BOOL_T	復元には別のアプリケーション名とデータベース名を使用します。 ● ESS_TRUE – 復元に別のアプリケーション名とデータベース名のいずれか、またはその両方を強制的に使用します。 ESS_TRUE を使用して、アプリケーション名とデータベース名がバックアップのものと同じ場合は、ESS_FALSE と同じ結果になります ● ESS_FALSE – バックアップ・ファイルに保管されているアプリケーション名とデータベース名が使用されます。バックアップ・ファイル内の名前が復元先のものと同じであることが確認されます。
Count	ESS_USHORT_T	オプション。 復元するディスク・ボリューム置換構造体の数。
ReplaceVol	ESS_PDISKVOLUME_REPLACE_T	オプション。 ディスク・ボリューム置換入力構造体。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

アクセス

呼出し元は、データベースへの管理者アクセス権を持っている必要があります。

例

```
    void RestoreDB()
{
    ESS_FUNC_M        sts = ESS_STS_NOERR;
    ESS_STR_T            AppName = "Backup";
    ESS_STR_T            DbName = "Basic";
    ESS_STR_T            BackupFileName =
                   "F:\\testArea\\ArchiveAndRestore\\TempBackup.arc";
    ESS_STR_T            optionsFileName = "";
    ESS_BOOL_T        bOverWrite;
    ESS_BOOL_T        bForceDiffName;
    ESS_USHORT_T        count;
    ESS_PDISKVOLUME_REPLACE_T    replaceVol;

    printf("\nArchive DB:\n");
    bOverWrite = ESS_TRUE;
    sts = EssArchiveDatabase(hCtx, AppName, DbName,
                                          BackupFileName, optionsFileName,
                                          bOverWrite)
    printf("EssArchiveDatabase sts: %ld\r\n",sts);

    sts = EssUnloadApplication(hCtx, AppName);
    printf("\nEssUnloadApplication sts: %ld\r\n",sts);
```

```
    printf("\nCase with no volume replacement:\n");
    bForceDiffName = ESS_FALSE;
    count = 0;
    replaceVol = ESS_NULL;
    sts =
    EssRestoreDatabase (hCtx, AppName, DbName,
                                                BackupFileName, bForceDiffName,
                                                count, replaceVol);


    printf("EssRestoreDatabase sts: %ld\r\n",sts);

    printf("\nCase with a replacement volume (index and page files to a different
volume):\n");
    bForceDiffName = ESS_FALSE;
    count = 1;
    if (count)
    {
        sts = EssAlloc(hInst, count * sizeof(ESS_DISKVOLUME_REPLACE_T),
                                    (ESS_PPVOID_T)&replaceVol);
        memset(replaceVol, 0, count * sizeof(ESS_DISKVOLUME_REPLACE_T));
    }
    strcpy(replaceVol->szPartition_Src, "C");
    strcpy(replaceVol->szPartition_Dest, "F");

    sts = EssUnloadApplication(hCtx, AppName);
    printf("\nEssUnloadApplication sts: %ld\r\n",sts);

    sts =
    EssRestoreDatabase (hCtx, AppName, DbName,
                                                BackupFileName, bForceDiffName,
                                                count, replaceVol);


    printf("EssRestoreDatabase sts: %ld\r\n",sts);

    if (replaceVol)
         EssFree(hInst, replaceVol);
}
```

関連トピック

- EssArchiveDatabase

EssSendString

アクティブなデータベースにデータの文字列を送信します。この関数は、EssBeginReport、EssBeginUpdate または EssBeginCalc を呼び出した後に呼び出す必要があります。

構文

```
    ESS_FUNC_M  EssSendString (
```

```
      hCtx, String
      );
```

パラメータ データ型 説明

hCtx ESS_HCTX_T API コンテキスト・ハンドル。

String ESS_STR_T データ文字列。

備考

- レポート開始、更新または計算の関数が正常に実行される前に関数を呼び出すと、エラーが発生します。
- この関数を EssBeginUpdate とともに使用するときは、更新文字列の終わりに復帰または改行の文字を付ける必要があります。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
      ESS_HINST_T  hInst
     )
{
 ESS_FUNC_M   sts   = ESS_STS_NOERR;
 ESS_STR_T   rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");

 if (!sts)
  sts = EssEndReport (hCtx);
 /**************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

Unicode 対応 Essbase アプリケーションとの通信に C のメイン API を使用する Unicode クライアントは、この関数を使用してテキスト・ストリーム内の UTF-8 でエンコードされたバイト・オーダー・マーク(BOM)を送信する必要があります。例としては、79 ページの「バイト・オーダー・エンコーディングの指定」を参照してください。

関連トピック

- EssBeginCalc
- EssBeginReport
- EssBeginUpdate
- EssGetString

EssSetActive

呼出し元のアクティブなアプリケーションとデータベースを設定します。

構文

```
ESS_FUNC_M  EssSetActive (
hCtx, AppName, DbName, pAccess
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pAccess	ESS_PACCESS_T	選択したデータベースに対するユーザーのアクセス・レベルを受け取る変数のアドレス。このフィールドに使用できる値のリストは、102 ページの「ビットマスク・データ型(C)」についての説明を参照してください。

備考

- アプリケーションおよびデータベースがロードされていない場合、これらはこの関数によってロードされます。
- Windows では、EssAutoLogin 関数を使用して、ユーザーに対するログインの許可およびアクティブなアプリケーションとデータベースの設定を行うこともできます。

戻り値

正常終了の場合は、選択したアプリケーションおよびデータベースに対するユーザーのアクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    ESS_FUNC_M
ESS_SetActive (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M sts = ESS_STS_NOERR;
 ESS_ACCESS_T Access;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;

 AppName = "Sample";
 DbName  = "Basic";

 sts = EssSetActive (hCtx, AppName, DbName,
    &Access);
 return (sts);
}
```

関連トピック

- EssClearActive
- EssGetActive
- EssListApplications
- EssListDatabases
- EssLogin

EssSetAlias

1 人のユーザーについて、アクティブなデータベースにアクティブな別名テーブルを設定します。

構文

```
ESS_FUNC_M EssSetAlias (
hCtx, AliasName
);
```

パラメータ　データ型　説明

hCtx　ESS_HCTX_T API コンテキスト・ハンドル。

AliasName　ESS_STR_T　アクティブに設定する別名テーブルの名前。

戻り値

なし。

例

```
    ESS_FUNC_M
ESS_SetAlias (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
```

```
 ESS_STR_T   AliasName;
 AliasName = "TestAlias";
 sts = EssSetAlias (hCtx, AliasName);

 return (sts);
}
```

関連トピック

- EssGetAlias
- EssListAliases

EssSetApplicationAccess

アプリケーションへのユーザーのアクセス権情報を含むユーザー・アプリケーション・アクセス構造体のリストを設定します。

構文

```
ESS_FUNC_M  EssSetApplicationAccess (
hCtx, Count, pUserApp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Count	ESS_USHORT_T	ユーザー・アプリケーション構造体のカウント。
pUserApp	200 ページの「ESS_USERAPP_T、ESS_GROUPAPP_T」	ユーザー・アプリケーション構造体の配列へのポインタ。

備考

- ユーザー・アプリケーション構造体の Access フィールドを使用して、ユーザーに付与されたアプリケーションへのアクセスを設定します。この呼出しでは MaxAccess フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定されたアプリケーションに対してアプリケーション設計権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetApplicationAccess (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_USHORT_T  Count;
 ESS_USERAPP_T UserApp;
```

```
 strcpy(UserApp.UserName,"Jim Smith");
 strcpy(UserApp.AppName,"Sample");
 UserApp.Access    = ESS_PRIV_APPDESIGN;
 UserApp.MaxAccess  = ESS_PRIV_APPDESIGN;
 sts = EssSetApplicationAccess(hCtx, Count,
   &UserApp);
 return (sts);
}
```

関連トピック

- EssSetApplicationAccessEx
- EssGetApplicationAccess
- EssListUsers
- EssSetDatabaseAccess
- EssSetUser

EssSetApplicationAccessEx

アプリケーションへのユーザーのアクセス権情報を含むユーザー・アプリケーション・アクセス構造体のリストを設定します。EssSetApplicationAccess に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

構文

```
ESS_FUNC_M EssSetApplicationAccessEx (
hCtx
,
Count
,
pUserApp
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
Count	ESS_USHORT_T	ユーザー・アプリケーション構造体のカウント(入力)。
pUserApp	ESS_PUSERAPPEX_T	ユーザー・アプリケーション構造体の配列へのポインタ(入力)。

備考

ユーザー・アプリケーション構造体の Access フィールドを使用して、ユーザーに付与されたアプリケーションへのアクセスを設定します。この呼出しでは MaxAccess フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定されたアプリケーションに対してアプリケーション設計権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
void DisplayUserAppInfo(ESS_PUSERAPPEX_T userApp, ESS_USHORT_T count)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T ind;

 printf ("\n------Application Access List----\n\n");
 for (ind = 0; ind < count; ind++)
 {
  printf("\tUser: %s\n", userApp[ind].UserName);
  printf("\tProvider Name: %s\n", userApp[ind].ProviderName);
  printf("\tConnection Param: %s\n", userApp[ind].connparam);
  printf("\tAppName: %s\n", userApp[ind].AppName);
  switch(userApp[ind].Access)
  {
   case ESS_PRIV_NONE:
    printf("\tAccess: %d - ESS_PRIV_NONE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_READ:
    printf("\tAccess: %d - ESS_PRIV_READ\n", userApp[ind].Access);
    break;
   case ESS_PRIV_WRITE:
    printf("\tAccess: %d - ESS_PRIV_WRITE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_CALC:
    printf("\tAccess: %d - ESS_PRIV_CALC\n", userApp[ind].Access);
    break;
   case ESS_PRIV_METAREAD:
    printf("\tAccess: %d - ESS_PRIV_METAREAD\n", userApp[ind].Access);
    break;
   case ESS_PRIV_DBLOAD:
    printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userApp[ind].Access);
    break;
   case ESS_PRIV_DBMANAGE:
    printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_DBCREATE:
    printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_APPLOAD:
    printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userApp[ind].Access);
    break;
   case ESS_PRIV_APPMANAGE:
    printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_APPCREATE:
    printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userApp[ind].Access);
    break;
   case ESS_PRIV_USERCREATE:
```

```
    printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userApp[ind].Access);
    break;

  case ESS_ACCESS_READ:
    printf("\tAccess: %d - ESS_ACCESS_READ\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_WRITE:
    printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_CALC:
    printf("\tAccess: %d - ESS_ACCESS_CALC\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_METAREAD:
    printf("\tAccess: %d - ESS_ACCESS_METAREAD\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_DBMANAGE:
    printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_DBCREATE:
    printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_APPMANAGE:
    printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_APPCREATE:
    printf("\tAccess: %d - ESS_ACCESS_APPCREATE\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_FILTER:
    printf("\tAccess: %d - ESS_ACCESS_FILTER\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_DBALL:
    printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_APPALL:
    printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userApp[ind].Access);
    break;
  case ESS_ACCESS_ADMIN:
    printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userApp[ind].Access);
    break;
  default:
    printf("\tAccess: Unknown\n");
}

switch(userApp[ind].MaxAccess)
{
  case ESS_PRIV_NONE:
    printf("\tMax Access: %d - ESS_PRIV_NONE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_READ:
    printf("\tMax Access: %d - ESS_PRIV_READ\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_WRITE:
    printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userApp[ind].MaxAccess);
    break;
  case ESS_PRIV_CALC:
    printf("\tMax Access: %d - ESS_PRIV_CALC\n", userApp[ind].MaxAccess);
    break;
```

```
case ESS_PRIV_METAREAD:
 printf("\tMax Access: %d - ESS_PRIV_METAREAD\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_DBLOAD:
 printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userApp[ind].MaxAccess);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userApp[ind].MaxAccess);
 break;

case ESS_ACCESS_READ:
 printf("\tMax Access: %d - ESS_ACCESS_READ\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_WRITE:
 printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_CALC:
 printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_METAREAD:
 printf("\tMax Access: %d - ESS_ACCESS_METAREAD\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_APPCREATE:
 printf("\tMax Access: %d - ESS_ACCESS_APPCREATE\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_FILTER:
 printf("\tMax Access: %d - ESS_ACCESS_FILTER\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_DBALL:
 printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userApp[ind].MaxAccess);
 break;
case ESS_ACCESS_APPALL:
 printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userApp[ind].MaxAccess);
```

```
      break;
     case ESS_ACCESS_ADMIN:
      printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userApp[ind].MaxAccess);
      break;
     default:
      printf("\tMax Access: Unknown\n");
    }

   printf("\n");
  }
}

ESS_FUNC_M ESS_SetApplicationAccessEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T userId;
 ESS_BOOL_T bIsIdentity;
 ESS_USHORT_T type;
 ESS_STR_T AppName;
 ESS_USHORT_T count;
 ESS_USERAPPEX_T userApp[2];
 ESS_PUSERAPPEX_T pUserApp = ESS_NULL;

 memset(&userApp, '\0', sizeof(userApp));

 userId = "IDUser1";
 AppName  = "";
 type = ESS_TYPE_USER;
 bIsIdentity = ESS_FALSE;

 count = 1;
 strcpy(userApp[0].UserName, "IDUser1");
 strcpy(userApp[0].ProviderName, "LDAP");
 strcpy(userApp[0].connparam, "");
 userApp[0].type = ESS_TYPE_USER;
 strcpy(userApp[0].AppName, AppName);
 userApp[0].Access = ESS_PRIV_APPMANAGE;
 userApp[0].MaxAccess = ESS_PRIV_APPMANAGE;
 sts = EssSetApplicationAccessEx(hCtx, count, &userApp);
 printf("EssSetApplicationAccessEx sts: %ld\n", sts);
 if(!sts)
 {
  userId = userApp[0].UserName;
  type = userApp[0].type;
  sts = EssGetApplicationAccessEx(hCtx, userId, bIsIdentity, type, AppName, &count,
&pUserApp);
  printf("EssGetApplicationAccessEx sts: %ld\n", sts);
  if(!sts)
  {
   if(count && pUserApp)
   {
    DisplayUserAppInfo(pUserApp, count);
    sts = EssFree (hInst, pUserApp);
   }
   else
    printf ("\rUser Application list is empty\n\n");
```

```
   }
  }


 return (sts);
}
```

関連トピック

- EssGetApplicationAccessEx
- EssListUsersInfoEx
- EssSetDatabaseAccessEx

EssSetApplicationState

アプリケーションの状態構造体を使用して、ユーザーが構成可能なアプリケーションのパラメータを設定します。

構文

```
ESS_FUNC_M  EssSetApplicationState (
hCtx, AppName, pAppState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
pAppState	123 ページの「ESS_APPSTATE_T」	アプリケーション状態構造体へのポインタ。

備考

- パラメータ値を変更するときは、最初に EssGetApplicationState を呼び出して、変更しないパラメータの正しい値を取得することをお薦めします。
- 次のパラメータは集約ストレージ・データベースに適用されません:LockTimeout および lroSizeLimit。

戻り値

なし。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアプリケーション・デザイナ権限(ESS_PRIV_APPDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetAppState (ESS_HCTX_T  hCtx,
        ESS_HINST_T  hInst
```

```
        )
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_PAPPSTATE_T  AppState;
 ESS_STR_T     AppName;
 AppName = "Sample";

 sts = EssGetApplicationState (hCtx, AppName,
    &AppState);
 if (!sts)
 {
  if (AppState)
  {
   /*****************************
    * Update AppState structure *
    *****************************/
    sts = EssSetApplicationState (hCtx,
       AppName, AppState);
    EssFree (hInst, AppState);
 }
 return (sts);
}
```

関連トピック

- EssGetApplicationState
- EssSetDatabaseState

EssSetCalcList

ユーザーがアクセス可能な計算スクリプト・オブジェクトのリストを設定します。

構文

```
ESS_FUNC_M  EssSetCalcList (
hCtx, UserName, AppName, DbName, AllCalcs, Count, pCalcList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
AllCalcs	ESS_BOOL_T	すべての計算を許可するフラグ。TRUE の場合は、ユーザーはすべての計算スクリプトにアクセスできます。それ以外の場合は、CalcList 引数で指定された計算スクリプトにのみアクセスできます。
Count	ESS_USHORT_T	アクセス可能な計算スクリプト・オブジェクト数のカウント。

パラメータ	データ型	説明
pCalcList	ESS_POBJNAME_T	計算スクリプト・オブジェクト名の配列へのポインタ。

備考

- AllCalcs フラグが TRUE に設定されている場合は、Count および pCalcList 引数は無視されます。
- 計算スクリプト・オブジェクトにアクセスするには、ユーザーは少なくとも該当のデータベースに対して計算アクセス権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetCalcList (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    UserName;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_BOOL_T    AllCalcs;
 ESS_USHORT_T   Count;
 ESS_OBJNAME_T  pCalcList[3];

 UserName = "Newuser";
 AppName  = "Sample";
 DbName   = "Basic";
 AllCalcs = ESS_FALSE ;
 Count = 3;
 strcpy(pCalcList[0],"test1");
 strcpy(pCalcList[1],"test2");
 strcpy(pCalcList[2],"test3");

 sts = EssSetCalcList(hCtx, UserName, AppName,
    DbName, AllCalcs, Count, pCalcList);

 return (sts);
}
```

関連トピック

- EssSetCalcListEx
- EssGetCalcList
- EssListObjects
- EssListUsers

EssSetCalcListEx

指定したユーザーまたはグループにアクセス可能な計算リストを設定します。EssSetCalcList に似ていますが、ユーザー・ディレクトリにホストされているユーザーおよびグループが含まれます。

構文

```
ESS_FUNC_M EssSetCalcListEx (
hCtx
,
UserId
,
bIsIdentity
,
entityType
,
AppName
,
DbName
,
AllCalc
,
count
,
pCalcList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
UserID	ESS_STR_T	入力。計算リストを設定するユーザーまたはグループ。name@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。UserID が名前か ID かを示します。TRUE の場合、UserID は ID です。
entityType	ESS_USHORT_T	入力。UserID に含まれるエンティティのタイプ。次のいずれかになります: ● ESS_TYPE_USER ● ESS_TYPE_GROUP
AppName	ESS_STR_T	アプリケーション名(入力)。
DbName	ESS_STR_T	データベース名(入力)。
AllCalc	ESS_BOOL_T	すべての計算を許可するフラグ(入力)。TRUE の場合、ユーザーまたはグループはすべての計算スクリプトにアクセスできます。それ以外の場合は、CalcList で指定されている計算スクリプトにのみアクセスできます。
Count	ESS_USHORT_T	アクセス可能な計算スクリプト・オブジェクト数のカウント(入力)。
pCalcList	ESS_POBJNAME_T	アクセス可能な計算スクリプト・オブジェクト名の配列へのポインタ(入力)。

備考

- AllCalcs フラグが TRUE に設定されている場合は、Count および pCalcList 引数は無視されます。
- 計算スクリプト・オブジェクトにアクセスするには、ユーザーは少なくとも該当のデータベースに対して計算アクセス権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void GetCalcList(ESS_STR_T userName)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_BOOL_T AllCalcs = ESS_FALSE;
 ESS_USHORT_T Count, ind;
 ESS_POBJNAME_T pCalcList = NULL;

 sts = EssGetCalcList(hCtx, userName, AppName, DbName, &AllCalcs, &Count,
&pCalcList);
 printf("EssGetCalcList sts: %ld\n", sts);
 //sts = EssGetCalcListEx(hCtx, userName, AppName, DbName, &AllCalcs, &Count,
&pCalcList);
 //printf("EssGetCalcListEx sts: %ld\n", sts);

 if(AllCalcs)
  printf("\tThis user has access to all script on %s %s\n", AppName, DbName);
 else
 {
  if(!sts && pCalcList)
  {
   printf("-------- Get Calc List -----------\r\n");
     for (ind = 0; ind < Count; ind ++)
      printf(" %s\n",pCalcList[ind]);

   EssFree(hInst, pCalcList);
  }
 }
}


ESS_FUNC_M ESS_SetCalcListEx (ESS_HCTX_T hCtx)

{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T calcUser;
 ESS_BOOL_T AllCalcs;
 ESS_USHORT_T count;
 ESS_OBJNAME_T CalcList[2];
```

```
 ESS_PUSERINFO_T pUserInfo;
 ESS_BOOL_T bIsIdentity;
 ESS_USHORT_T type;
 ESS_USERDBEX_T userDb[1];
 ESS_PUSERDBEX_T pUserDb = ESS_NULL;

 bIsIdentity = ESS_FALSE;
 type = ESS_TYPE_USER;
 AllCalcs = ESS_FALSE;
 count = 2;
 strcpy(CalcList[0],"calc1");
 strcpy(CalcList[1],"calc2");

 sts = EssSetCalcListEx(hCtx, calcUser, bIsIdentity, type, AppName, DbName,
AllCalcs, count, CalcList);
 printf("EssSetCalcListEx sts: %ld\n", sts);
 if(!sts)
  GetCalcList(calcUser);

 return (sts);
}
```

関連トピック

- EssGetCalcList
- EssListObjects
- EssListUsersInfoEx

EssSetDatabaseAccess

データベースへのユーザー・アクセスに関する情報を含む、ユーザーのデータベース・アクセス構造体のリストを設定します。

構文

```
ESS_FUNC_M EssSetDatabaseAccess (
hCtx, Count, pUserDb
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Count	ESS_USHORT_T	ユーザー・データベース構造体のカウント。
pUserDb	202 ページの「ESS_USERDB_T、ESS_GROUPDB_T」	ユーザー・データベース構造体の配列へのポインタ。

備考

ユーザー・データベース構造体の Access フィールドを使用して、ユーザーに付与されたデータベースへのアクセスを設定します。この呼出しでは MaxAccess と FilterName フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetDatabaseAccess (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_USHORT_T   Count;
 ESS_USERDB_T   UserDb[2];

 Count = 2;
 /* Initialize user database structure for user1 */
 strcpy(UserDb[0].UserName,"Newuser");
 strcpy(UserDb[0].AppName,"Sample");
 strcpy(UserDb[0].DbName,"Basic");
 UserDb[0].Access = ESS_PRIV_WRITE;

 /* Initialize user database structure for user2 */
 strcpy(UserDb[1].UserName,"Newuser2");
 strcpy(UserDb[1].AppName,"Sample");
 strcpy(UserDb[1].DbName,"Basic");
 UserDb[1].Access = ESS_PRIV_READ;
 sts = EssSetDatabaseAccess(hCtx, Count, UserDb);
 return (sts);
}
```

関連トピック

- EssSetDatabaseAccessEx
- EssGetDatabaseAccess
- EssListUsers
- EssSetApplicationAccess
- EssSetUser

EssSetDatabaseAccessEx

データベースへのユーザー・アクセスに関する情報を含む、ユーザーのデータベース・アクセス構造体のリストを設定します。EssSetDatabaseAccess に似ていますが、入力構造体には、ユーザー・ディレクトリおよび一意の ID 属性を含められます。

構文

```
ESS_FUNC_M EssSetDatabaseAccessEx (
hCtx
,
```

```
    Count
    ,
    pUserDb
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
Count	ESS_USHORT_T	ユーザー・データベース構造体のカウント(入力)。
pUserDb	ESS_PUSERDBEX_T	ユーザー・データベース構造体の配列へのポインタ(入力)。

備考

ユーザー・データベース構造体の Access フィールドを使用して、ユーザーに付与されたデータベースへのアクセスを設定します。この呼出しでは MaxAccess と FilterName フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void DisplayUserDbInfo(ESS_PUSERDBEX_T userDb, ESS_USHORT_T count)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T ind;

 printf ("\n------Database Access List----\n\n");
 for (ind = 0; ind < count; ind++)
 {
  printf("\tUser: %s\n", userDb[ind].UserName);
  printf("\tProvider Name: %s\n", userDb[ind].ProviderName);
  printf("\tConnection Param: %s\n", userDb[ind].connparam);
  printf("\tApp Name: %s\n", userDb[ind].AppName);
  printf("\tDb Name: %s\n", userDb[ind].DbName);
  switch(userDb[ind].Access)
  {
   case ESS_PRIV_NONE:
    printf("\tAccess: %d - ESS_PRIV_NONE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_READ:
    printf("\tAccess: %d - ESS_PRIV_READ\n", userDb[ind].Access);
    break;
   case ESS_PRIV_WRITE:
    printf("\tAccess: %d - ESS_PRIV_WRITE\n", userDb[ind].Access);
    break;
   case ESS_PRIV_CALC:
```

```
 printf("\tAccess: %d - ESS_PRIV_CALC\n", userDb[ind].Access);
 break;
case ESS_PRIV_METAREAD:
 printf("\tAccess: %d - ESS_PRIV_METAREAD\n", userDb[ind].Access);
 break;
case ESS_PRIV_DBLOAD:
 printf("\tAccess: %d - ESS_PRIV_DBLOAD\n", userDb[ind].Access);
 break;
case ESS_PRIV_DBMANAGE:
 printf("\tAccess: %d - ESS_PRIV_DBMANAGE\n", userDb[ind].Access);
 break;
case ESS_PRIV_DBCREATE:
 printf("\tAccess: %d - ESS_PRIV_DBCREATE\n", userDb[ind].Access);
 break;
case ESS_PRIV_APPLOAD:
 printf("\tAccess: %d - ESS_PRIV_APPLOAD\n", userDb[ind].Access);
 break;
case ESS_PRIV_APPMANAGE:
 printf("\tAccess: %d - ESS_PRIV_APPMANAGE\n", userDb[ind].Access);
 break;
case ESS_PRIV_APPCREATE:
 printf("\tAccess: %d - ESS_PRIV_APPCREATE\n", userDb[ind].Access);
 break;
case ESS_PRIV_USERCREATE:
 printf("\tAccess: %d - ESS_PRIV_USERCREATE\n", userDb[ind].Access);
 break;

case ESS_ACCESS_READ:
 printf("\tAccess: %d - ESS_ACCESS_READ\n", userDb[ind].Access);
 break;
case ESS_ACCESS_WRITE:
 printf("\tAccess: %d - ESS_ACCESS_WRITE\n", userDb[ind].Access);
 break;
case ESS_ACCESS_CALC:
 printf("\tAccess: %d - ESS_ACCESS_CALC\n", userDb[ind].Access);
 break;
case ESS_ACCESS_METAREAD:
 printf("\tAccess: %d - ESS_ACCESS_METAREAD\n", userDb[ind].Access);
 break;
case ESS_ACCESS_DBMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_DBMANAGE\n", userDb[ind].Access);
 break;
case ESS_ACCESS_DBCREATE:
 printf("\tAccess: %d - ESS_ACCESS_DBCREATE\n", userDb[ind].Access);
 break;
case ESS_ACCESS_APPMANAGE:
 printf("\tAccess: %d - ESS_ACCESS_APPMANAGE\n", userDb[ind].Access);
 break;
case ESS_ACCESS_APPCREATE:
 printf("\tAccess: %d - ESS_ACCESS_APPCREATE\n", userDb[ind].Access);
 break;
case ESS_ACCESS_FILTER:
 printf("\tAccess: %d - ESS_ACCESS_FILTER\n", userDb[ind].Access);
 break;
case ESS_ACCESS_DBALL:
 printf("\tAccess: %d - ESS_ACCESS_DBALL\n", userDb[ind].Access);
 break;
```

```
  case ESS_ACCESS_APPALL:
   printf("\tAccess: %d - ESS_ACCESS_APPALL\n", userDb[ind].Access);
   break;
  case ESS_ACCESS_ADMIN:
   printf("\tAccess: %d - ESS_ACCESS_ADMIN\n", userDb[ind].Access);
   break;
  default:
   printf("\tAccess: Unknown\n");
 }

 switch(userDb[ind].MaxAccess)
 {
  case ESS_PRIV_NONE:
   printf("\tMax Access: %d - ESS_PRIV_NONE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_READ:
   printf("\tMax Access: %d - ESS_PRIV_READ\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_WRITE:
   printf("\tMax Access: %d - ESS_PRIV_WRITE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_CALC:
   printf("\tMax Access: %d - ESS_PRIV_CALC\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_METAREAD:
   printf("\tMax Access: %d - ESS_PRIV_METAREAD\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_DBLOAD:
   printf("\tMax Access: %d - ESS_PRIV_DBLOAD\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_DBMANAGE:
   printf("\tMax Access: %d - ESS_PRIV_DBMANAGE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_DBCREATE:
   printf("\tMax Access: %d - ESS_PRIV_DBCREATE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_APPLOAD:
   printf("\tMax Access: %d - ESS_PRIV_APPLOAD\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_APPMANAGE:
   printf("\tMax Access: %d - ESS_PRIV_APPMANAGE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_APPCREATE:
   printf("\tMax Access: %d - ESS_PRIV_APPCREATE\n", userDb[ind].MaxAccess);
   break;
  case ESS_PRIV_USERCREATE:
   printf("\tMax Access: %d - ESS_PRIV_USERCREATE\n", userDb[ind].MaxAccess);
   break;

  case ESS_ACCESS_READ:
   printf("\tMax Access: %d - ESS_ACCESS_READ\n", userDb[ind].MaxAccess);
   break;
  case ESS_ACCESS_WRITE:
   printf("\tMax Access: %d - ESS_ACCESS_WRITE\n", userDb[ind].MaxAccess);
   break;
  case ESS_ACCESS_CALC:
   printf("\tMax Access: %d - ESS_ACCESS_CALC\n", userDb[ind].MaxAccess);
```

```
      break;
     case ESS_ACCESS_METAREAD:
      printf("\tMax Access: %d - ESS_ACCESS_METAREAD\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_DBMANAGE:
      printf("\tMax Access: %d - ESS_ACCESS_DBMANAGE\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_DBCREATE:
      printf("\tMax Access: %d - ESS_ACCESS_DBCREATE\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_APPMANAGE:
      printf("\tMax Access: %d - ESS_ACCESS_APPMANAGE\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_APPCREATE:
      printf("\tMax Access: %d - ESS_ACCESS_APPCREATE\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_FILTER:
      printf("\tMax Access: %d - ESS_ACCESS_FILTER\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_DBALL:
      printf("\tMax Access: %d - ESS_ACCESS_DBALL\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_APPALL:
      printf("\tMax Access: %d - ESS_ACCESS_APPALL\n", userDb[ind].MaxAccess);
      break;
     case ESS_ACCESS_ADMIN:
      printf("\tMax Access: %d - ESS_ACCESS_ADMIN\n", userDb[ind].MaxAccess);
      break;
     default:
      printf("\tMax Access: Unknown\n");
    }

    printf("\tFilter Name: %s\n", userDb[ind].FilterName);
    printf("\n");
   }
}

ESS_FUNC_M ESS_SetDatabaseAccessEx (ESS_HCTX_T hCtx, ESS_HINST_T hInst)

{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_STR_T userId;
  ESS_BOOL_T bIsIdentity;
  ESS_USHORT_T type;
  ESS_USHORT_T count;
  ESS_USERDBEX_T userDb[2];
  ESS_PUSERDBEX_T pUserDb = ESS_NULL;

  memset(&userDb, '\0', sizeof(userDb));


  count = 1;
  strcpy(userDb[0].UserName, "IDUser1");
  strcpy(userDb[0].ProviderName, "");
  strcpy(userDb[0].connparam, "");
  userDb[0].type = ESS_TYPE_USER;
  strcpy(userDb[0].AppName, AppName);
```

```
 strcpy(userDb[0].DbName, DbName);
 userDb[0].Access = ESS_PRIV_READ;
 userDb[0].MaxAccess = ESS_PRIV_READ;
 sts = EssSetDatabaseAccessEx(hCtx, count, &userDb);
 printf("EssSetDatabaseAccessEx sts: %ld\n\n", sts);
 if(!sts)
 {
  sts = EssGetDatabaseAccessEx(hCtx, userId, bIsIdentity, type, AppName, DbName,
&count, &pUserDb);
  printf("EssGetDatabaseAccessEx sts: %ld\n", sts);
  if(!sts)
  {
   if(count && pUserDb)
   {
    DisplayUserDbInfo(pUserDb, count);
    sts = EssFree (hInst, pUserDb);
   }
   else
    printf ("\rUser Application list is empty\n\n");
  }
 }


 return (sts);
}
```

関連トピック

- EssGetDatabaseAccessEx
- EssListUsersInfoEx
- EssSetApplicationAccessEx

EssSetDatabaseNote

データベースの最新情報に関するメッセージを設定します。このメッセージを使用して、ユーザーがデータベースに接続する前に、データベースに関する有用な情報(データがロードされているかどうか、データが最後に計算されたのはいつかなど)を表示できます。

構文

```
ESS_FUNC_M  EssSetDatabaseNote (
hCtx, AppName, DbName, DbNote
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。

パラメータ	データ型	説明
DbNote	ESS_STR_T	データベース・ノート文字列へのポインタ。

備考

データベース・ノート文字列の長さは、64KB 未満である必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetDatabaseNote (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    DbNote;

 AppName = "Sample";
 DbName  = "Basic";
 DbNote  = "This is a test";

 sts = EssSetDatabaseNote(hCtx, AppName, DbName,
    DbNote);

 return (sts);
}
```

関連トピック

- EssGetDatabaseNote

EssSetDatabaseState

データベースの状態構造体を使用して、ユーザーが構成可能なデータベースのパラメータを設定します。

構文

```
ESS_FUNC_M EssSetDatabaseState (
hCtx, AppName, DbName, pDbState
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
pDbState	134 ページの「ESS_DBSTATE_T」	データベース状態構造体へのポインタ。

備考

- この関数を呼び出す前に EssGetDatabaseState を呼び出して、ESS_DBSTATE_T 構造体を初期化する必要があります。
- この関数は、ユーザーが構成可能なサーバー・データベースのパラメータのみ設定できます。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetDbState (ESS_HCTX_T  hCtx,
       ESS_HINST_T hInst
      )
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_PDBSTATE_T  DbState;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 AppName = "Sample";
 DbName  = "Basic";

 sts = EssGetDatabaseState (hCtx, AppName,
    DbName, &DbState);
 if (!sts)
 {
  if (DbState)
  {
   /****************************
    * Update DbState structure *
    ****************************/
   sts = EssSetDatabaseState (hCtx, AppName,
      DbName, DbState);
   EssFree (hInst, DbState);
  }
 }
 return (sts);
}
```

関連トピック

- EssGetDatabaseState
- EssSetApplicationState

EssSetDefaultCalc

アクティブ・データベースのデフォルトの計算スクリプトを設定します。

構文

```
ESS_FUNC_M  EssSetDefaultCalc (
hCtx, CalcScript
);
```

パラメータ　データ型　　説明

hCtx　　ESS_HCTX_T API コンテキスト・ハンドル。

CalcScript ESS_STR_T　デフォルト計算スクリプト文字列。

備考

計算スクリプト文字列の長さは、64KB 以下である必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetDefaultCalc (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M      sts = ESS_STS_NOERR;
 sts = EssSetDefaultCalc (hCtx, "CALC ALL;");

 return (sts);
}
```

関連トピック

- EssDefaultCalc
- EssGetDefaultCalc
- EssSetActive
- EssSetDefaultCalcFile

EssSetDefaultCalcFile

計算スクリプト・ファイルからアクティブ・データベースに対してデフォルト計算スクリプトを設定します。

構文

```
ESS_FUNC_M EssSetDefaultCalcFile (
hDestCtx, hSrcCtx, AppName, DbName, FileName
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	計算スクリプト・ファイルの場所の API コンテキスト・ハンドル。計算スクリプトは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	計算スクリプト・ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	計算スクリプト・ファイルの場所のデータベース名。
FileName	ESS_STR_T	デフォルト計算スクリプト・ファイルの名前。

備考

- デフォルト計算スクリプトの長さは、64KB 以下である必要があります。
- この関数を呼び出すと、サーバーは計算スクリプト・ファイル内のテキストをコピーします。それ以降、計算スクリプト・ファイルに対して行われた変更は、この関数を再度呼び出して更新するまで、デフォルト計算に反映されません。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESS_PRIV_CALC)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetDefaultCalcFile (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M  sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;
 AppName  = "Sample";
 DbName   = "Basic";
 FileName = "DefTest";
 hSrcCtx = hCtx;
 sts = EssSetDefaultCalcFile (hCtx, hSrcCtx,
```

```
    AppName, DbName, FileName);
  return(sts);
}
```

関連トピック

- EssDefaultCalc
- EssGetDefaultCalc
- EssSetDefaultCalc

EssSetEasLocation

Essbase 管理サーバーの場所を設定または変更します。これは、アプリケーションの作成または移行のとき Shared Services に登録されます。

構文

```
ESS_FUNC_M EssSetEasLocation (
hCtx
,
EasLocation
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
EasLocation	ESS_STR_T	Essbase 管理サーバーが稼働するコンピュータの名前(または IP アドレス)とポート番号。例: Aspen:10080 127.0.0.1:10080

備考

Essbase 管理サーバーの場所を変更した後は、EssReRegisterApplication を使用して既存のアプリケーションを Shared Services に再登録する必要があります。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetEasLocation(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
  ESS_STS_T     sts = ESS_STS_NOERR;
  ESS_STR_T     easLoc = ESS_NULL;
```

```
 /* Eas Location */
 sts = EssAlloc(hInst, sizeof(ESS_PATHLEN), &easLoc);
 if(sts)
  return (sts);
 memset(easLoc, 0, sizeof(ESS_PATHLEN));
 strcpy( easLoc, "localhost:10080");

 sts = EssSetEasLocation(hCtx, easLoc);

 if (sts)
  printf("Failed to set EAS Location.\n");

 if (easLoc)
  EssFree(hInst, easLoc);

 return (sts);
}
```

拡張された付録 B も参照してください

関連トピック

- EssReRegisterApplication

EssSetExtUser

外部認証ユーザーのユーザー情報を設定します。

構文

```
ESS_FUNC_M EssSetExtUser (hCtx, type, UserName, Password, Protocol, ConnParam);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
type	ESS_USHORT_T	ユーザーのタイプ(外部)。
UserName	ESS_STR_T	ユーザーの名前。
Password	ESS_STR_T	ユーザーのパスワード。
Protocol	ESS_STR_T	外部認証プロトコル: Shared Services モードの場合は CSS。
ConnParam	ESS_STR_T	外部認証接続パラメータ。CSS プロトコルの場合は NULL。

アクセス

この関数を使用するには、呼出し元が自分自身のユーザー情報を設定しないかぎり、ログインしているサーバーに対するユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

EssSetFilter

フィルタを作成または置換し、フィルタのコンテンツの設定を開始します。

構文

```
ESS_FUNC_M  EssSetFilter (
hCtx, AppName, DbName, FilterName, Active, Access
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
FilterName	ESS_STR_T	フィルタ名。1904 ページの「フィルタ名の制限」を参照してください。
Active	ESS_BOOL_T	フィルタのアクティブ・フラグ。TRUE の場合はフィルタがアクティブに設定され、TRUE でない場合は非アクティブに設定されます。
Access	ESS_ACCESS_T	デフォルトのフィルタ・アクセス・レベル。

備考

- フィルタが存在しない場合は、この呼出しによって最初にそのフィルタが作成されます。
- この呼出しの後に EssSetFilterRow を続けて呼び出して、フィルタのすべての行を設定する必要があります。
- 既存のフィルタへの上書きを避けるため、EssCreateFilter を使用します。EssCreateFilter は特定のデータベースに一意な名前のフィルタのみを作成しますが、同じデータベース上の同一名の既存のフィルタには上書きしません。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetFilter (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FilterName;
 ESS_BOOL_T    Active;
 ESS_ACCESS_T   Access, AccessAry[3];
 ESS_STR_T    RowString[3];
```

```
 ESS_USHORT_T   ind;

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "NewFilter";
 Active    = ESS_TRUE;

  /***** Set Filter *****/
  sts = EssSetFilter(hCtx, AppName, DbName,
     FilterName, Active, Access);
 if(!sts)
 {
  RowString[0] = "@IDESCENDANTS(Scenario)";
  RowString[1] = "@IDESCENDANTS(Product)";
  RowString[2] = "Qtr1, @IDESCENDANTS(\"Colas\")";

  AccessAry[0] = ESS_ACCESS_READ;
  AccessAry[1] = ESS_ACCESS_NONE;
  AccessAry[2] = ESS_ACCESS_WRITE;
  /***** Set Filter Rows *****/

  for(ind = 0; ind < 3; ind++)
  {
   sts = EssSetFilterRow(hCtx, RowString[ind],
        AccessAry[ind]);
   if(sts)
    printf("Cannot set Filter row %s\r\n",
        RowString[ind]);
  }
   sts = EssSetFilterRow(hCtx,
     "",ESS_ACCESS_NONE);
 }
 return (sts);
}
```

関連トピック

- EssCreateFilter
- EssGetFilter
- EssListFilters
- EssSetFilterRow

EssSetFilterList

フィルタに割り当てられたグループまたはユーザーのリストを設定します。count パラメータはフィルタに割り当てられたグループまたはユーザーの数を制御します。count がゼロの場合、グループまたはユーザーすべてがリストから削除されます。

構文

```
ESS_FUNC_M  EssSetFilterList (
hCtx, AppName, DbName, FilterName, Count, pUserList
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。
FilterName	ESS_STR_T	フィルタ名。
Count	ESS_USHORT_T	このフィルタに割り当てられているグループまたはユーザーのカウント。
pUserList	ESS_PUSERNAME_T	ユーザー名の配列へのポインタ。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetFilterList (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     FilterName;
 ESS_USHORT_T   Count = 0;
 ESS_USERNAME_T  UserList[2];

 AppName   = "Sample";
 DbName    = "Basic";
 FilterName = "Test";
 strcpy(UserList[0],"Jim Smith");
 strcpy(UserList[1],"Newuser");
 Count = 2;

 sts = EssSetFilterList(hCtx, AppName, DbName,
    FilterName, Count, UserList);
 return (sts);
}
```

関連トピック

- EssSetFilterListEx
- EssGetFilterList
- EssListFilters
- EssSetFilter

EssSetFilterListEx

フィルタに割り当てられたグループまたはユーザーのリストを設定します。count パラメータはフィルタに割り当てられたグループまたはユーザーの数を制御します。count がゼロの場合、グループまたはユーザーすべてがリストから削除されます。

EssSetFilterList に似ていますが、ユーザー・ディレクトリにホストされているユーザーおよびグループが含まれます。

構文

```
ESS_FUNC_M EssSetFilterListEx (
hCtx
,
AppName
,
DbName
,
FilterName
,
bIsIdentity
,
entityType
,
Count
,
UserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
AppName	ESS_STR_T	アプリケーション名(入力)。
DbName	ESS_STR_T	データベース名(入力)。
FilterName	ESS_STR_T	フィルタ名(入力)。
bIsIdentity	ESS_BOOL_T	入力。userList に名前または ID を含めるかどうかを指定します。TRUE の場合、userList には ID が含まれます。 リストには名前か ID を含めることができますが、両方を含めることはできません。
entityType	ESS_USHORT_T	userList に含まれるエンティティのタイプ(入力)。次のいずれかになります: ● ESS_TYPE_USER ● ESS_TYPE_GROUP リストにはユーザーかグループを含めることができますが、両方を含めることはできません。
Count	ESS_USHORT_T	userList に含まれるエンティティのカウント(入力)。
UserList	ESS_PSTR_T	ユーザー名かグループ名または ID の配列(入力)。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイン権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
void GetFilterList()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T FilterName;
 ESS_USHORT_T Count = 0;
 ESS_USHORT_T ind;
 ESS_PUSERNAME_T UserList = NULL;

 FilterName = "Filter1";

 sts = EssGetFilterList(hCtx, AppName, DbName, FilterName, &Count, &UserList);
 printf("EssGetFilterList sts: %ld\n", sts);
 if(!sts)
 {
  printf("--------%s User List---------\n", FilterName);
  if(Count && UserList)
  {
   for (ind = 0; ind < Count; ind++)
    printf("%s\n",UserList[ind]);
   EssFree(hInst, UserList);
  }
  else
   printf("none.");
  printf("\n");
 }

}


ESS_FUNC_M ESS_SetFilterListEx (ESS_HCTX_T hCtx)

{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T FilterName;
 ESS_USHORT_T Count = 0;
 //ESS_USERNAME_T UserList[2];
 ESS_STR_T UserList[2];
 ESS_BOOL_T bIsIdentity;
 ESS_USHORT_T type;

 FilterName = "Filter1";
 UserList[0] = "IDUser9@ldap";
 UserList[1] = "IDUser10@ldap";
 Count = 2;
 bIsIdentity = ESS_TRUE;
```

```
 type = ESS_TYPE_USER;

 sts = EssSetFilterListEx(hCtx, AppName, DbName, FilterName, bIsIdentity, type,
Count, UserList);
 printf("EssSetFilterListEx sts: %ld\n", sts);
 if(!sts)
  GetFilterList();

 return (sts);
}
```

関連トピック

- EssGetFilterList
- EssListFilters
- EssSetFilter

EssSetFilterRow

フィルタの次の行を設定します。

構文

```
ESS_FUNC_M EssSetFilterRow (
hCtx, RowString, Access
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RowString	ESS_STR_T	フィルタの次の行へのポインタ。
Access	ESS_ACCESS_T	フィルタ行のアクセス・レベル。

備考

この関数は、EssSetFilter を呼び出した後に、フィルタの各行について 1 回ずつ繰り返し呼び出し、行リストを NULL 行文字列ポインタで終了する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

EssSetFilter の例を参照してください。

関連トピック

- EssListFilters
- EssSetFilter

EssSetGlobalState

システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を設定します。

構文

```
ESS_FUNC_M EssSetGlobalState (
hCtx, pGlobal
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pGlobal	147 ページの「ESS_GLOBAL_T」	グローバル状態構造体へのポインタ。

備考

パラメータ値を変更するときは、最初に EssGetGlobalState を呼び出して、変更しないパラメータの正しい値を取得することをお薦めします。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M
ESS_SetGlobalState (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_GLOBAL_T  Global;

 /* Initialize Global State */
 Global.Security = 1;
 Global.Logins = 1;
 Global.Access = ESS_ACCESS_NONE;
 Global.Validity = 200;
 Global.Currency = 1;
 Global.PwMin = 8;
 Global.InactivityTime = 3600;
 Global.InactivityCheck = 300;

 sts = EssSetGlobalState(hCtx, &Global);
 return (sts);
```

```
}
```

関連トピック

- EssGetGlobalState

EssSetGroup

グループのセキュリティ情報を含むグループ情報構造体を設定します。

構文

```
ESS_FUNC_M EssSetGroup (
hCtx, pGroupInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pGroupInfo	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	グループ情報構造体へのポインタ。

備考

- 設定するグループ名は、グループ情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更できるグループ情報構造体のフィールドは、Access フィールドのみです(その他のフィールドはユーザーに情報を提供する目的のみに使用されます)。詳細は ESS_GROUPINFO_T 構造体の説明を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetGroup (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts  = ESS_STS_NOERR;
 ESS_USERINFO_T  Group;

 strcpy(Group.Name,"PowerUsers");
 strcpy(Group.AppName,"Sample");
 strcpy(Group.DbName,"Basic");
 Group.Access = ESS_ACCESS_SUPER;

 sts = EssSetGroup(hCtx, &Group);
```

```
 return (sts);
}
```

関連トピック

- EssGetGroup
- EssListGroups

EssSetGroupList

グループのメンバーであるユーザーのリストを設定します。

構文

```
ESS_FUNC_M  EssSetGroupList (
hCtx, GroupName, Count, pUserList
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	グループ名またはユーザー名。
Count	ESS_USHORT_T	ユーザー名のカウント。
pUserList	ESS_PUSERNAME_T	ユーザー名文字列の配列へのポインタ。

備考

この関数では、GroupName 引数としてユーザー名を使用し、pUserList 引数としてグループのリストを渡すことにより、ユーザーが属するグループのリストも設定できます。

Essbase 管理者以外の管理者がユーザーやグループの権限を管理するには、アクセスを管理するアプリケーション上でユーザーやグループより高い権限を持っている必要があります。この制限を回避するには MaxL を使用してグループにユーザーを追加します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetGroupList (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_USERNAME_T  UserList[3];
```

```
 ESS_USHORT_T   Count;
 ESS_STR_T    GroupName;
 GroupName = "Powerusers";
 strcpy(UserList[0],"App Designer");
 strcpy(UserList[1],"Db Designer");
 strcpy(UserList[2],"User Creator");
 Count = 3;

 sts = EssSetGroupList (hCtx, GroupName, Count,
    UserList);
 return (sts);
}
```

関連トピック

- EssSetGroupListEx
- EssAddToGroup
- EssDeleteFromGroup
- EssListGroups

EssSetGroupListEx

グループのメンバーであるユーザーのリストを設定します。EssSetGroupList に似ていますが、ユーザー・ディレクトリの指定、または EntityId の一意の ID 属性を受け入れることができます。

構文

```
ESS_FUNC_M EssSetGroupListEx (
hCtx
,
EntityId
,
bIsIdentity
,
entityType
,
Count
,
pIdList
,
bUsingIdentity
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(入力)。
EntityId	ESS_STR_T	ユーザー名またはグループ名(入力)。name@provider または一意の ID 属性として指定できます。
bIsIdentity	ESS_BOOL_T	入力。EntityId が名前か ID かを示します。TRUE の場合、EntityId は ID です。

パラメータ	データ型	説明
entityType	ESS_USHORT_T	入力。EntityId がグループかユーザーかを示します。ユーザーとして指定された場合、リストはグループのみで構成されます。グループとして指定された場合、リストにはネイティブ・セキュリティ・モードのグループのみ含まれますが、EPM System セキュリティ・モードでは、リストはユーザーとグループの両方で構成されます。
Count	ESS_USHORT_T	ID のカウント(入力)。
pIdList	ESS_PSTR_T	ID の配列へのポインタ(入力)。
bUsingIdentity	ESS_BOOL_T	入力。EntityId が名前か ID かを示します。TRUE の場合、EntityId は ID です。

備考

この関数では、EntityID 引数としてユーザー名を使用し、pUserIDList 引数としてグループのリストを渡すことにより、ユーザーが属するグループのリストも設定できます。

Essbase 管理者以外の管理者がユーザーやグループの権限を管理するには、アクセスを管理するアプリケーション上でユーザーやグループより高い権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

関連トピック

- EssAddToGroupEx
- EssDeleteFromGroupEx
- EssListGroupsInfoEx

EssSetGroupToSS

1 つのグループを EPM System セキュリティ・モードに移行します。
EssSetSSSecurityMode を使用したグループ移行が失敗した場合に役に立ちます。

構文

```
ESS_FUNC_M EssSetGroupToSS (
hCtx
,
GroupName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
GroupName	ESS_STR_T	Shared Services(入力)に変換するグループの名前。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetGroupToSS(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     groupName = ESS_NULL;

 sts = EssAlloc(hInst, sizeof(ESS_USERNAME_T), &groupName);
 if(sts)
  return (sts);
 memset(groupName, 0, sizeof(ESS_USERNAME_T));
 strcpy( groupName, "essgrp");

 sts = EssSetGroupToSS(hCtx, groupName);

 if(sts)
  printf("Failed to migrate Group %s to Shared Services mode.\n", groupName);

 if (groupName)
  EssFree(hInst, groupName);

 return (sts);
}
```

拡張された付録 B も参照してください

EssSetGroupsToSS

すべてのグループを EPM System セキュリティ・モードに移行します。
EssSetSSSecurityMode を使用したグループ移行が失敗した場合に役に立ちます。

構文

```
ESS_FUNC_M EssSetGroupsToSS (
hCtx
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetGroupsToSS(ESS_HCTX_T  hCtx)
{
 ESS_STS_T     sts = ESS_STS_NOERR;

 sts = EssSetGroupsToSS(hCtx);

 if(sts)
  printf("Failed to migrate Groups to Shared Services mode.\n");

 return (sts);
}
```

拡張された付録 B も参照してください

EssSetPassword

既存のパスワードを消去して、ユーザーのパスワードを設定します。

構文

```
ESS_FUNC_M  EssSetPassword (
hCtx, UserName, Password
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザー名。
Password	ESS_STR_T	ユーザーの新パスワード。

備考

- パスワードを変更するには、呼出し元は、スーパーバイザ・アクセス権を持っているか、または変更するパスワードが呼出し元のものである必要があります。
- 新パスワードは、次にユーザーがログインするときに有効になります。

戻り値

なし。

アクセス

この関数を使用するには、独自のパスワードを設定する場合を除き、呼出し元は、ログインしているサーバーに対してユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetPassword (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   UserName;
 ESS_STR_T   Password;

 UserName = "Jim Smith";
 Password = "newpwd";
 sts = EssSetPassword (hCtx,UserName, Password);
 return (sts);
}
```

関連トピック

- EssListUsers
- EssSetUser

EssSetPath

実行時プロセスのための ESSBASEPATH 環境変数を設定します。

構文

```
ESS_FUNC_M EssSetPath (
pszPath
);
```

パラメータ　データ型　説明

pszPath;　ESS_STR_T ESSBASEPATH 環境変数を説明する文字列へのポインタ

備考

- EssSetPath()は、EssInit()を呼び出す前に呼び出します。
- pszPath は、ESS_PATHLEN で定義する場合 120 文字以下とする必要があります。
- pszPath は、現在のプロセスにのみ適用されます。
- Essbase DLL はシステム・パスからアクセスできる必要があります。EssSetPath() は Essbase DLL のパスを解決しません。

戻り値

- 正常終了の場合は、ESS_STS_NOERR が戻されます。
- pszPath が長すぎる場合は、API_NAME_TOO_LONG(1030009)が戻されます。

例

```
    ESS_STS_T
ESS_SetPath()
{
 ESS_STS_T sts;
 ESS_STR_T pszPath = "C:\Hyperion\products\Essbase";
 sts = EssSetPath (pszPath);
 return sts;
}
```

EssSetServerMode

Essbase サーバーのモードを Unicode または非 Unicode に設定します。Unicode モードの場合にかぎり、Essbase サーバーで Unicode モードのアプリケーション作成や、非 Unicode モードから Unicode モードへのアプリケーションの移行が可能になります。Unicode モードのサーバーを非 Unicode に設定しても、既存のアプリケーションの Unicode 関連モードには影響を及ぼしません。

構文

```
ESS_FUNC_M EssSetServerMode(
hCtx
,
bUnicode
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル(ログイン済)
bUnicode	ESS_BOOL_T	入力パラメータ、bUnicode。bUnicode には次のどちらかの値を指定できます: ● ESS_TRUE – サーバー・モードを Unicode に設定します。Essbase サーバーでは Unicode モードのアプリケーションの作成や、非 Unicode モードから Unicode モードへのアプリケーションの移行が可能です。 ● ESS_FALSE – サーバー・モードを非 Unicode に設定します。Essbase サーバーでは Unicode モードのアプリケーション作成や、非 Unicode モードから Unicode モードへのアプリケーションの移行は行えません。

戻り値

なし。

アクセス

この関数を使用するには、呼出し側が、ログインするサーバーに対する(AD_ACCESS_SUPER)権限を持っている必要があります。

関連トピック

- EssGetServerMode

EssSetSpanRelationalPartition

Essbase に関係するデータが接続したリレーショナル・ストアに存在することを通知する、ブール bSpanRelPart フィールドを設定します。
EssQueryDatabaseMembers などのその他の API 関数の一部は、bSpanRelPart を読み込み、bSpanRelPart が設定されている場合はリレーショナル・ストアにアクセスします。

構文

```
ESS_FUNC_M EssSetSpanRelationalPartition (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

備考

一部の API 関数は、リレーショナル・ストアから情報を取得するように拡張されています。

- EssQueryDatabaseMembers - リレーショナル・ストアからメンバー名を戻します。
- EssGetMemberInfo - リレーショナル・ストア内のメンバーに関する情報を戻します。
- EssCheckMemberName - リレーショナル・ストアで有効なメンバー名を確認します。
- EssGetMemberCalc - 入力として渡されたリレーショナル・メンバーを認識し、すべてのリレーショナル・メンバーに対して NULL 文字列を戻します。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_Report (ESS_HCTX_T  hCtx,
     ESS_HINST_T  hInst
    )
{
```

```
 ESS_FUNC_M    sts    = ESS_STS_NOERR;
 ESS_STR_T    rString = NULL;
 sts = EssBeginReport (hCtx,ESS_TRUE,ESS_FALSE);
 if (!sts)
  sts = EssSendString (hCtx, "<Desc Year !");
 if (!sts)
  sts = EssSetSpanRelationalPartition (hCtx);
 /**************
  * Get report *
  **************/

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
    printf ("%s", rString);
    EssFree (hInst, rString);
    sts = EssGetString (hCtx, &rString);
 }
 printf ("\r\n");

 return(sts);
}
```

関連トピック

- EssClrSpanRelationalSource

EssSetSSSecurityMode

Essbase サーバーと既存のユーザーおよびグループを EPM System セキュリティ・モードに移行します。

構文

```
ESS_FUNC_M EssSetSSSecurityMode (
hCtx
,
option
,
NewPassword
,
FileName
,
flag
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
option	ESS_USHORT_T	移行したユーザーに必要なパスワードの作成方法を示す整数。 ● 0 - 管理者が指定したパスワードを使用します。 ● 1 - Shared Services に移行されるユーザーに対してユーザー名と同じパスワードが作成されます。 **注：** ユーザー名に大文字が存在する場合でも、パスワードは小文字で作成されます。 ● 2 - Shared Services に移行するユーザーの新パスワードを自動的に生成します。
NewPassword	ESS_STR_T	パスワードの文字列(option 2 が使用されている場合)。
FileName	ESS_STR_T	保存されたパスワードを含むファイルの名前。指定しない場合、デフォルトのファイルは$ARBORPATH/bin/MigratedUsersPassword.txt です。
flag	ESS_USHORT_T	パスワード・ファイルがすでに存在する場合に、上書きされるかどうかを設定します。 ● ESS_FILE_OVERWRITE - 上書き ● ESS_FILE_NOOVERWRITE - 上書きせず、エラーを戻す。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetSSSecurityMode(ESS_HCTX_T  hCtx)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     newpassword = ESS_NULL;
 ESS_USHORT_T    option;
 ESS_STR_T     fName = ESS_NULL;
 ESS_USHORT_T    flag = 0;

 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/

 option = 1; /* Using user name as password */

 sts = EssSetSSSecurityMode(hCtx, option, newpassword, fName, flag);

 if(sts)
  printf("Failed to migrate Essbase Server to Shared Services mode.\n");

 return (sts);
```

}

拡張された付録 B も参照してください

関連トピック

- EssSetUserToSS
- EssSetGroupToSS

EssSetUserToSS

1 人のユーザーを EPM System セキュリティ・モードに移行します。
EssSetSSSecurityMode を使用したユーザー移行が失敗した場合に役に立ちます。

構文

```
ESS_FUNC_M EssSetUserToSS (
hCtx
,
UserName
,
option
,
NewPassword
,
FileName
,
flag
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	Shared Services に変換するユーザー名(入力)。
option	ESS_USHORT_T	移行したユーザーに必要なパスワードの作成方法を示す整数。 ● 0 - 管理者が指定したパスワードを使用します。 ● 1 - Shared Services に移行されるユーザーに対してユーザー名と同じパスワードが作成されます。 **注：** ユーザー名に大文字が存在する場合でも、パスワードは小文字で作成されます。 ● 2 - Shared Services に移行するユーザーの新パスワードを自動的に生成します。
NewPassword	ESS_STR_T	パスワードの文字列(option 2 が使用されている場合)。
FileName	ESS_STR_T	保存されたパスワードを含むファイルの名前。指定しない場合、デフォルトのファイルは$ARBORPATH/bin/MigratedUsersPassword.txt です。

パラメータ	データ型	説明
flag	ESS_USHORT_T	パスワード・ファイルがすでに存在する場合に、上書きされるかどうかを設定します。 ● ESS_FILE_OVERWRITE - 上書き ● ESS_FILE_NOOVERWRITE - 上書きせず、エラーを戻す。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetUserToSS(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_USHORT_T   option;
 ESS_STR_T     userName = ESS_NULL;
 ESS_STR_T     newpassword = ESS_NULL;
 ESS_STR_T     fName = ESS_NULL;
 ESS_USHORT_T   flag = ESS_FILE_OVERWRITE;

 sts = EssAlloc(hInst, sizeof(ESS_USERNAME_T), &userName);
 if(sts)
  return (sts);
 memset(userName, 0, sizeof(ESS_USERNAME_T));
 strcpy( userName, "essexer");

 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/

 option = 2; /* Generate password */

 sts = EssSetUserToSS(hCtx, userName, option, newpassword, fName, flag);

 if(sts)
  printf("Failed to migrate User %s to Shared Services mode.\n", userName);

 if (userName)
  EssFree(hInst, userName);

 return (sts);
}
```

拡張された付録 B も参照してください

関連トピック

- EssSetUsersToSS
- EssSetGroupToSS
- EssSetSSSecurityMode

EssSetUser

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。

構文

```
ESS_FUNC_M EssSetUser (
hCtx, pUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pUserInfo	204 ページの「ESS_USERINFO_T, ESS_GROUPINFO_T」	ユーザー情報構造体へのポインタ。

備考

- 設定するユーザー名は、ユーザー情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更可能なユーザー情報構造体内のフィールドは、アクセス、Expiration および PwdChgNow のみです(他のフィールドは単なる情報提供用)。詳細は、ESS_USERINFO_T 構造体の説明を参照してください。
- 呼出し元は、指定したユーザーに対して、呼出し元ユーザーが所有していないアクセス権限は付与できません。
- 新規のユーザー設定は、次にユーザーがログインする際に有効になります。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_SetUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts  = ESS_STS_NOERR;
 ESS_USERINFO_T  User;

 strcpy(User.Name,"Jim Smith");
 strcpy(User.AppName,"Sample");
 strcpy(User.DbName,"Basic");
```

```
 User.Access = ESS_ACCESS_SUPER;

 sts = EssSetUser (hCtx,&User);
 return (sts);
}
```

関連トピック

- EssGetUser
- EssListUsers
- EssSetApplicationAccess
- EssSetPassword

EssSetUserEx

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。

構文

```
ESS_FUNC_M EssSetUser (
hCtx, pUserInfo
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pUserInfoEx	207 ページの「ESS_USERINFOEX_T」	外部認証ユーザーの情報構造体へのポインタ。

備考

- 設定するユーザー名は、ユーザー情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更可能なユーザー情報構造体内のフィールドは、Access、Expiration および PwdChgNow のみです(他のフィールドは単なる情報提供用)。詳細は、ESS_USERINFO_T 構造体の説明を参照してください。
- 呼出し元は、指定したユーザーに対して、呼出し元ユーザーが所有していないアクセス権限は付与できません。
- 新規のユーザー設定は、次にユーザーがログインする際に有効になります。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

例

```
ESS_FUNC_M
```

```
ESS_SetUser (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts  = ESS_STS_NOERR;
 ESS_USERINFO_T  User;

 strcpy(User.Name,"Jim Smith");
 strcpy(User.AppName,"Sample");
 strcpy(User.DbName,"Basic");
 User.Access = ESS_ACCESS_SUPER;

 sts = EssSetUserEx (hCtx,&User);
 return (sts);
}
```

関連トピック

- EssGetUser
- EssListUsers
- EssCreateExtUser
- EssGetUserEx
- 207 ページの「ESS_USERINFOEX_T」
- EssSetApplicationAccess
- EssSetPassword

EssSetUsersToSS

すべてのユーザーを Oracle Enterprise Performance Management System セキュリティ・モードに移行します。EssSetSSSecurityMode を使用したユーザー移行が失敗した場合に役に立ちます。

構文

```
ESS_FUNC_M EssSetUsersToSS (
hCtx
,
option
,
NewPassword
,
FileName
,
flag
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。

パラメータ	データ型	説明
option	ESS_USHORT_T	移行したユーザーに必要なパスワードの作成方法を示す整数。 ● 0 - 管理者が指定したパスワードを使用します。 ● 1 - Shared Services に移行されるユーザーに対してユーザー名と同じパスワードが作成されます。 **注：** ユーザー名に大文字が存在する場合でも、パスワードは小文字で作成されます。 ● 2 - Oracle Hyperion Shared Services に移行するユーザーの新パスワードを自動的に生成します。
NewPassword	ESS_STR_T	パスワードの文字列(option 2 が使用されている場合)。
FileName	ESS_STR_T	保存されたパスワードを含むファイルの名前。NULL の場合、デフォルト・ファイルは$ARBORPATH/bin/MigratedUsersPassword.txt です。
flag	ESS_USHORT_T	パスワード・ファイルがすでに存在する場合に、上書きされるかどうかを設定します。 ● ESS_FILE_OVERWRITE - 上書き ● ESS_FILE_NOOVERWRITE - 上書きせず、エラーを戻す。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M ESS_SS_SetUsersToSS(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_USHORT_T   option;
 ESS_STR_T     newpassword = ESS_NULL;
 ESS_STR_T     fName = "PasswordList.txt";


 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/


 option = 2; /* Generate a password */




 sts = EssSetUsersToSS(hCtx, option, newpassword, fName, ESS_FILE_OVERWRITE);
```

```
 if(sts)
  printf("Failed to migrate Users to Shared Services mode.\n");

 return (sts);

}
```

拡張された付録 B も参照してください

関連トピック

- EssSetUserToSS
- EssSetGroupToSS
- EssSetSSSecurityMode

EssSetUserType

ユーザーのアプリケーション・アクセス・タイプを定義できます。ユーザー名に対する様々なアプリケーション・アクセス・タイプの追加、削除や置換が可能です。

アプリケーション・アクセス・タイプはユーザー・プロパティです。Oracle Hyperion Planning で作成されたユーザーには planning、Administration Services で作成されたユーザーには Essbase というアプリケーション・アクセス・タイプが、それぞれ自動的に割り当てられます。ユーザーの作成後、対応するアプリケーションで EssSetUserType を使用してアプリケーション・アクセス・タイプを変更できます。

構文

```
ESS_FUNC_M EssSetUserType (
hCtx
,
UserName
,
UserType
,
Cmd
)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
UserName	ESS_STR_T	ユーザーの名前。
UserType	ESS_USER_TYPE_T	アプリケーション・アクセス・タイプが指定されていない場合、ESS_USER_ESSBASE がアプリケーション・アクセス・タイプになります。このユーザーはすべての機能が使用可能です。

パラメータ	データ型	説明
Cmd	ESS_USERTYPE_CMD_T	指定されたタイプの追加/削除/置換のいずれかを示します。Essbase タイプである ESS_USER_ESSBASE のみ、追加または削除できます。 ● ESS_USERTYPE_CMD_ADD - 指定された新しいタイプを既存のアプリケーション・アクセス・タイプに追加します。 ● ESS_USERTYPE_CMD_REMOVE - 指定されたタイプを既存のアプリケーション・アクセス・タイプから削除します。

戻り値

API のステータスを戻します。

例

```
    ESS_FUNC_M
ESS_SetUserType (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M          sts = ESS_STS_NOERR;
 ESS_STR_T          UserName="jsmith";
 ESS_USER_TYPE_T       UserType = ESS_USER_ESSBASE;
 ESS_USERTYPE_CMD_T      cmd = ESS_USERTYPE_CMD_ADD;
 sts = EssSetUserType     (hCtx, UserName, UserType, Cmd);
 return (sts);
}
```

関連トピック

- EssGetUserType

EssShutdownServer

Essbase エージェントを停止します。この関数はエージェント(ESSBASE.EXE)に対して、自身をシャットダウンするように要求を送信します。エージェントは、データのコミット、すべてのアプリケーションとデータベースの終了、ユーザーのログオフ後の停止など、通常のシャットダウン手順を実行します。

スーパーバイザ権限を持っているユーザーのみ、エージェントをシャットダウンできます。

この関数はいつでも呼び出せますが、通常はバックグラウンドで起動されたエージェントのシャットダウンのために呼び出します。詳細は、『Oracle Essbase データベース管理者ガイド』を参照してください。

構文

```
ESS_FUNC_M  EssShutdownServer (
hInstance, Server, UserName, Password
);
```

パラメータ	データ型	説明
hInstance	ESS_HINST_T	API インスタンス・ハンドル。
Server	ESS_STR_T	ネットワーク・サーバー名の文字列。シャットダウンするサーバーの名前を指定します。
UserName	ESS_STR_T	ユーザー名の文字列。シャットダウンを要求しているユーザーを指定します。
Password	ESS_STR_T	パスワード文字列。シャットダウンを要求しているユーザーのパスワードを指定します。

戻り値

この関数の結果として考えられるエラー条件には、次のものがあります:

- AD_AMSG_IPO は、この操作のための権限が不足していることを示します
- AD_AMSG_IPW は、パスワードが正しくないことを示します
- AD_AMSG_UNE は、ユーザーが存在しないことを示します
- AD_MSGAR_NOSHUTDOWN は、アプリケーションをシャットダウンできないことを示します
- ネットワーク・エラー: NET_TCP_HOSTS は、ホスト・ファイルで検索できないことを示します
- ネットワーク・エラー: NET_NP_NOSERVER は、サーバーを検索できないことを示します

アクセス

この関数を使用するには、スーパーバイザ権限を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_ShutdownServer (ESS_HINST_T hInst)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   Server;
 ESS_STR_T   UserName;
 ESS_STR_T   Password;

 Server = "Rainbow";
 UserName = "Admin";
 Password = "password";
 sts = EssShutdownServer(hInst, Server,
    UserName, Password);
 return (sts);
}
```

関連トピック

- EssSetPassword
- EssUnloadApplication
- EssUnloadDatabase

EssTerm

API を終了し、API で使用されているすべてのシステム・リソースを解放します。この関数は、通常は他のすべての API 呼出しが完了した後すぐに、プログラムが終了する前に呼び出す必要があります。

構文

```
ESS_FUNC_M  EssTerm (
hInstance
);
```

パラメータ　データ型　　説明

hInstance　ESS_HINST_T API インスタンス・ハンドル。

備考

この関数は Essbase API の使用を終了させるため、この関数が実行された後に API 関数(EssInit 以外)を呼び出すと、いずれもエラーを戻します。

戻り値

なし。

アクセス

この関数には、特別なアクセス権は必要ありません。

例

```
    /* Terminate the Essbase API */
if ((sts = EssTerm (hInstance)) != ESS_STS_NOERR)
{
  /* error terminating API */
  exit ((ESS_USHORT_T) sts);
}
```

関連トピック

- EssInit

EssUnloadApplication

サーバー上のアプリケーションを停止します。

構文

```
ESS_FUNC_M  EssUnloadApplication (
hCtx, AppName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	ロードするアプリケーションの名前。

備考

アプリケーションをアンロードするには、接続しているユーザーがアプリケーションに対してロード権限を持っている必要があります。アプリケーションに関連付けられているデータベースを Essbase が再構築している場合は、そのアプリケーションをアンロードできません。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定されたアプリケーションに対してアプリケーションのロード/アンロード権限(ESS_PRIV_APPLOAD)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_UnloadApplication (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_STR_T   AppName;

 AppName = "Sample";
 sts = EssUnloadApplication(hCtx, AppName);

 return (sts);
}
```

関連トピック

- EssLoadApplication
- EssUnloadDatabase

EssUnloadDatabase

サーバー上でアプリケーション内のデータベースを停止します。

構文

```
ESS_FUNC_M  EssUnloadDatabase (
hCtx, AppName, DbName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	アンロードするデータベースの名前。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースのロード/アンロード権限(ESS_PRIV_APPLOAD)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_UnloadDb (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 AppName = "Sample";
 DbName  = "Basic";
 /*
  * IF the current active is the same as the
  * unload db, ClearActive first
  */
   sts = EssClearActive(hCtx);
 /*
  * ELSE
  */
   sts = EssUnloadDatabase(hCtx, AppName,
      DbName);
 return (sts);
}
```

関連トピック

- EssLoadDatabase
- EssUnloadDatabase

EssUnlockObject

サーバーまたはクライアント・オブジェクト・システム上でロックされているオブジェクトをロック解除します。

構文

```
ESS_FUNC_M  EssUnlockObject (
hCtx, ObjType, AppName, DbName, ObjName
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。EssCreateLocalContext によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	ESS_OBJTYPE_T	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、102 ページの「ビットマスク・データ型(C)」を参照してください。
AppName	ESS_STR_T	アプリケーション名
DbName	ESS_STR_T	データベース名。NULL の場合は、アプリケーション・サブディレクトリを使用します。
ObjName	ESS_STR_T	ロック解除するオブジェクトの名前。

備考

オブジェクトをロック解除するには、そのオブジェクトが存在し、呼出し元によってロックされている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、オブジェクトが含まれる指定されたアプリケーションまたはデータベースに対してアプリケーション・デザイン権限またはデータベース・デザイン権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_UnlockObject (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M     sts = ESS_STS_NOERR;
 ESS_STR_T     AppName;
 ESS_STR_T     DbName;
 ESS_STR_T     ObjName;
 ESS_OBJTYPE_T   ObjType;

 AppName = "Sample";
 DbName  = "Basic";
 ObjName = "Basic";
 ObjType = ESS_OBJTYPE_OUTLINE;

 sts = EssUnlockObject(hCtx, ObjType, AppName,
    DbName, ObjName);
 if(!sts)
  printf("The Object is unlocked\r\n");
 return (sts);
}
```

関連トピック

- EssGetObject
- EssGetObjectInfo
- EssListObjects
- EssLockObject
- EssPutObject

EssUpdate

アクティブなデータベースに更新指定を単一文字列として送信します。

構文

```
ESS_FUNC_M  EssUpdate (
hCtx, Store, Unlock, UpdtSpec
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
UpdtSpec	ESS_STR_T	単一文字列としての更新指定。

備考

- この関数は、EssBeginUpdate を呼び出し、その後 EssSendString を呼び出し、最後に EssEndUpdate を呼び出すのと同じです。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。
- この関数によってデータが保管される場合(Store フラグが TRUE)、関連データ・ブロックは更新のためにロックされている必要があります(たとえば、Lock フラグを TRUE に設定して EssReport を呼び出します)。
- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。呼出し元が情報を書き込もうとした場合。

例

```
    ESS_FUNC_M
ESS_Update (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;

 sts = EssUpdate (hCtx, ESS_TRUE, ESS_FALSE,
   "Year Market Scenario Measures Product 100");
 return(sts);
}
```

関連トピック

- EssBeginUpdate
- EssEndUpdate
- EssReport
- EssSendString
- EssUpdateFile

EssUpdateBakFile

セキュリティ・バックアップ・ファイル essbase_timestamp.bak とセキュリティ・ファイル essbase.sec を比較し、両者が一致しない場合はバックアップ・ファイルをセキュリティ・ファイルで上書きします。

構文

```
ESS_FUNC_M EssUpdateBakFile (
hCtx
);
```

パラメータ　データ型　　説明

hCtx;　　ESS_HCTX_T API コンテキスト・ハンドル。

備考

Essbase はセキュリティ・ファイルとバックアップ・ファイルをサーバーが起動するたびに比較します。MaxL ステートメント **alter system sync security_backup** を使用して、指定した間隔またはオンデマンドで、セキュリティ・バックアップ・ファイルとセキュリティ・ファイルを自動的に比較できます。Essbase がファイルを比較すると、バックアップ・ファイルがセキュリティ・ファイルと一致するように更新されます。

戻り値

成功の場合、ゼロが戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    ESS_FUNC_M
ESS_UpdateBakFile (ESS_HCTX_T  hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;

 sts = EssUpdateBakFile(hCtx);

 return sts;
}
```

EssUpdateDrillThruURL

アクティブなデータベース・アウトライン内で、指定された名前のドリルスルー URL を更新します。

1904 ページの「ドリルスルー URL の制限」を参照してください。

構文

```
ESS_FUNC_M EssUpdateDrillThruURL (
hCtx, ESS_PDURLINFO_T pUrl
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
pUrl	ESS_PDURLINFO_T	URL 定義。
bMerge	ESS_BOOL_T	• TRUE の場合、pUrl 内のドリルスルー領域定義を、指定された URL 定義内にある既存のドリルスルー領域のリストに追加します • FALSE の場合、既存のドリルスルー領域定義のリストを、pUrl 内のリストで置き換えます

戻り値

- 正常に処理されると、URL XML の置換と、pUrl 内の対応するフィールドによるドリルスルー領域のリストの更新または置換によって、アクティブなデータベース内の指定されたドリルスルー URL が更新されます。
- 指定された名前の URL が存在しない場合は、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESS_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EssSetActive を使用して、指定したデータベースをアクティブ・データベースとして選択しておく必要があります。

例

```
    /* Sample Code for EssUpdateDrillThruURL */

ESS_STS_T sts = ESS_STS_NOERR;
ESS_DURLINFO_T url;
ESS_PDURLINFO_T urlInfo;
ESS_STR_T fileName = "";
ESS_CHAR_T xmlString[XML_CHAR_MAX];
ESS_BOOL_T bMerge;
ESS_USHORT_T i;

memset(&url, '\0', sizeof(ESS_DURLINFO_T));
fileName = "F:\\testarea\\mainapi\\sample1.xml";
GetFileContent(fileName, xmlString);

/* Update URL*/
url.bIsLevel0 = ESS_TRUE;
url.cpURLName = "Drill Through to EPMI";
url.cpURLXml = xmlString;
url.iURLXmlSize = (ESS_SHORT_T) strlen(xmlString)+1;
url.iCountOfDrillRegions = 1;
sts = EssAlloc (hInst, sizeof(ESS_STR_T) * url.iCountOfDrillRegions,
&(url.cppDrillRegions));

/* With bMerge = ESS_FALSE, update Drill Regions */

bMerge = ESS_FALSE;  // replace
url.cppDrillRegions[0] = "Mar";
sts = EssUpdateDrillThruURL(hCtx, &url, bMerge);
printf("EssUpdateDrillThruURL sts: %ld\n",sts);
```

EssUpdateEx

すべてのデータ・ロード・エラーを ppMbrError に取得して、アクティブなデータベースに更新指定を単一文字列として送信します。

構文

```
ESS_FUNC_M  EssUpdateEx (
hCtx, Store, Unlock, UpdtSpec, ppMbrError
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
UpdtSpec	ESS_STR_T	単一文字列としての更新指定。
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● AD_MSGDL_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_UNKNOWN - データのロード時に不明なメンバー[membername]が見つかりました。[number]レコードが戻されました。。

備考

- 更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。
- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
void TestUpdateEx()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_PMBRERR_T pMbrError;
    ESS_STR_T updtSpec = "";
```

```
    sts = EssUpdateEx(hCtx, ESS_TRUE, ESS_FALSE, "'Jan' 'New York' 'Actual' 'Sales'
'100-10' 123 \n '100-20' 345 \n '100-30' 678", &pMbrError);
    printf("EssUpdateEx sts: %ld\n",sts);
    if(!sts)
    {
        printf("\nVerify data:\n");
        VerifyDataload("'Jan' 'New York' 'Actual' 'Sales' <IDESC '100'!");

        printf("\nMember Error Info:\n");
        if(pMbrError)
            DisplayError(pMbrError);
        else
            printf("\tError structure is empty.\n");
    }

    if(pMbrError)
        EssFree(hInst, pMbrError);
}
```

関連トピック

- EssUpdateFileASO
- EssUpdateFileASOEx
- EssUpdateFileEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateFile

ファイルからアクティブ・データベースに対して更新指定を送信します。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

構文

```
ESS_FUNC_M  EssUpdateFile (
hDestCtx, hSrcCtx, AppName, DbName, FileName, Store, Unlock
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	レポート・ファイルの場所に対する API コンテキスト・ハンドル。レポート・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。

パラメータ	データ型	説明
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。

備考

- この関数によってデータが保管される場合(Store フラグが TRUE)、関連データ・ブロックは更新のためにロックされている必要があります(たとえば、Lock フラグを TRUE に設定して EssReport を呼び出します)。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    ESS_FUNC_M
ESS_UpdateFile (ESS_HCTX_T   hCtx)
{
 ESS_FUNC_M   sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_BOOL_T  isStore;
 ESS_BOOL_T  isUnlock;
 ESS_STR_T   AppName;
 ESS_STR_T   DbName;
 ESS_STR_T   FileName;

 AppName  = "Sample";
 DbName   = "Basic";
 hSrcCtx  = hCtx;
 isStore  = ESS_TRUE;
 isUnlock = ESS_FALSE;
 sts = EssUpdateFile (hCtx, hSrcCtx, AppName,
    DbName, FileName, isStore, isUnlock);
 return(sts);
}
```

関連トピック

- EssBeginUpdate
- EssReportFile

- EssUpdate

EssUpdateFileASO

ファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。

構文

```
      ESS_FUNC_M EssUpdateFileASO (
      hDestCtx, hSrcCtx, AppName, DbName,
FileName, Store, Unlock, ulBufferId
      );
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	集約ストレージ・データベースではサポートされていません。このパラメータには必ず ESS_FALSE を渡す必要があります。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID 番号。

備考

Store フラグが FALSE に設定されている場合、データベースは更新指定の構文チェックのみを行います。

戻り値

正常終了の場合は 0 が戻され、それ以外の場合はエラー・コードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
    void TestUpdateFileASO(ESS_HCTX_T hCtx, ESS_STR_T AppName, ESS_STR_T DbName)
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HCTX_T    hSrcCtx;
    ESS_BOOL_T    isStore;
```

```
    ESS_BOOL_T   isUnlock;
    ESS_STR_T    FileName;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T   ulDuplicateAggregationMethod;
    ESS_ULONG_T   ulOptionsFlags;
    ESS_ULONG_T   ulSize;
    ESS_ULONG_T   ulBufferCnt;
    ESS_ULONG_T   ulCommitType ;
    ESS_ULONG_T   ulActionType;
    ESS_ULONG_T   ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];

    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 100;
    ulBufferId = 101;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* Update from server*/
    hSrcCtx  = hCtx;
    isStore  = ESS_TRUE;
    isUnlock = ESS_FALSE;
    FileName = "data1.txt";

    sts = EssUpdateFileASO (hCtx, hSrcCtx, AppName, DbName, FileName, isStore,
isUnlock, ulBufferId);
    printf("EssUpdateFileASO sts: %ld\n",sts);

    /* Commit and delete the buffer */
    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
    printf("\nLoad data to main slice and destroy buffer:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);
}
```

関連トピック

- EssUpdateEx
- EssUpdateFileASOEx
- EssUpdateFileEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateFileASOEx

すべてのデータ・ロード・エラーを ppMbrError に取得して、ファイルからアクティブな集約ストレージ・データベースに更新指定を送信します。

構文

```
     ESS_FUNC_M EssUpdateFileASOEx (
     hDestCtx, hSrcCtx, AppName, DbName, FileName, Store, Unlock, ulBufferId,
ppMbrError
     );
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	集約ストレージ・データベースではサポートされていません。このパラメータには必ず ESS_FALSE を渡す必要があります。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID 番号。
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● AD_MSGDL_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_UNKNOWN - データのロード時に不明なメンバー[membername]が見つかりました。[number]レコードが戻されました。。

備考

Store フラグが FALSE に設定されている場合、データベースは更新指定の構文チェックのみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
void TestUpdateFileASOEx()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_HCTX_T hSrcCtx;
    ESS_BOOL_T isStore;
    ESS_BOOL_T isUnlock;
    ESS_STR_T FileName;
    ESS_ULONG_T ulBufferId;
    ESS_ULONG_T    ulDuplicateAggregationMethod;
    ESS_ULONG_T    ulOptionsFlags;
    ESS_ULONG_T    ulSize;
    ESS_ULONG_T    ulBufferCnt;
    ESS_ULONG_T    ulCommitType ;
    ESS_ULONG_T    ulActionType;
    ESS_ULONG_T    ulOptions;
    ESS_ULONG_T ulBufferIdAry[1];
    ESS_PMBRERR_T pMbrError;

    ulDuplicateAggregationMethod = ESS_ASO_DATA_LOAD_BUFFER_DUPLICATES_ADD;
    ulOptionsFlags = ESS_ASO_DATA_LOAD_BUFFER_IGNORE_MISSING_VALUES;
    ulSize = 1;
    ulBufferId = 101;
    sts = EssLoadBufferInit(hCtx, AppName, DbName, ulBufferId,
ulDuplicateAggregationMethod,
        ulOptionsFlags, ulSize);
    printf("EssLoadBufferInit sts: %ld\n", sts);

    /* Update from server*/
    hSrcCtx  = hCtx;
    isStore  = ESS_TRUE;
    isUnlock = ESS_FALSE;
    FileName = "apgeaso1.txt";

    sts = EssUpdateFileASOEx (hCtx, hSrcCtx, AppName, DbName, FileName, isStore,
isUnlock, ulBufferId, &pMbrError);
    printf("EssUpdateFileASOEx sts: %ld\n",sts);
    if(!sts)
    {
        printf("\nMember Error Info:\n");
        if(pMbrError)
            DisplayError(pMbrError);
        else
            printf("\tError structure is empty.\n");
    }

    ulBufferCnt = 1;
    ulBufferIdAry[0] = ulBufferId;
    ulCommitType = ESS_ASO_DATA_LOAD_BUFFER_STORE_DATA;
    ulActionType = ESS_ASO_DATA_LOAD_BUFFER_COMMIT;
```

```
    printf("\nIncrement to main slice and destroy buffer:\n");
    ulOptions = ESS_ASO_DATA_LOAD_INCR_TO_MAIN_SLICE;
    sts = EssLoadBufferTerm(hCtx, AppName, DbName, ulBufferCnt, ulBufferIdAry,
ulCommitType,
        ulActionType, ulOptions);
    printf("EssLoadBufferTerm sts: %ld\n",sts);
    if(!sts)
    {
        VerifyDataload("'Mar' 'Sale' 'Curr Year' 'Original Price' '017589' '13668'
'Cash' 'No Promotion' '1 to 13 Years' 'Under 20,000' 'Digital Cameras' 10\n
'Camcorders' 20\n 'Photo Printers' 30 !");
    }

    if(pMbrError)
        EssFree(hInst, pMbrError);
}
```

関連トピック

- EssUpdateEx
- EssUpdateFileEx
- EssUpdateFileASO
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateFileEx

すべてのデータ・ロード・エラーを ppMbrError に取得して、ファイルからアクティブなデータベースに更新指定を送信します。

構文

```
ESS_FUNC_M EssUpdateFileEx (
hDestCtx, hSrcCtx, AppName, DbName, FileName, Store, Unlock, ppMbrError
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。

パラメータ	データ型	説明
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● AD_MSGDL_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_UNKNOWN - データのロード時に不明なメンバー[membername]が見つかりました。[number]レコードが戻されました。。

備考

- 更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。
- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

```
void TestUpdateFileEx()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_HCTX_T hSrcCtx;
    ESS_BOOL_T isStore;
    ESS_BOOL_T isUnlock;
    ESS_STR_T FileName;
    ESS_PMBRERR_T pMbrError;

    hSrcCtx  = hCtx;
    FileName = "apgebso1.txt";
    isStore  = ESS_TRUE;
    isUnlock = ESS_FALSE;
```

```
    sts = EssUpdateFileEx (hCtx, hSrcCtx, AppName, DbName, FileName, isStore,
isUnlock, &pMbrError);
    printf("EssUpdateFileEx sts: %ld\n",sts);
    if(!sts)
    {
        printf("\nVerify data:\n");
        VerifyDataload("'Jan' 'New York' 'Actual' 'Sales' <IDESC '100'!");

        printf("\nMember Error Info:\n");
        if(pMbrError)
            DisplayError(pMbrError);
        else
            printf("\tError structure is empty.\n");
    }

    if(pMbrError)
        EssFree(hInst, pMbrError);
}
```

関連トピック

- EssUpdateEx
- EssUpdateFileASO
- EssUpdateFileASOEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateFileUTF8ASO

UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに対して更新指定を送信します。

構文

```
     ESS_FUNC_M EssUpdateFileUTF8ASO (
     hDestCtx, hSrcCtx, AppName, DbName,
FileName, Store, Unlock, ulBufferId
     );
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。

パラメータ	データ型	説明
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	集約ストレージ・データベースではサポートされていません。このパラメータには必ず ESS_FALSE を渡す必要があります。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID 番号。

備考

Store フラグが FALSE に設定されている場合、データベースは更新指定の構文チェックのみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

EssUpdateFileAso の例を参照してください。

関連トピック

- EssUpdateFileASO
- EssUpdateFileASO
- EssUpdateFileASOEx
- EssUpdateFileEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateUtf8Ex

EssUpdateFileUTF8ASOEx

すべてのデータ・ロード・エラーを ppMbrError に取得して、UTF-8 でエンコードされたファイルからアクティブな集約ストレージ・データベースに更新指定を送信します。

構文

```
      ESS_FUNC_M EssUpdateFileUTF8ASOEx (
      hDestCtx, hSrcCtx, AppName, DbName,
FileName, Store, Unlock, ulBufferId, ppMbrError
      );
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	集約ストレージ・データベースではサポートされていません。このパラメータには必ず ESS_FALSE を渡す必要があります。
ulBufferId	ESS_ULONG_T	データ・ロード・バッファの ID 番号。
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● AD_MSGDL_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_UNKNOWN - データのロード時に不明なメンバー[membername]が見つかりました。[number]レコードが戻されました。。

備考

Store フラグが FALSE に設定されている場合、データベースは更新指定の構文チェックのみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

例

EssUpdateFileAso の例を参照してください。

関連トピック

- EssUpdateEx
- EssUpdateFileEx
- EssUpdateFileASO

- EssUpdateFileASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateFileUtf8Ex

すべてのデータ・ロード・エラーを ppMbrError に取得して、UTF-8 でエンコードされたファイルからアクティブなデータベースに更新指定を送信します。

構文

```
ESS_FUNC_M EssUpdateUtf8Ex (
hDestCtx, hSrcCtx, AppName, DbName, FileName, Store, Unlock, ppMbrError
);
```

パラメータ	データ型	説明
hDestCtx	ESS_HCTX_T	サーバー上のターゲット・データベースの API コンテキスト・ハンドル。
hSrcCtx	ESS_HCTX_T	更新ファイルの場所に対する API コンテキスト・ハンドル。更新ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	ESS_STR_T	更新ファイルの場所のアプリケーション名。
DbName	ESS_STR_T	更新ファイルの場所のデータベース名。
FileName	ESS_STR_T	更新指定ファイル名。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_UNKNOWN -

備考

- 更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

関連トピック

- EssUpdateEx
- EssUpdateFileASO
- EssUpdateFileASOEx
- EssUpdateFileEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUTF8ASO
- EssUpdateUtf8Ex

EssUpdateUtf8Ex

アクティブなデータベースに更新指定を UTF-8 でエンコードされた単一文字列として送信します。

構文

```
ESS_FUNC_M  EssUpdateUtf8Ex (
hCtx, Store, Unlock, UpdtSpec, ppMbrError
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
Store	ESS_BOOL_T	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	ESS_BOOL_T	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
UpdtSpec	ESS_STR_T	単一文字列としての更新指定。

パラメータ	データ型	説明
ppMbrError	ESS_PPMBRERR_T	ESS_MBRERR_T に含まれるエラーのリンク・リストへのポインタ。考えられるエラーは次のとおりです: ● ESS_MBRERR_BADDATA - データ列に無効なメンバー[membername]があります。 ● ESS_MBRERR_DBACCESS - このデータベースでのロックの実行に必要なアクセス権がありません。 ● ESS_MBRERR_DUPLICATE - データ・レコードの同一次元に重複メンバーがあります。[number]レコードが完了しました。 ● ESS_MBRERR_ERRORLOAD - アイテム/レコード[number]でのデータロードができません。 ● ESS_MBRERR_UNKNOWN -

備考

- 更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。
- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードおよびエラーの原因となったレコードが戻されます。

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

関連トピック

- EssUpdateEx
- EssUpdateFileASO
- EssUpdateFileASOEx
- EssUpdateFileEx
- EssUpdateFileUTF8ASOEx
- EssUpdateFileUtf8Ex
- EssUpdateFileUTF8ASO

EssValidateDB

データの整合性のためにデータベースを検証します。

構文

```
ESS_FUNC_M EssValidateDB (
hCtx, DbName, FileName
```

```
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
DbName	ESS_STR_T	データベース名。必須で、NULL は指定できません。
FileName	ESS_STR_T	エラー・ログ・ファイル。サーバー上の app¥db に保存されます。必須。

備考

- この関数は、検証確認を実行してデータベースの整合性を検証します。
- この呼出しの前に、EssSetActive を呼び出します。
- この関数は非同期なので、検証プロセスが終了するまで EssGetProcessState を続けて呼び出す必要があります。
- この関数によって現在のデータベースが検証されます。この関数を呼び出す前に、データベースを選択する必要があります。
- この関数は、各ブロックのデータ整合性を確認します。最上位から最下位まで読み取り、検証プロセスでデータベース全体を検証し、ブロック、セクション、ブロック・タイプ、ブロック長および浮動小数点数の有効性を確認します。
- この関数によってブロックと不正ブロックに関する情報がログ・ファイルに書き込まれます。
- この関数によって整合性エラーが検出されると、検証プロセス・エラー・メッセージがテキスト・フォーマットのログ・ファイルに書き込まれます。ファイルのデフォルト場所は、たとえば%ARBORPATH%\APP\DB\VALIDATE.LST のような application\database ディレクトリです
- Essbase インデックスには各データ・ブロックのインデックスが含まれています。すべての読取り操作について、この関数は自動的にインデックス・ページ内のインデックス・キーをそれに対応するデータ・ブロックのインデックス・キーと比較し、ブロックのその他のヘッダー情報を確認します。不一致がある場合、この関数はエラー・メッセージを表示し、データベース全体を確認するまで、処理を続行します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_VOID_T
ESS_ValidateDB (ESS_HCTX_T hCtx)
{
  ESS_FUNC_M     sts = ESS_STS_NOERR;
  ESS_STR_T     DbName;
  ESS_STR_T     FileName;
```

```
 ESS_PROCSTATE_T  pState;

 DbName  = "Basic";
 FileName =
  "D:\\AnalyticServices\\app\\sample\\basic\\Validate.lst";

 sts = EssValidateDB (hCtx, DbName, FileName);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts && (pState.State !=
      ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
 }
}
```

関連トピック

- EssSetActive
- EssGetProcessState

EssValidateHCtx

特定の API コンテキスト・ハンドル(hCtx)を検証します。

構文

```
ESS_FUNC_M EssValidateHCtx (
hCtx
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	検証する API コンテキスト・ハンドル。

備考

この関数を待機期間延長後に使用すると、プログラムのコンテキスト・ハンドルがサーバーによって認識される状態を確保できます。

戻り値

この関数はコンテキスト・ハンドルが有効な場合は 0 を戻し、それ以外の場合は無効なコンテキスト・ハンドルを示すエラー・コードを戻します。無効なコンテキスト・ハンドルに対して考えられる理由には、ログインがタイムアウトした、またはユーザーがスーパーバイザによって明示的にログアウトされたなどがあります。

アクセス

この関数には、特別なアクセス権は必要ありません。

例

```
    #include <essapi.h>

char sApplication[] = "accept";
char sDbName[] = "basic";
char sFilename[] = "basic";
char SvrName[] = "local";
char User[] = "test";
char Password[] = "testing";

ESS_HINST_T hInst;
ESS_HCTX_T hCtx;
FILE *fpOutfile;

void ESS_Init()
{
  ESS_STS_T  sts;
  ESS_INIT_T InitStruct = { ESS_API_VERSION,  /* This should be set to
ESS_API_VERSION */
              NULL,       /* void pointer to user's message context */
              0L,        /* max number of context handles required */
              255,       /* max size of buffer that can be allocated*/
              NULL,       /* local path to use for file operations */
              NULL,       /* full path name of message database file */
              NULL,       /* user-defined memory allocation function */
              NULL,       /* user-defined memory reallocation function*/
              NULL,       /* user-defined memory free function */
              NULL,       /* user-defined message callback function */
              NULL,       /* user-defined help file path */
              0L        /* reserved for internal use */
             };

 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {    fprintf(stdout, "EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
 fprintf(stdout, "EssInit sts: %ld\n", sts);
}

void ESS_Login ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssLogin (hInst, SvrName, User, Password, &Items, &pAppsDbs, &hCtx);
 printf("EssLogin sts: %ld\r\n", sts);
}

void ESS_Term()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
 {
  /* error terminating API */
```

```
   exit((ESS_USHORT_T) sts);
  }
  fprintf(stdout, "EssTerm sts: %ld\r\n", sts);
}

void ESS_Logout()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  sts = EssLogout (hCtx);
  fprintf(stdout, "\n\nEssLogout sts: %ld\n",sts);
}

void ESS_SetActive()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_ACCESS_T Access;
  ESS_STR_T AppName;
  ESS_STR_T DbName;
  AppName = sApplication;
  DbName = sDbName;
  sts = EssSetActive(hCtx, AppName, DbName, &Access);
  fprintf(stdout, "EssSetActive sts: %ld\r\n",sts);
}
/*****************************************************/
/*************** MAIN FUNCTION ***********************/
void main(int argc, char ** argv)
{
    ESS_STS_T   sts;
    ESS_Init();
    ESS_Login();
    ESS_SetActive();
    /* Do something else, not related to Essbase*/
    sts = EssValidateHCtx (hCtx);
    if (sts) {
        ESS_Login()  ;
        ESS_SetActive();
    }
     /* Do the actual processing now */
      EssClearActive(hCtx);
      ESS_Logout();
      ESS_Term();
}
```

関連トピック

- EssLogin
- EssAutoLogin
- EssTerm

EssVerifyFilter

指定したデータベースに照らしあわせて、一連のフィルタ行の文字列の構文を確認します。

構文

```
ESS_FUNC_M  EssVerifyFilter (
hCtx, AppName, DbName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AppName	ESS_STR_T	アプリケーション名。
DbName	ESS_STR_T	データベース名。

備考

この呼出しの後に EssVerifyFilterRow を続けて呼び出して、フィルタのすべての行を確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    ESS_VOID_T
ESS_VerifyFilter (ESS_HCTX_T hCtx)
{
 ESS_FUNC_M    sts = ESS_STS_NOERR;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_USHORT_T   Count = 4;
 ESS_STR_T    RowString[4];
 ESS_USHORT_T   ind;

 AppName = "Sample";
 DbName = "Basic";
 /* Initialize Filter Row */

 RowString[0] = "@IDESCENDANTS(Scenario)";
 RowString[1] = "@IDESCENDANTS(Product)";
 RowString[2] = "Qtr1, @IDESCENDANTS(\"Colas\")";
 RowString[3] = "";

 /* Verify Filter */

 sts = EssVerifyFilter(hCtx, AppName, DbName);

 /* Verify Count Filter Rows */

 if(!sts)
 {
```

```
   for (ind = 0; ind < Count; ind++)
    sts = EssVerifyFilterRow(hCtx,
       RowString[ind]);
  }
}
```

関連トピック

- EssGetFilter
- EssVerifyFilterRow

EssVerifyFilterRow

指定したデータベースに照らしあわせて、単一のフィルタ行の文字列の構文を確認します。

構文

```
ESS_FUNC_M  EssVerifyFilterRow (
hCtx, RowString
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
RowString	ESS_STR_T	フィルタ行文字列。

備考

この関数は、EssVerifyFilter を呼び出した後に、フィルタの各行について 1 回ずつ繰り返し呼び出し、行リストを NULL 行文字列ポインタで終了する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対してデータベース・デザイナ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

例

EssVerifyFilter の例を参照してください。

関連トピック

- EssGetFilter
- EssVerifyFilter

EssVerifyFormula

指定した式の構文を確認します。この関数は EssOtlVerifyFormula から呼び出され、戻されたエラーの詳細情報を提供します。

構文

```
ESS_FUNC_M  EssVerifyFormula (
hCtx, FormulaName
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
FormulaName	ESS_STR_T	確認する式の名前。

備考

この関数は直接呼び出されるものではありません。そのかわり、対応するアウトライン API 関数 EssOtlVerifyFormula を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外は、エラー番号が戻されます。

関連トピック

- EssOtlVerifyOutline
- EssOtlVerifyOutlineEx
- EssOtlVerifyFormula

EssVerifyRulesFile

指定したルール・ファイルの構文を確認します。

構文

```
ESS_FUNC_M  EssVerifyRulesFile (
hCtx, ruleFileName, pNmColumns, ppColumnErrors
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
ruleFileName	ESS_STR_T	確認するルール・ファイル名。
pNmColumns	ESS_PULONG_T	ルール・ファイルの列の数へのポインタ。
ppColumnErrors	ESS_ULONG_T	見つかったエラーの配列へのポインタ。

備考

- この関数を実行するには、特定のデータベースがアクティブになっている必要があります。つまり、EssSetActive()を実行しておく必要があります。
- この関数は、ルール・ファイルをサーバーに配置した後で実行します。
- ルール・ファイルの各列の配列 ppColumnErrors に 1 つの値があります。配列の n 番目の値は、ルール・ファイルの n 番目の列にあるエラーに対応します。各エラー値は、0 または次のエラー・コードを論理 OR で組み合せることができます。

エラー・コード	意味
DAT_VERIFY_INVALIDMBR	フィールド名に不明なメンバーがあります(またはメンバーが存在しません)。
DAT_VERIFY_INVALIDHDR	ヘッダーに不明なメンバーがあります。
DAT_VERIFY_SAMENAME	このフィールドには、別のフィールドと同じ名前が付いています。
DAT_VERIFY_DIMUSED	別のフィールドまたはヘッダーには、次元名が使用されます。
DAT_VERIFY_MBRUSED	このフィールドで組合せの一部として使用されているメンバー名は、フィールド名で単一のメンバー名として使用されています。
DAT_VERIFY_ DIMINCROSSDIM	次元名が、フィールド名で次元間の参照として使用されています。
DAT_VERIFY_DATAFIELD	データ・フィールド属性を持つことができるフィールドは 1 つのみです。
DAT_VERIFY_SIGNFLIPDIM	符号反転確認に使用される次元が、関連付けられたアウトラインにありません。
DAT_VERIFY_DUPINHEADER	このフィールド名は、ヘッダー定義でも定義されています。
DAT_VERIFY_DATEANDDATA	フィールドは、データ・フィールドまたは日付フィールドとして指定できますが、両方に指定できません。
DAT_VERIFY_ DATEFIELDNAME	日付フィールドのフィールド名は日付次元の名前である必要があります。
DAT_VERIFY_DATEFORMAT	この日付列には未承認の日付フォーマットがあります。

戻り値

正常終了の場合、ルール・ファイル内の列数が pNmColumns に、見つかったエラーの配列が ppColumnErrors に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
   {
ESS_STS_T    sts = ESS_STS_NOERR;
ESS_ULONG_T   numColumns = 0, i;
ESS_PULONG_T  pColumnErrors = NULL;

sts = EssVerifyRulesFile(hCtx, "rule_file", &numColumns, &pColumnErrors);
```

```
 if(!sts)
 {
  if(numColumns && pColumnErrors)
  {
   printf("NumColumns: %d\n", numColumns);
   for(i=0; i<numColumns; i++)
   {
    printf("Column[%d]:\n", i+1);
    if( pColumnErrors[i] == 0 )
     printf("   No error\n");
    else
    {
     if( pColumnErrors[i] & DAT_VERIFY_INVALIDMBR )
      printf("   There is an unknown member (or no member) in the field name.\n");
     if( pColumnErrors[i] & DAT_VERIFY_INVALIDHDR )
      printf("   There is an unknown member in the header.\n");
     if( pColumnErrors[i] & DAT_VERIFY_SAMENAME )
      printf("   This field has the same field name as another field.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DIMUSED )
      printf("   The dimension name is used in another field name or in the header.
\n");
     if( pColumnErrors[i] & DAT_VERIFY_MBRUSED )
      printf("   A member name used as part of a combination in this field is used
as a single
              member name in another field.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DIMINCROSSDIM )
      printf("   A dimension name is used in a cross-dimensional reference in the
field name.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DATAFIELD )
      printf("   Only one field can have the Data Field attribute.\n");
     if( pColumnErrors[i] & DAT_VERIFY_SIGNFLIPDIM )
      printf("   The dimension used for Sign Flip checking is not in the associated
outline.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DUPINHEADER )
      printf("   This field name is also defined in the header definition.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DATEANDDATA )
      printf("   A field may be designated a Data Field or a Date Field, but not
both.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DATEFIELDNAME )
      printf("   The field name of a date field must be the name of a date
dimension.\n");
     if( pColumnErrors[i] & DAT_VERIFY_DATEFORMAT )
      printf("   There is an unrecognized date format for this date column.\n");
    }
   }
   EssFree(hInst, pColumnErrors);
  }
 }
}
```

関連トピック

- EssVerifyFormula
- EssOtlVerifyFormula
- EssOtlVerifyOutlineEx

EssWriteToLogFile

Essbase サーバー・ログ・ファイル(essbase.log)またはアプリケーション・ログ・ファイル(appname.log)にメッセージを書き込みます。

構文

```
ESS_FUNC_M  EssWriteToLogFile (
hCtx, AgentLog, Message
);
```

パラメータ　データ型　説明

hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。
AgentLog	ESS_BOOL_T	TRUE の場合、メッセージは Essbase サーバー・ログ・ファイル essbase.log に書き込まれます。FALSE の場合、メッセージはアプリケーション・ログ・ファイル appname.log に書き込まれます。
Message	ESS_STR_T	Essbase サーバー・ログ・ファイル(essbase.log)またはアプリケーション・ログ・ファイル(appname.log)に記録されるメッセージ。

備考

- メッセージ・ログを表示するには、EssGetLogFile を使用します。
- essbase.log および appname.log の場所は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

呼出し元は、指定したアプリケーションに対してスーパーバイザ権限(ESS_ACCESS_SUPER)を持っている必要があります。

例

```
    ESS_FUNC_M ESS_WriteToLogFile (ESS_HCTX_T hCtx)
{
 ESS_STR_T   Message = NULL;
 ESS_FUNC_M   sts = ESS_STS_NOERR;

 Message = "Received login request";

 /*
  * Writes the message (Received login request) to the Agent log file.
  */
 sts = EssWriteToLogFile(hCtx, ESS_TRUE, Message);
 return(sts);
}
```

関連トピック

- EssDeleteLogFile
- EssGetLogFile
- EssLogSize

第 III 部
C のアウトライン API

C のアウトライン API の内容：

9 CのアウトラインAPIの使用

この章の内容

C アウトライン API の概要

アウトライン API は Essbase アウトラインをカスタム・アプリケーション内から作成、維持、操作する一連の関数です。アウトライン API を使用して、Administration Services のアウトライン・エディタを使用する場合と同様にしてコード内からデータベース・アウトラインを操作できます。

アウトライン API は Essbase API の重要な部分で、C および Visual Basic のインタフェースを備えています。アウトライン API は Essbase API とともに使用され、サーバー接続を必要とします。

C のアウトライン API のエラー処理

アウトライン API 関数が正常終了の場合は 0 が戻されます。失敗すると C の場合は `esserror.h` で定義されたエラー・ステータスの値が、Visual Basic の場合は `esberror.bas` で定義されたエラー・ステータスの値が戻されます。メイン API の関数はエラー・メッセージ・コールバック・ルーチンを使用して、メッセージ・ハンドラにエラー番号を戻します。ハンドラは `essbase.mdb` メッセージ・データベースを使用してエラー・メッセージを特定し、ユーザーにエラー・メッセージを表示します。

アウトライン API 関数は通常、エラー・ステータスを戻す際にエラー・メッセージ・コールバック・ルーチンを使用しません。エラー・コールバック・ルーチンは次の場合に呼び出されます:

- ネットワークを使用する関数(**EsxOtlOpenOutline**()、**EsxOtlWriteOutline**()および **EsxOtlRestructure**())を呼び出して、アウトラインに関係のないアクションでエラーが発生した場合。

- アウトライン API に渡されたときルーチン確認で NULL が検出され、API_NULL_ARG が戻された場合。
- 不良なアウトライン・ハンドル(HOUTLINE)が、アウトライン・ハンドルを必要とする呼出しに渡され、OTLAPI_BAD_HOUTLINE が戻された場合。

C のアウトライン API サーバー・アウトライン・クエリー

いくつかの関数はアウトライン API へのクエリー・インタフェースをサポートしているため、アウトラインをサーバーからダウンロードしてメモリーに完全に読み込む必要がありません。この種のアウトライン API 関数はサーバー・アウトラインのみをサポートしています。アウトラインを開く前に、ユーザーはサーバーにログインして、有効な Essbase ログイン・コンテキストを設定する必要があります。

これらの関数のエラー処理は、標準的な API エラー処理メカニズムで行われます。したがって、エラーの際は、呼出し元が **EsxInit()**から指定したメッセージ・コールバックが呼び出されます。

このメカニズムを次に説明します:

1. プログラマは **EsxInit()**と **EsxLogin()**を呼び出して、通常どおり API を初期化します。
2. プログラマは **EsxOtlOpenOutlineQuery()**を呼び出してサーバーからアウトラインを開き、ある程度の初期情報を取得します。その際、**ESX_OUTLINEINFO_T** 構造体内のすべての情報と、**ESX_MBRINFO_T** 構造体を含む各次元における **ESX_OTLMBR_T** 内部構造体内のすべての関連情報が、サーバーから取得されます。
3. 呼出し元はメンバーに関する情報を入手する必要があるため、該当するフラグを設定して **EsxOtlQueryMembers()**を呼び出し、メンバー・ハンドルの配列を取り戻します。**EsxOtlQueryMembers()**の呼出しによって内部構造体である **ESX_OTLMBR_T** に関連するすべての情報が戻されます。その結果、ユーザーは戻されたメンバー・ハンドルのいずれかを渡して、特定のメンバーに関連するいずれかの **EsxOtlGetXxxx()**を呼び出すことができます。アウトラインをクエリー・モードで開いている場合にサポートされている呼出しの詳細は、**EsxOtlQueryMembers()**呼出しのコメント部分を参照してください。
4. **EsxOtlQueryMembers()**からデータが戻され、呼出しが完了したら、**EssOtlFreeMembers()**または **EsbOtlFreeMember()**を呼び出してメンバーの配列を解放します。
5. 完了したら、呼出し元は **EsxOtlCloseOutline()**を呼び出して、内部のデータ構造体をクリーン・アップします。
6. 呼出し元は **EsxLogout()**と **EsxTerm()**を呼び出して、通常どおり API を終了します。

C のアウトライン API のアウトライン確認

アウトライン API によって、呼出し元が不正なアウトラインを作成するのを防止できます。アウトラインを確認するには、**EsxOtlVerifyOutline()**関数を使用して、サーバーに保存する前に確認します。アウトライン API は、アウトラインがサーバーに書き込まれる際、**EsxOtlVerifyOutline()**がまだ呼び出されていない場合には、自動的に呼び出します。

アウトライン API の各関数は、呼出し元による処理によって不正なアウトラインが生成されないか検証します。たとえば **EsxOtlRenameMember()**は新しいメンバー名を確認して、有効で、アウトラインに既存のものでないことを確認します。この種の自動検証にはいくつかの例外があります:

- **EsxOtlOpenOutline()**では、呼出し元が前に作成された不正なアウトラインを読み取ることができます。Administration Services のアウトライン・エディタで不正なアウトラインをローカル・ファイルに保存できるため、このアウトラインは不正な状態です。**EsxOtlVerifyOutline()**を呼び出すと、既存のエラーが検出されます。また、アウトラインが不正なものとして起動した場合は、処理中に個別の操作が不正になります。
- **EsxOtlDeleteMember()**および **EsxOtlDeleteDimension()**は、削除されたメンバーを含む別名の組合せを確認しません。この状態は **EsxOtlVerifyOutline()**によって検出されます。
- **EsxOtlSetMemberFormula()**では不正な式を入力でき、**EsxOtlVerifyOutline()**ではメンバー式を確認しません。不正なメンバー式があると、再構築中に障害が発生します。**EsxGetProcessState()**は、サーバーから戻されたエラー・メッセージを表示します。

C のアウトライン API メモリー割当て

Essbase API では、**EsxAlloc()**、**EsxRealloc()**、**EsxFree()**という、一連のメモリー管理関数を備えています。これらの関数とすべての内部 API メモリー割当ては、**ESX_INIT_T** 初期化構造体の `AllocFunc`、`ReallocFunc`、および `FreeFunc` フィールドが指すメモリー割当てルーチンを呼び出します。

ユーザー独自のメモリー割当てルーチンを使用している場合、多数の小規模なメモリー・バッファの割当てを処理できるようなメモリー割当ての仕組みを使用していることを確認してください。

C のアウトライン API のセキュリティ要件

アウトライン API を使用して、アウトラインの作成、編集および削除ができるので、アウトライン API を使用するアプリケーションを作成するときは、セキュリティの問題に注意する必要があります。これらの問題は、セッションの間にアウトラインを作成、編集または保存するプログラムにのみ影響します。

Administration Services のアウトライン・エディタによってアウトラインを操作するには、アプリケーション・マネージャ以上の権限を持っている必要があります。また、実行の間にアウトライン API を使用するプログラムを使用するためにも、

これらの権限を必要とします。これらの権限を持っていない場合、サーバーからアウトラインを読取りまたは書込みするアウトライン API 呼出しは機能しません。セキュリティと権限のレベルに関する詳細情報は、『Oracle Essbase データベース管理者ガイド』を参照してください。

たとえば、ユーザーがセッションの間に複数の「仮定」状況を調査できる、新しい EIS エンドユーザー・アプリケーションを書いているとします。これを行うために、プログラムは、セッションの間に複数の Essbase データベースを動的に作成します。これらのデータベース(およびそのアウトライン)は、一時的なものであり、セッションが終了した後に保存されません。この状況にアプローチできる方法はいくつかあります:

- セッションの間にアプリケーションおよび複数のデータベースをユーザーが作成できるようにする場合、ユーザーに**アプリケーションの作成/削除**権限を与えます。Essbase 管理者は、プログラムを実行する前に、この権限を割り当てる必要があります。これは、Essbase では比較的高い権限レベルです。しかし、ユーザーが他のプログラムにアクセスできない場合、システムのセキュリティ全体への影響はほとんどありません。
- 同時に利用可能な複数のデータベースを必要としない場合、Essbase 管理者に、プログラムのインストールの間に一時アプリケーションおよびデータベースを作成してもらうことができます。プログラム自体が一時データベースを操作し、各"what-if"状況で新しいデータベースを作成する必要がありません。

別の方法では、ユーザーは、より低く制限されたデータベース・マネージャ権限のみを必要とします。Essbase 管理者に、一時アプリケーションおよびデータベースのためにのみ、データベース・マネージャ権限を持つ特別なグループを設定してもらうことができます。ユーザーをそのグループに割り当てることができます。ユーザーは、システムへの他のアクセスでは、通常のユーザー権限に戻ります。この方法では、セキュリティの露出は少なくなりますが、プログラムを実行する前に、より多くの設定が必要です。

C のアウトライン API 関数の呼出し順序

アウトライン API を使用する場合、一部の API 関数は他の関数より先に呼び出す必要があります。基本的な呼出し順序は次のとおりです:

1. **EsxInit()**は、他の API 関数よりも先に呼び出します。

   この API はインスタンス・ハンドルを戻します。

2. サーバーにログ・オンするには、**EsxLogin()**または **EsxAutoLogin()**を呼び出します。

   この API はコンテキスト・ハンドルを戻します。

3. アウトラインを開く、または作成するには、**EsxOtlOpenOutline()**または **EsxOtlNewOutline()**を呼び出します。

   この API はアウトライン・ハンドルを戻します。

4. 現在のアウトラインをサーバーに書き込むには、**EsxOtlWriteOutline()**を呼び出します。**EsxOtlVerifyOutline()**は、この関数より先に呼び出していないかぎり、アウトラインの保存前に API によって自動的に呼び出されます。
5. アウトラインに対して行った変更に基づいてデータベースを再構築するには、**EsxOtlRestructure()**を呼び出します。
6. アウトラインを開いたときにロックされたアウトライン・オブジェクトのロックを解除するには、**EsxUnlockObject()**を呼び出します。
7. アウトラインに関連付けられているリソースを解放するには、**EsxOtlCloseOutline()**を呼び出します。
8. サーバーからログアウトするには、**EsxLogout()**を呼び出します。

   これでコンテキスト・ハンドルが無効になります。
9. セッションを終了するには、**EsxTerm()**を呼び出します。

   これでインスタンス・ハンドルが無効になります。

C のアウトライン API タスクの一般的な順序

単純なアウトライン API アプリケーションの、一般的な操作順序を次に示します。

1. **ESX_INIT_T** 構造体を作成し、初期化します。
2. **EsxInit()**を呼び出してアウトライン API を初期化します。
3. ローカルの静的な構造体またはグローバル構造体を割り当てます。
4. **EsxLogin()**または **EsxAutoLogin()**を呼び出して、必要なサーバーにログインします。
5. **ESX_OUTLINEINFO_T** 構造体を作成し、初期化します(新規のアウトラインに対してのみ)。
6. **EsxOtlOpenOutline()**または **EsxOtlNewOutline()**を呼び出して、既存のアウトラインを開くか、新規のアウトラインを作成します。
7. アウトラインの処理を行います。
8. **EsxOtlVerifyOutline()**を呼び出してアウトラインを確認します。
9. **EsxOtlWriteOutline()**を呼び出して、確認済のアウトラインをサーバーに書き込みます。

   .OTN という拡張子でアウトラインが保存されます。
10. **EsxOtlRestructure()**を呼び出して、データベースを再構築します。

    .OTN ファイルが.OTL ファイルに変更されます。これは非同期の関数呼出しなので、プロセスが完了するまで **EsxGetProcessState()**を呼び出す必要があります。
11. **EsxUnlockObject()**を呼び出して、アウトラインのロックを解除します(オープン時にロックされた場合)。
12. **EsxOtlCloseOutline()**を呼び出して、アウトラインに関連付けられているすべての情報を解放します。

13. **EsxLogout()**を呼び出して、サーバーからログアウトします。
14. ローカルの静的な構造体またはグローバル構造体を解放します。
15. **EsxTerm()**を呼び出して、API を終了します。

10 CのアウトラインAPIの宣言

この章の内容

C のアウトライン API のエラー戻り値

表 7 は、アウトライン API 呼出しが失敗したときに戻されたエラー・ステータス定数を説明しています。これらの値は、アウトライン API C 言語ヘッダー・ファイル esserror.h で定義されています。

完全なリストについては、esserror.h を参照してください。

表 7　C のアウトライン API のエラー戻り値

値	説明
OTLAPI_BAD_ALIASTABLE	別名テーブルが正しくありません
OTLAPI_BAD_CONSOL	集計のタイプが正しくありません(+、- など)
OTLAPI_BAD_GENLEVELNAME	世代名またはレベル名が正しくありません
OTLAPI_BAD_HOUTLINE	EssOtl...関数に渡されたアウトライン・ハンドルが無効です
OTLAPI_BAD_MBRNAME	メンバー名が無効です

値	説明
OTLAPI_BAD_MEMBER	メンバーのハンドルが無効です
OTLAPI_BAD_MOVE	メンバーの移動が不正です。メンバーを子孫に移動できません。
OTLAPI_BAD_OBJTYPE	オブジェクト・タイプが不正です
OTLAPI_BAD_OUTLINETYPE	アウトライン型が無効です
OTLAPI_BAD_PERSPECTIVE2	パースペクティブが無効です
OTLAPI_BAD_RENAMESHARE	共有メンバー名は変更できません
OTLAPI_BAD_RESTRUCTTYPE	再構築のタイプが正しくありません
OTLAPI_BAD_SCA_VALIDITYSET_TYPE	パースペクティブ/妥当性セットはこの妥当性セット・タイプをサポートしていません
OTLAPI_BAD_SMARTLISTNAME	テキスト・リスト名が無効です
OTLAPI_BAD_SORT COMPAREFUNC	ソート比較関数が無効です
OTLAPI_BAD_SORTTYPE	ソート型が無効です
OTLAPI_BAD_TRANSTYPE	トランザクション作成時のトランザクションのタイプが不明です(内部エラー)
OTLAPI_BAD_USERATTR	ユーザー属性が無効です
OTLAPI_CUR_NOACCOUNTS	会計次元がありません。通貨データベースを作成するには、会計次元が必要です。
OTLAPI_CUR_NOCOUNTRY	国次元がありません。通貨データベースを作成するには、国次元が必要です。
OTLAPI_CUR_NOTIME	時間次元がありません。通貨データベースを作成するには、時間次元が必要です。
OTLAPI_ERR_ADDDELETEDIMDYNAMICCALC	データを保管するメンバーのタイプが動的計算になっています
OTLAPI_ERR_ADDNAMEUSED	メンバー名はすでに使用されています(追加操作)
OTLAPI_ERR_ALIASTABLEEXISTS	別名テーブル名はすでに存在します
OTLAPI_ERR_ALIASLANGUAGE_UNAVAILABLE	別名テーブルの言語は 11.1.2.0.00 より前のアウトライン・バージョンでは使用できません。
OTLAPI_ERR_ALIASTABLENAME	別名テーブル名が正しくありません
OTLAPI_ERR_ALREADY CURRENCY	このアウトラインは通貨アウトラインです。通貨アウトラインを作成しようとしていますが、初期アウトラインがすでに通貨アウトラインです。
OUTAPI_ERR_ASO_ COMPRESSIONMUSTBEDYNAMIC	集約ストレージ・アウトラインは、圧縮次元が単一の動的階層であることを必要とします
OUTAPI_ERR_ASO_ DIFFERENTNUMBEROFSHARES	このプロトタイプは、共有メンバーの数が前の兄弟と同数である必要があります

値	説明
OUTAPI_ERR_ASO_ SHAREDMEMBERSNOTINSAMEORDER	このプロトタイプは、各共有メンバーが前の兄弟の共有メンバーの次の兄弟である必要があります
OTLAPI_ERR_ATTR_ATTACHED_WRONGLEVEL	この属性は、少なくとも 1 度不正なレベルで添付されています
OTLAPI_ERR_ATTRMBR_ALREADYASSOCIATED	基本メンバーに同じ次元の属性メンバーがすでに関連付けられています
OTLAPI_ERR_BADDIM	次元引数が無効です
OTLAPI_ERR_BADHIER	階層タイプが無効です
OTLAPI_ERR_BADHIER_TOP	階層メンバー指定が無効です。メンバーは世代 1 または 2 にある必要があります
OTLAPI_ERR_BADSHARE	共有値が正しくありません
OTLAPI_ERR_BADSKIP	タイム・バランス・スキップ値が正しくありません
OTLAPI_ERR_BADSTORAGE	次元ストレージ値が正しくありません
OTLAPI_ERR_BADSTORAGECATEGORY	ストレージ・カテゴリが正しくありません
OTLAPI_ERR_BADTIMEBAL	タイム・バランス値が正しくありません
OTLAPI_ERR_BSO_SOLVEORDER	メンバー・タイプで使用可能になっていないブロック・ストレージ・アウトラインには、解決順を設定できません
OTLAPI_ERR_CANTIDENTIFYMBR_ DUPLICATEDNAME	名前が重複しているためメンバーを区別できません
OTLAPI_ERR_CNTS_INDEP_LAST	独立次元リストには、連続する独立次元が最後に並んでいる必要があります
OTLAPI_ERR_CONFIGTOOMANYDIMS	次元が多すぎるため、自動的に構成できません
OTLAPI_ERR_COPYALIASTABLE	ソースと宛先のテーブルが同じです
OTLAPI_ERR_CREATETEMP	一時ファイル名を作成できません。読取り専用ドライブに作成しようとしている可能性があります。アウトラインをサーバーから開くかサーバーへ書き込むたびに、クライアントに一時ファイルが作成されます。
OTLAPI_ERR_CURTOOMANYDIMS	通貨アウトライン内の次元が多すぎます。1 つの通貨アウトラインに指定できる次元は、最大 4 つです。
OTLAPI_ERR_DELETEDEFALIAS	デフォルトの別名テーブルを削除できません
OTLAPI_ERR_DISCRETE_DIFFERENT	独立範囲には同じ個別の開始メンバーおよび終了メンバーが必要です
OTLAPI_ERR_DISCRETE_OR_CNTS	独立次元タイプは、個別または連続のどちらかです
OTLAPI_ERR_DUP_LANGCODE	言語コードが、同じデータベース内の異なる別名テーブルに割り当てられています
OTLAPI_ERR_DUPLICATEALIAS	別名が重複しています
OTLAPI_ERR_DUPLICATENAME	メンバー名が重複しています

値	説明
OTLAPI_ERR_DUPGENLEVNAME	世代名またはレベル名と重複するメンバー名または別名の追加、変更または設定はできません
OTLAPI_ERR_EXPORT_INCORRECT_FLAGS	無効なエクスポート・フラグが存在します。抽出をツリーおよび別名テーブルに制限するためにエクスポートを使用できません
OTLAPI_ERR_EXPORT_INVALID_ALIASTABLE	エクスポート・オプションに無効な別名テーブルが指定されています
OTLAPI_ERR_EXPORT_INVALID_DIMLIST	次元の数、またはエクスポート・オプションに指定された次元リストが無効です
OTLAPI_ERR_EXPORT_INVALIDDIM_DIMLIST	エクスポート・オプションの次元リストに指定された次元名が無効です
OTLAPI_ERR_EXPORT_INVALID_VERSION	このエクスポートのバージョンは無効です。有効なエクスポートのバージョンを入力してください
OTLAPI_ERR_EXPORT_UNABLE_FILE	ファイルを開いてアウトラインをエクスポートできません
OTLAPI_ERR_EXPORT_UNABLE_PROCESS	サポートしていないアウトライン・タイプのため、アウトラインを処理できません
OTLAPI_ERR_FAILED_GET_ALIASNAMES	別名識別子検索が失敗したため、すべての別名を取得できませんでした
OTLAPI_ERR_FEATURE_UNAVAILABLE	このアウトライン・バージョンではこの機能を使用できません。アウトラインを最初に移行してください
OTLAPI_ERR_FILEIO	ファイルの読取りまたはファイルへの書込みができませんでした
OTLAPI_ERR_FILEOPEN	ファイルを開けませんでした
OTLAPI_ERR_FORMATSTRING_MISMATCH	暗黙の共有メンバーまたはラベルのみのメンバーに、元のメンバーとは異なるフォーマット文字列があります。元のメンバーのフォーマットが適用されます
OTLAPI_ERR_FORMATSTRING_NOT_MEMBTYPE_ENABLED	フォーマット文字列を使用するには、アウトラインのメンバー・タイプを使用可能にする必要があります。このアウトラインではメンバー・タイプは使用可能ではありません
OTLAPI_ERR_FORMATSTRINGTOOLONG	フォーマット文字列が単一ロケール構成には長すぎます
OTLAPI_ERR_FUNCTION_OBSOLETE	関数は廃止されています
OTLAPI_ERR_GENLEVELEXISTS	世代またはレベルには、すでに名前が付いています
OTLAPI_ERR_GENLEVELNAMEEXISTS	世代名またはレベル名がすでに存在しています
OTLAPI_ERR_GENLEVELVALUE	世代値またはレベル値が正しくありません
OTLAPI_ERR_GENLEVNAMEMBR	メンバー名または別名と重複する世代名またはレベル名は追加できません
OTLAPI_ERR_ILLEGALALIAS STRING	別名の組合せメンバーが正しくありません
OTLAPI_ERR_ILLEGALCOMBOALIAS	別名の組合せが正しくありません
OTLAPI_ERR_ILLEGALCURRENCY	通貨メンバーが正しくありません

値	説明
OTLAPI_ERR_ILLEGALDEFALIAS	デフォルトの別名が正しくありません
OTLAPI_ERR_ILLEGALNAME	メンバー名が正しくありません
OTLAPI_ERR_ILLEGALTAG	次元タグ(カテゴリ)が正しくありません
OTLAPI_ERR_ILLEGALOPTION	ユーザーが無効なオプションを **EssOtlGetGenNames()**または **EssOtlGetLevelNames()**に対して渡した場合に発生します
OTLAPI_ERR_IMPLIED_SHARE_OLD_VERSION	アウトラインのバージョンが古すぎるため暗黙の共有を設定できません
OTLAPI_ERR_INCORRECT_MEMBERTYPE	メンバー・タイプは数値型または日付型である必要があります
OTLAPI_ERR_INVALID_SMARTLIST_HANDLE	無効なテキスト・リスト・ハンドルです
OTLAPI_ERR_INVALID_SMARTLIST_ IMPORTFILE	テキスト・リストをインポートするための入力ファイルが無効です
OTLAPI_ERR_INVALIDID_SMARTLIST_ IMPORTFILE	テキスト・リストのインポート・ファイルに含まれる ID が、無効であるか重複しています
OTLAPI_ERR_LANGCODE_TOOLONG	別名テーブルの言語コードが最大長を超えています
OTLAPI_ERR_LEAFLABEL	リーフ・メンバーがラベル・メンバーとして定義されています
OTLAPI_ERR_MAXALIASTABLES	別名テーブルの数が最大に達しました
OTLAPI_ERR_MEMBERCALC	メンバー式が正しくありません
OTLAPI_ERR_MEMBERTYPE_OFF	アウトラインのメンバー・タイプの使用可能な設定をオフにできません
OTLAPI_ERR_MBRCOMMENTEXLEN	拡張メンバー・コメントが長すぎます
OTLAPI_ERR_MISSINGTEXT_SMARTLIST_ IMPORTFILE	テキスト・リストのインポート・ファイルに含まれる ID のテキストが欠落しています
OTLAPI_ERR_MULT_DATE_DIMS	1 つのアウトラインに設定できるのは静的メンバーのデータ型を持つ、多くとも 1 つの次元のみです
OTLAPI_ERR_MULTIHIER_NOT_ENABLED	階層タイプを設定できません。複数の階層が次元に対して使用可能になっていません
OTLAPI_ERR_MULT_SMARTLIST_DIMS	1 つのアウトラインに設定できるのは静的メンバーのスマートリストを持つ、多くとも 1 つの次元のみです
OTLAPI_ERR_MUSTSAVE_BEFORE_EDIT	アウトラインを編集するには、一度保存して再度開く必要があります
OTLAPI_ERR_NOALIAS	このメンバーには、別名がありません
OTLAPI_ERR_NOALIASCODE	別名テーブルの言語コードの取得/設定はまだ実装されていません
OTLAPI_ERR_NOALIASCOMBO	別名の組合せがありません
OTLAPI_ERR_NOATTRONCOMPRESSEDDIM	属性は圧縮された次元で使用できません

値	説明
OTLAPI_ERR_NOFORMULA	このメンバーには、式がありません
OUTAPI_ERR_NOMEMBTYPE	このアウトライン・バージョンでは、型付きメンバーはサポートされていません
OTLAPI_ERR_NOSHAREPROTO	共有メンバーに実際のメンバーが付加されていません
OUTAPI_ERR_NOSMARTLISTS	このアウトライン・バージョンでは、テキスト・リストはサポートされていません
OTLAPI_ERR_NOTADIM	次元名が必要です
OTLAPI_ERR_NOT_A_TIME_MBR	無効な引数が渡されました。日時次元メンバーではありません
OTLAPI_ERR_NOT_LINKEDATTRIBUTEDIM	リンク属性次元のハンドルではありません
OTLAPI_ERR_NOT_MEMBTYPE_ENABLED	このアウトラインではメンバー・タイプは使用可能ではありません
OTLAPI_ERR_NOTIMEDIM	時間次元が定義されていません(時間次元がない場合、タイム・バランス操作が実行できません)
OTLAPI_ERR_NOTVERIFIED	アウトラインにエラーがあります(サーバーへの保存時)
OTLAPI_ERR_OBJ_NOTFOUND	オブジェクトが見つかりません
OTLAPI_ERR_OBJTYPE_NOTSUPPORTED	関数がサーバー側の編集モードでサポートされていません
OTLAPI_ERR_OPENMODE	この呼出しを行うために、ファイルが不適切なモードで開かれました。アウトラインを開くために **EssOtlOpenOutlineQuery()**を呼び出す場合、呼出しのすべてが機能するとはかぎりません。
OTLAPI_ERR_OTLDATEFORMAT	アウトライン・プロパティの日付フォーマットは無効です
OTLAPI_ERR_OTLSHARED_FORMAT	アウトライン・メンバーのフォーマット文字列は、共有メンバーに設定できません
OTLAPI_ERR_OTLSHARED_TYPE	アウトライン・メンバーのタイプは、共有メンバーに設定できません
OTLAPI_ERR_QUERYHINT_INVALIDARRAYSIZE	クエリー・ヒントの配列サイズが無効です
OTLAPI_ERR_RENAMEDEFALIAS	デフォルトの別名テーブル名を変更できません
OTLAPI_ERR_RENAMENAMEUSED	メンバー名はすでに使用されています(名前変更操作)
OTLAPI_ERR_SCA_NOT_ENABLED	このアウトラインでは、可変属性は使用可能ではありません
OTLAPI_ERR_SCA_UNAVAILABLE	可変属性機能は、このバージョンでは使用できません
OTLAPI_ERR_SHAREDMEMBERFORMULA	共有メンバーは式を持つことができません
OTLAPI_ERR_SHARENOTLEVEL0	共有メンバーのレベルが 0 になっていません(共有メンバーを別のメンバーの親にすることはできません)
OTLAPI_ERR_SHAREUDA	共有メンバーに対してユーザー属性は設定できません
OTLAPI_ERR_SMARTLISTNAMEUSED	テキスト・リストを追加できません。テキスト・リスト名はすでに使用されています

値	説明
OTLAPI_ERR_SMARTLIST_MAPMAXREACHED	n を超えるテキスト・リストのテキストを ID マッピングに追加できません
OTLAPI_ERR_SMARTLISTMAXREACHED	テキスト・リストを追加できません。サポートされるテキスト・リストは最大 n です
OTLAPI_ERR_SMARTLIST_MISSING	テキスト・タイプ・メンバーのテキスト・リストの関連付けが欠落しています
OTLAPI_ERR_TBTAGS_WITH_DYN_HIERARCHY	このメンバーには TB タグがあります。時間次元が保管階層のみ持つことが必要です
OTLAPI_ERR_TIMESPARSE	会計次元が密で、時間次元が疎、つまり未使用の状態になっています
OTLAPI_ERR_TYPED_ATTR_LEVEL0	属性メンバーおよび非レベル 0 の集約ストレージメンバーは日付またはテキスト・タイプに設定できません
OTLAPI_ERR_TYPED_DIMS	テキスト・タイプ・メンバー、日付タイプ・メンバー、およびフォーマット文字列がある保管済メンバーは、同一次元と一緒に指定する必要があります
OTLAPI_ERR_UNKNOWNDTSMBR	不明な DTS メンバーです
OTLAPI_ERR_VALIDITYSET_MATCH	妥当性セットはアウトラインにある既存のセットと一致している必要があります
OTLAPI_ERR_VIRTLEVONOFORMULA	動的計算メンバーには式または子が必要です。そうでない場合にはメンバーの計算ができません
OTLAPI_ERR_VIRTBADPARENT	1 つの子メンバーが動的計算、または動的計算および保管であるときは、親も動的計算、または動的計算および保管である必要があります
OTLAPI_ERR_VIRTTOOMANYCHILDREN	動的計算メンバーに 100 を超える子があります
OTLAPI_FAILED_ASSIGN_DEFAULTGENNAMES	作成した日時次元に時間に関連する世代名を設定できませんでした
OTLAPI_ILLEGAL_SCA_TYPE_2	可変属性のアウトラインは重複する名前を許可しません。また、通貨アウトラインにすることはできません
OTLAPI_INVALID_ARG	無効な引数が ESSOTL 関数に渡されました
OTLAPI_INVALID_QUERYID	無効なクエリー ID 引数が渡されました
OTLAPI_INVALID_QUERY_OPTIONS	無効なクエリー・オプションが渡されました。無視されます
OTLAPI_NO_GENLEVELNAME	世代名またはレベル名が見つかりません
OTLAPI_NO_USERATTR	ユーザー属性が見つかりません
OTLAPI_NULL_ARG	NULL 引数が EssOtl...関数に渡されました
OTLAPI_OUTLINE_TOO_NEW	アウトラインが、このプログラムで認識できるものよりも新しいバージョンです
OTLAPI_SORT_TOOMANY	ソートの対象となるメンバーの数が多すぎます(最大のソート容量は、64K/4 メンバーです)

値	説明
OTLAPI_SMARTLIST_ASSOC_EXISTS	既存の関連付けを持つテキスト・リストは削除できません
OTLAPI_SMARTLIST_DUP_IDORNAME	テキスト・リストの要素 ID または名前が重複しています
OTLAPI_SMARTLIST_INVALID_TEXT	無効なテキスト・リストのテキストです
OTLAPI_WRONG_INDEPDIM_NM	パースペクティブで指定された独立次元の番号がアウトラインと一致していません

C のアウトライン API DTS メンバー構造体

これらの構造体は、動的時系列(DTS)メンバーに関する情報を含んでいます。

```
   /*
 ESS_DTSMBRNAME_T, ESS_PDTSMBRNAME_T
 DTS member name structure
*/
ESS_TSA_ARRAY_API_typedef(char,         ESS_DTSMBRNAME_T, ESS_MBRNAMELEN);
ESS_TSA_API_typedef    (ESS_DTSMBRNAME_T *,  ESS_PDTSMBRNAME_T);
ESS_TSA_API_typedef    (ESS_PDTSMBRNAME_T *, ESS_PPDTSMBRNAME_T);
```

データ型	フィールド	説明
ESS_DTSMBRNAME_T	szDTSMember	DTS メンバーの名前。
ESS_MBRNAMELEN	szName	DTS メンバー名の長さ。

```
   /*
 ESS_DTSMBRINFO_T, ESS_PDTSMBRINFO_T
 DTS member info structure
*/
ESS_TSA_API_typedef_struct(ess_dtsmbrinfo_t)
{
  ESS_TSA_ELEMENT(ESS_DTSMBRNAME_T, szDTSMember);
  ESS_TSA_ELEMENT(ESS_USHORT_T,   usGen);
} ESS_TSA_END(ESS_DTSMBRINFO_T);

ESS_TSA_API_typedef(ESS_DTSMBRINFO_T *,  ESS_PDTSMBRINFO_T);
ESS_TSA_API_typedef(ESS_DTSMBRINFO_T **, ESS_PPDTSMBRINFO_T);
```

データ型	フィールド	説明
ESS_DTSMBRNAME_T	szDTSMember	DTS メンバーの名前。
ESS_USHORT_T	usGen	DTS メンバーの世代番号。

C のアウトライン API シンボル定数定義

このセクションでは、アウトライン API で使用されるシンボル定数について説明します。これらの定数は Essbase アウトライン API C 言語ヘッダー・ファイル `essotl.h` で定義されています:

会計メンバーの通貨換算カテゴリ値

値	説明
ESS_CONV_NONE	デフォルトの変換カテゴリ。メンバーは、親からカテゴリを継承します。
ESS_CONV_CATEGORY	このメンバーに対する通貨換算カテゴリを定義します
ESS_CONV_NOCONV	このメンバーには換算がありません

会計メンバーのタイム・バランス・スキップ値

タイム・バランスが ESS_TIMEBAL_NONE でない場合にのみ有効

値	説明
ESS_SKIP_NONE	何もスキップしません
ESS_SKIP_MISSING	データが#missing の場合、値をスキップします
ESS_SKIP_ZEROS	データが 0 の場合、値をスキップします
ESS_SKIP_BOTH	データが#missing または 0 の場合、値をスキップします

会計メンバーのタイム・バランス値

値	説明
ESS_TIMEBAL_NONE	タイム・バランスはありません
ESS_TIMEBAL_FIRST	最初のタイム・バランス・メンバー
ESS_TIMEBAL_LAST	最後のタイム・バランス・メンバー
ESS_TIMEBAL_AVG	平均のタイム・バランス・メンバー

次元カテゴリ

ストレージ自動構成の使用時に、ストレージの最適化のために使用されます

値	説明
ESS_STORECAT_ACCOUNTS	会計ストレージ・カテゴリ
ESS_STORECAT_ATTRCALC	属性の計算(集約)ストレージ・カテゴリ
ESS_STORECAT_ATTRIBUTE	属性ストレージ・カテゴリ
ESS_STORECAT_BUSUNIT	ビジネス・ユニット・ストレージ・カテゴリ
ESS_STORECAT_CUSTOMER	顧客ストレージ・カテゴリ
ESS_STORECAT_DIST	流通チャネル・ストレージ・カテゴリ
ESS_STORECAT_GEOG	地域ストレージ・カテゴリ
ESS_STORECAT_MARKET	マーケット・ストレージ・カテゴリ
ESS_STORECAT_ORGAN	組織ストレージ・カテゴリ
ESS_STORECAT_OTHER	ストレージ・カテゴリがないか、または不明です
ESS_STORECAT_PRODUCT	製品ストレージ・カテゴリ
ESS_STORECAT_SCENARIO	シナリオ・ストレージ・カテゴリ
ESS_STORECAT_TIME	時間ストレージ・カテゴリ
ESS_STORECAT_UNITS	単位ストレージ・カテゴリ

次元カテゴリ(タグ)

値	説明
ESS_CAT_ACCOUNTS	会計次元
ESS_CAT_ATTRCALC	属性計算次元またはメンバー。集約のために内部的に使用されます。

値	説明
ESS_CAT_ATTRIBUTE	属性次元またはメンバー
ESS_CAT_COUNTRY	国次元
ESS_CAT_CURPARTITION	通貨パーティション次元。非通貨データベースでのみ有効です。
ESS_CAT_NONE	カテゴリはありません
ESS_CAT_TIME	時間次元
ESS_CAT_TYPE	タイプ次元。通貨データベースでのみ有効です。

メンバー・タイプ

値	説明
ESS_MEMBERTYPE_NONE	タイプがありません
ESS_MEMBERTYPE_NUMERIC	数値型
ESS_MEMBERTYPE_SMARTLIST	テキスト・リスト(スマートリスト)タイプ
ESS_MEMBERTYPE_DATE	日付型

世代/レベル・オプション

EssOtlGetGenNames()および **EssOtlGetLevelNames()**で使用できます

値	説明
ESS_GENLEV_ALL	デフォルト名およびユーザー定義名を戻します
ESS_GENLEV_ACTUAL	ユーザー定義名のみを戻します
ESS_GENLEV_DEFAULT	ユーザー定義名も持つ世代およびレベルのデフォルト名を含むすべてのデフォルト名を戻します
ESS_GENLEV_NOACTUAL	ユーザー定義名も持つ世代およびレベルのデフォルト名を除くすべてのデフォルト名を戻します

クエリー・オプション

778 ページの「ESS_PREDICATE_T」に特定のクエリーのタイプを指定できます

値	説明
ESS_MEMBERSONLY	ESS_SEARCH、ESS_WILDSEARCH に対して有効です
ESS_ALIASESONLY	ESS_SEARCH、ESS_WILDSEARCH に対して有効です

値	説明
ESS_MEMBERSANDALIASES	ESS_SEARCH、ESS_WILDSEARCH に対して有効です
ESS_COUNTONLY	任意のクエリー・タイプに対して有効です。データを戻さずにアウトラインにクエリーを行います。768 ページの「ESS_MBRCOUNTS_T」の ulTotalCount フィールドに値を入力して、クエリー・タイプに一致したメンバー数を戻します。
ESS_INCLUDEHYBRIDANALYSIS	存在する場合はリレーショナル・ソースを含みます。
ESS_EXCLUDEHYBRIDANALYSIS	存在する場合はリレーショナル・ソースを除外します。

クエリー・タイプ

778 ページの「ESS_PREDICATE_T」において実行する操作を定義するために使用します:

- ESS_CHILDREN
- ESS_DESCENDANTS
- ESS_BOTTOMLEVEL
- ESS_SIBLINGS
- ESS_SAMELEVEL
- ESS_SAMEGENERATION
- ESS_PARENT
- ESS_DIMENSION
- ESS_NAMEDGENERATION
- ESS_NAMEDLEVEL
- ESS_SEARCH
- ESS_WILDSEARCH
- ESS_USERATTRIBUTE
- ESS_ANCESTORS
- ESS_DTSMEMBERS
- ESS_DIMUSERATTRIBUTES
- ESS_INDEPDIMS
- ESS_SIBLINGS65
- ESS_INDEPDIMS_DISCRETE
- ESS_INDEPDIMS_CONTINUOUS

再構築値

値	説明
ESS_DOR_ALLDATA	すべてのデータを保持します
ESS_DOR_NODATA	すべてのデータを破棄します
ESS_DOR_LOWDATA	レベル 0 のデータのみを保持します
ESS_DOR_INDATA	入力データのみを保持します
ESS_DOR_FORCE_ALLDATA	すべてのデータをリロードします

共有定数

値	説明
ESS_SHARE_DYNCALCNOSTORE	共有メンバー。非動的計算および保管としてタグ付けされるメンバー。
ESS_SHARE_DYNCALCSTORE	共有メンバー。動的計算および保管としてタグ付けされるメンバー。
ESS_SHARE_DATA	通常メンバー(デフォルト値)
ESS_SHARE_LABEL	ラベル・メンバー。このメンバーに対してはデータを保管しません。
ESS_SHARE_NEVER	たとえ暗黙的に共有になるような場合でも、このメンバーを共有しません。
ESS_SHARE_SHARE	共有メンバー。このメンバーは子を持つことができず、同じ次元に同じ名前の実メンバーが存在している必要があります。

ソート・オプション

値	説明
ESS_SORT_ASCENDING	昇順でのソート
ESS_SORT_DESCENDING	降順でのソート
ESS_SORT_USERDEFINED	ユーザー指定のカスタム・ソート・ルーチンが設定されます

ESS_ATTRIBUTEQUERY_T

EssOtlQueryAttributes によって、属性に関する複雑なクエリーに使用されます。

```
   typedef struct ESS_ATTRIBUTEQUERY_T
{
 ESS_BOOL_T          bInputMemberIsHandle;
 union
```

```
 {
  ESS_HMEMBER_T  hMember;
  ESS_STR_T    szMember;
 }             uInputMember;
 ESS_USHORT_T        usInputMemberType ;
 ESS_USHORT_T        usOutputMemberType;
 ESS_ATTRIBUTEVALUE_T    Attribute;
 ESS_USHORT_T        usOperation;
} ESS_ATTRIBUTEQUERY_T, *ESS_PATTRIBUTEQUERY_T, **ESS_PPATTRIBUTEQUERY_T;
```

データ型	フィールド	説明
ESS_BOOL_T	bInputMemberIsHandle	ブール値: ● TRUE: メンバーのハンドルによる属性クエリー ● FALSE: メンバー名文字列による属性クエリー
ESS_HMEMBER_T ESS_STR_T	uInputMember uInputMember.hMember uInputMember.szMember	次のメンバー参照値に対するユニオン変数: ● メンバーのハンドル ● メンバー名の文字列
ESS_USHORT_T	usInputMemberType	クエリーの対象であるメンバーのデータ型を示す定数識別子: ● ESS_ATTRIBUTE_DIMENSION ● ESS_ATTRIBUTE_MEMBER ● ESS_STANDARD_DIMENSION ● ESS_STANDARD_MEMBER ● ESS_BASE_DIMENSION ● ESS_BASE_MEMBER ● ESS_ATTRIBUTED_MEMBER 表 6 を参照してください。
ESS_USHORT_T	usOutputMemberType	戻されたメンバーのデータ型を示す定数識別子: ● ESS_ATTRIBUTE_DIMENSION ● ESS_ATTRIBUTE_MEMBER ● ESS_STANDARD_DIMENSION ● ESS_STANDARD_MEMBER ● ESS_BASE_DIMENSION ● ESS_BASE_MEMBER ● ESS_ATTRIBUTED_MEMBER ● ESS_INVALID_MEMBER
124 ページの「ESS_ATTRIBUTEVALUE_T」	Attribute	クエリー入力の属性値を定義する構造体

データ型	フィールド	説明
ESS_USHORT_T	usOperation	クエリー操作のタイプを示す定数識別子: ● ESS_EQ: 次と等しい ● ESS_NEQ: 次と等しくない ● ESS_GT: 次より大きい ● ESS_LT: 次より小さい ● ESS_GTE: 次と等しいか大きい ● ESS_LTE: 次と等しいか小さい ● ESS_TYPEOF ● ESS_ALL

ESS_GENLEVELNAME_T

世代名およびレベル名に関する情報を含んでいます。

```
   typedef struct ESS_GENLEVELNAME_T
{
 ESS_USHORT_T  usNumber;
 ESS_MBRNAME_T  szName;
} ESS_GENLEVELNAME_T, *ESS_PGENLEVELNAME_T, **ESS_PPGENLEVELNAME_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usNumber	世代番号またはレベル番号。
ESS_MBRNAME_T	szName	世代名またはレベル名。

ESS_GENLEVELNAMEEX_T

世代名およびレベル名に関する情報を含んでいます。

```
   typedef struct ESS_GENLEVELNAMEEX_T
{
 ESS_USHORT_T  usNumber;
 ESS_BOOL_T,   bNameUnique
 ESS_MBRNAME_T  szName;
} ESS_GENLEVELNAMEEX_T, *ESS_PGENLEVELNAMEEX_T, **ESS_PPGENLEVELNAMEEX_T;
```

データ型	フィールド	説明
ESS_USHORT_T	usNumber	世代番号またはレベル番号。
ESS_BOOL_T	bNameUnique	世代またはレベルのメンバー名の一意性。
ESS_MBRNAME_T	szName	世代名またはレベル名。

ESS_MBRCOUNTS_T

クエリーのメンバー数に関する情報を含んでいます。

```
   typedef struct ESS_MBRCOUNTS_T
{
 ESS_ULONG_T   ulStart;
 ESS_ULONG_T   ulMaxCount;
 ESS_ULONG_T   ulTotalCount;
 ESS_ULONG_T   ulReturnCount;
} ESS_MBRCOUNTS_T, *ESS_PMBRCOUNTS_T, **ESS_PPMBRCOUNTS_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulStart	情報の取得のための開始メンバー。
ESS_ULONG_T	ulMaxCount	取得するメンバーの最大数。
ESS_ULONG_T	ulTotalCount	クエリーの結果存在するメンバーの総数を戻します。これは ulMaxCount を超えることがあります。
ESS_ULONG_T	ulReturnCount	戻されたメンバーのハンドルの数を戻します。これは ulMaxCount を超えることはありません。

ESS_MBRINFO_T

アウトライン・メンバーに関する情報を含みます。

```
   typedef struct ESS_MBRINFO_T
{
 ESS_MBRNAME_T     szMember;
 ESS_USHORT_T      usLevel;
 ESS_USHORT_T      usGen;
 ESS_USHORT_T      usConsolidation;
 ESS_BOOL_T       fTwoPass;
 ESS_BOOL_T       fExpense;
 ESS_USHORT_T      usConversion;
 ESS_MBRNAME_T     szCurMember;
 ESS_USHORT_T      usTimeBalance;
 ESS_USHORT_T      usSkip;
 ESS_USHORT_T      usShare;
 ESS_USHORT_T      usStorage;
 ESS_USHORT_T      usCategory;
 ESS_USHORT_T      usStorageCategory;
 ESS_MBRCOMMENT_T    szComment;
 ESS_ULONG_T      ulChildCount;
 ESS_MBRNAME_T     szDimName;
 ESS_BOOL_T       fAttributed;
 ESS_ATTRIBUTEVALUE_T  Attribute;
 ESS_BOOL_T       fHasRelDesc;
 ESS_BOOL_T       fHasHAEnabled;
 ESS_PVOID_T,      pLastSibling;
 ESS_ULONG_T,      ulSiblingCount;
```

```
 ESS_BOOL_T,      fFormula;
 ESS_BOOL_T,      fUda;
 ESS_BOOL_T,      fAlias;
 ESS_BOOL_T,      fIndependentDim;
 ESS_UCHAR_T,     ucHierarchyType;
 ESS_UCHAR_T,     ucDimSolveOrder;
 ESS_UCHAR_T,     ucSolveOrder;
 ESS_BOOL_T,      fNonUniqueName;
 ESS_BOOL_T,      fFlow;

} ESS_MBRINFO_T, *ESS_PMBRINFO_T, **ESS_PPMBRINFO_T;
```

データ型	フィールド	説明
ESS_MBRNAME_T	szMember	メンバー名。このフィールドは、メンバー作成時に呼出し元でのみ設定できます。
ESS_USHORT_T	usLevel	アウトラインのメンバーのレベル。このフィールドは変更できません。
ESS_USHORT_T	usGen	アウトラインのメンバーの世代。このフィールドは変更できません。
ESS_USHORT_T	usConsolidation	単項集計タイプ。次のいずれかになります: ● ESS_UCALC_ADD ● ESS_UCALC_SUB ● ESS_UCALC_MULT ● ESS_UCALC_DIV ● ESS_UCALC_PERCENT ● ESS_UCALC_NOOP
ESS_BOOL_T	fTwoPass	2 パス計算メンバーの場合は ESS_TRUE。
ESS_BOOL_T	fExpense	支出メンバーの場合、ESS_TRUE。
ESS_USHORT_T	usConversion	通貨換算タイプ。このフィールドは会計次元のメンバーに対してのみ有効です。次のいずれかになります: ● ESS_CONV_NONE ● ESS_CONV_CATEGORY ● ESS_CONV_NOCONV
ESS_MBRNAME_T	szCurMember	メンバーが会計次元に属し、usConversion が ESS_CONV_CATEGORY である場合。このフィールドは通貨カテゴリを定義します。メンバーが国次元に属する場合。このフィールドは通貨名を定義します。このフィールドは、他のすべての状況においては定義されません。
ESS_USHORT_T	usTimeBalance	タイム・バランス・オプション。会計次元のメンバーに対してのみ有効なフィールド。次のいずれかになります: ● ESS_TIMEBAL_NONE ● ESS_TIMEBAL_FIRST ● ESS_TIMEBAL_LAST ● ESS_TIMEBAL_AVG

データ型	フィールド	説明
ESS_USHORT_T	usSkip	タイム・バランス・スキップ・オプション。usTimeBalance が ESS_TIMEBAL_NONE と等しくない場合に会計次元のメンバーにのみ有効なフィールド。次のいずれかになります: ● ESS_SKIP_NONE ● ESS_SKIP_MISSING ● ESS_SKIP_ZEROS ● ESS_SKIP_BOTH
ESS_USHORT_T	usShare	共有オプション。次のいずれかになります: ● ESS_SHARE_DATA (デフォルト値) ● ESS_SHARE_DYNCALCSTORE ● ESS_SHARE_DYNCALCNOSTORE ● ESS_SHARE_LABEL ● ESS_SHARE_NEVER ● ESS_SHARE_SHARE (レベル 0 メンバーにのみ有効)
ESS_USHORT_T	usStorage	次元ストレージ・タイプ。このフィールドは、次元メンバーに対してのみ有効です。次のいずれかの値になります: ● ESS_DIMTYPE_DENSE ● ESS_DIMTYPE_SPARSE
ESS_USHORT_T	usCategory	次元カテゴリ。このフィールドは、次元メンバーおよび属性メンバーに対してのみ有効です。次のいずれかになります: ● ESS_CAT_ACCOUNTS ● ESS_CAT_ATTRCALC (システム内部でのみ使用) ● ESS_CAT_ATTRIBUTE ● ESS_CAT_COUNTRY ● ESS_CAT_CURPARTITION (非通貨データベースのみ) ● ESS_CAT_NONE ● ESS_CAT_TIME ● ESS_CAT_TYPE (通貨データベースのみ)

データ型	フィールド	説明
ESS_USHORT_T	usStorageCategory	次元ストレージ・カテゴリ。このフィールドは、次元メンバーおよび属性メンバーに対してのみ有効です。アウトラインが自動最適化用に構成されているとき、次元のストレージ・タイプを最適化します。次のいずれかになります: ● ESS_STORECAT_ACCOUNTS ● ESS_STORECAT_ATTRCALC (システム内部でのみ使用) ● ESS_STORECAT_ATTRIBUTE ● ESS_STORECAT_BUSUNIT ● ESS_STORECAT_CUSTOMER ● ESS_STORECAT_DIST ● ESS_STORECAT_GEOG ● ESS_STORECAT_MARKET ● ESS_STORECAT_ORGAN ● ESS_STORECAT_OTHER ● ESS_STORECAT_PRODUCT ● ESS_STORECAT_SCENARIO ● ESS_STORECAT_TIME ● ESS_STORECAT_UNITS
ESS_MBRCOMMENT_T	szComment	メンバー・コメント配列
ESS_ULONG_T	ulChildCount	このフィールドには、ESS_MBRNAME_T で指定されたメンバーの子の合計数が含まれています。
ESS_MBRNAME_T	szDimName	属性次元名
ESS_BOOL_T	fAttributed	メンバーに属性が関連付けられているかどうかを示します。値: ESS_TRUE および ESS_FALSE。
124 ページの「ESS_ATTRIBUTEVALUE_T」	Attribute	属性値
ESS_BOOL_T	fHasRelDesc	メンバーは、リレーショナル上の子孫を持ちます。
ESS_BOOL_T	fHasHAEnabled	次元では、リレーショナル・パーティションが使用可能になっています。 次元メンバーにのみ有効です。
RSS_PVOID_T	pLastSibling	最後の兄弟ポインタ
ESS_ULONG_T	uSiblingCount	兄弟カウント
ESS_BOOL_T	fFormula	式を持つかどうかを示します
ESS_BOOL_T	fUda	UDA を持つかどうかを示します
ESS_BOOL_T	fAlias	別名を持つかどうかを示します
ESS_BOOL_T	fIndependentDim	可変属性アウトラインの次元用。独立次元かどうかを示します

データ型	フィールド	説明
ESS_UCHAR_T	ucHierarchyType	世代に基づいて階層タイプを定義します。 メンバーが世代 1 である場合: ● ESS_STORED_HIERARCHY は単一の保管階層を示します ● ESS_DYNAMIC_HIERARCHY は単一の動的階層を示します ● ESS_MULTIPLE_HIERARCHY_IS_ENABLED は複数の階層を示します メンバーが世代 2 である場合: ● ESS_STORED_HIERARCHY はサブ階層を示します ● ESS_DYNAMIC_HIERARCHY は動的サブ階層を示します
ESS_UCHAR_T	udDimSolveOrder	次元についての解決順を定義します。
ESS_UCHAR_T	udSolveOrder	解決順の値を示します。解決順は 0-127 になります。
ESS_BOOL_T	fNonUniqueName	メンバー名が一意かどうかを示します
ESS_BOOL_T	fFlow	メンバーのタイプがフローであることを示します

ESS_OTLQUERYERRORLIST_T

拡張メンバーのクエリー中、つまり、EssOtlQueryMembersEx の呼出し中に検出されたエラーの一覧を保管します。

```
   typedef struct ESS_OTLQUERYERRORLIST_T
{
 ESS_ULONG_T     ulCount;
 ESS_OTLQUERYERROR_T* ErrorArray;

} ESS_OTLQUERYERRORLIST_T, *ESS_POTLQUERYERRORLIST_T, **ESS_PPOTLQUERYERRORLIST_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulCount	クエリー中に、エラーの数が戻されます
ESS_OTLQUERYERROR_T*	ErrorArray	クエリー中に、エラーの配列に対するポインタが戻されます

ESS_OUTERROR_T

アウトラインを確認するときに、各メンバーのエラーを戻します。エラーは、32 ビットのステータス・ワードで戻される、ビット・フィールド値です。各エラー値は、表 7 に示した関数呼出しエラー戻り値に対応します。

```
   typedef struct ESS_OUTERROR_T
{
 ESS_HMEMBER_T  hMember;
```

```
 ESS_ULONG_T  ulErrors;
} ESS_OUTERROR_T, *ESS_POUTERROR_T, **ESS_PPOUTERROR_T;
```

データ型	フィールド	説明
ESS_HMEMBER_T	hMember	エラーのあるメンバーへのハンドル。
ESS_ULONG_T	ulErrors	メンバーに対するエラーのビットマスク。ulErrors の値を参照してください。

ulErrors の値

ulErrors で使用できる値を次に示します:

- ESS_OUTERROR_ALIASSHARED
- ESS_OUTERROR3_ASO_BAD_AGGREGATION_OPERATOR
- ESS_OUTERROR3_ASO_BAD_NONLEAFMBR
- ESS_OUTERROR3_ASO_DYNASSOCD
- ESS_OUTERROR3_ASO_EITHERLABELORFORMULA
- ESS_OUTERROR3_ASO_INVALID_AGGLEVELUSAGE
- ESS_OUTERROR3_ASO_INVALIDATTRCALC
- ESS_OUTERROR3_ASO_ISDUPLICATESHAREINHIERARCHY
- ESS_OUTERROR3_ASO_LABEL_SPAN
- ESS_OUTERROR3_ASO_LEVELPRODUCT_TOO_LARGE
- ESS_OUTERROR3_ASO_NOATTRIBUTE_ON_ACCOUNTS
- ESS_OUTERROR3_ASO_NOFORMULA
- ESS_OUTERROR4_ASO_PROTOLEVELZERO
- ESS_OUTERROR3_ASO_SHAREDMBR
- ESS_OUTERROR3_ASO_TWOCHILDRENFORTHISOPER
- ESS_OUTERROR3_ASO_WHOLEACCOUNTSDIMVIRTUAL
- ESS_OUTERROR2_ATTRCALCABSENT
- ESS_OUTERROR2_ATTRDIMNOTASSOCIATED
- ESS_OUTERROR_BADATTRIBUTECODE
- ESS_OUTERROR_BADCATEGORY
- ESS_OUTERROR_BADSHARE
- ESS_OUTERROR_BADSKIP
- ESS_OUTERROR_BADSTORAGE
- ESS_OUTERROR_BADSTORAGECATEGORY
- ESS_OUTERROR_BADTIMEBAL
- ESS_OUTERROR2_BOOLEANNAMESETTING
- ESS_OUTERROR2_CHILDCOUNT

- ESS_OUTERROR_CURTOOMANYDIMS
- ESS_OUTERROR2_DATATYPEMISMATCH
- ESS_OUTERROR_DUPGENLEVNAME
- ESS_OUTERROR_DUPLICATEALIAS
- ESS_OUTERROR2_DUPLICATEATTRCALC
- ESS_OUTERROR_DUPLICATENAME
- ESS_OUTERROR4_DUPNAME_INDIMENSION
- ESS_OUTERROR4_DUPNAME_INGENERATION
- ESS_OUTERROR4_DUPNAME_INLEVEL
- ESS_OUTERROR4_FLOWTAGINCOMPLETE
- ESS_OUTERROR_ILLEGALALIASSTRING
- ESS_OUTERROR2_ILLEGALATTRCALC
- ESS_OUTERROR2_ILLEGALATTRCALCSET
- ESS_OUTERROR2_ILLEGALATTRVALUE
- ESS_OUTERROR2_ILLEGALATTRIBUTEPARENT
- ESS_OUTERROR2_ILLEGALATTRSET
- ESS_OUTERROR_ILLEGALCOMBOALIAS
- ESS_OUTERROR_ILLEGALCURRENCY
- ESS_OUTERROR2_ILLEGALDATATYPE
- ESS_OUTERROR_ILLEGALDEFALIAS
- ESS_OUTERROR_ILLEGALNAME
- ESS_OUTERROR2_ILLEGALORDER
- ESS_OUTERROR2_ILLEGALSCAASSOCS
- ESS_OUTERROR_ILLEGALTAG
- ESS_OUTERROR2_ILLEGALUDA
- ESS_OUTERROR2_INDEPMBR_BADORDER
- ESS_OUTERROR2_INDEPMBR_NOTLEVEL0
- ESS_OUTERROR2_INDEPMBR_SHAREORLABEL
- ESS_OUTERROR_LEAFLABEL
- ESS_OUTERROR2_LEVELMISMATCH
- ESS_OUTERROR_MEMBERCALC
- ESS_OUTERROR_NOSHAREPROTO
- ESS_OUTERROR2_NOTATTRIBUTE
- ESS_OUTERROR_NOTIMEDIM
- ESS_OUTERROR2_NOTLEVEL0

- ESS_OUTERROR4_PROTO_NONUNIQUE
- ESS_OUTERROR_SHAREDMEMBERFORMULA
- ESS_OUTERROR_SHARENOTLEVEL0
- ESS_OUTERROR_SHAREUDA
- ESS_OUTERROR4_TI_INCORRECT_MBRTIMESPANS
- ESS_OUTERROR4_TI_INVALIDCONSOLIDATION
- ESS_OUTERROR4_TI_LINKATTR_INVALID
- ESS_OUTERROR4_TI_LINKATTR_INVALIDASSOC
- ESS_OUTERROR4_TI_LINKATTR_UNBALANCEDHIER
- ESS_OUTERROR4_TI_ONLYONE_SINGLEHIER
- ESS_OUTERROR_TIMESPARSE
- ESS_OUTERROR2_TWOPASSPARENTNONTWOPASS
- ESS_OUTERROR_VIRTLEV0NOFORMULA
- ESS_OUTERROR_VIRTBADCHILD
- ESS_OUTERROR_VIRTBADPARENT
- ESS_OUTERROR_VIRTWHOLEDIMVIRTUAL
- ESS_OUTERROR4_20DUPNAME_INPATH

ESS_OUTLINEINFO_T

アウトラインに関する情報が含まれます。

```
   typedef struct ESS_OUTLINEINFO_T
{
 ESS_BOOL_T      fCaseSensitive;
 ESS_USHORT_T     usOutlineType;
 ESS_BOOL_T      fAutoConfigure;
 ESS_USHORT_T     usNumAliasTables;
 ESS_ALIASNAME_T   pAliasTables[1];
 ESS_BOOL_T      fEnableVaryingAttrs;
 ESS_BOOL_T      fNonUniqueName;
 ESS_UCHAR_T,     ucImpliedShareSetting;
 ESS_BOOL_T,     fEnableMemberType;
} ESS_OUTLINEINFO_T, *ESS_POUTLINEINFO_T, **ESS_PPOUTLINEINFO_T;
```

データ型	フィールド	説明
ESS_BOOL_T	fCaseSensitive	大文字と小文字を区別するメンバー名フラグ。

データ型	フィールド	説明
ESS_USHORT_T	usOutlineType	アウトラインのタイプ。次のいずれかになります: ● ESS_DBTYPE_NORMAL 通常のデータベース ● ESS_DBTYPE_CURRENCY 通貨データベース ● ESS_DBTYPE_NORMALMDX MDX タイプの式を含むデータベース ● ESS_DBTYPE_ASO 集約ストレージ・データベース ● ESS_DBTYPE_ROLAP ROLAP データベース ● ESS_DBTYPE_ASO71 version 7.1 otl ファイルを含む集約ストレージ・データベース
ESS_BOOL_T	fAutoConfigure	ESS_TRUE を指定すると、ブロックストレージ・アウトラインが保存されるときに自動的に次元ストレージ(密/疎)が構成されます。
ESS_USHORT_T	usNumAliasTables	別名テーブルの数。このフィールドは読取り専用で、**EssOtlSetOutlineInfo()**の呼出しでは無視されます。
ESS_ALIASNAME_T	pAliasTables	アウトライン内の別名テーブル名の配列。「usNumAliasTables」フィールドでは、この配列内のエントリ数を定義します。これは読取り専用フィールドで、**EssOtlSetOutlineInfo()**の呼出しでは無視されます。
ESS_BOOL_T	fEnableVaryingAttrs	ESS_TRUE は可変属性をサポートするアウトラインを示します。
ESS_BOOL_T	fNonUniqueName	重複するメンバー名をアウトラインでサポートするかどうかを指定します。
ESS_UCHAR_T	ucImpliedShareSetting	暗黙の共有設定: ● TRUE (デフォルト)の場合、暗黙の共有は ON です ● FALSE の場合、暗黙の共有は OFF です
ESS_BOOL_T	fEnableMemberType	ESS_TRUE はメンバー・タイプが使用可能なことを示します。

ESS_OUTLINEINFOEX_T

アウトラインに関する情報が含まれます。

```
   typedef struct ESS_OUTLINEINFOEX_T
{
 ESS_BOOL_T      fCaseSensitive;
 ESS_USHORT_T     usOutlineType;
 ESS_BOOL_T      fAutoConfigure;
 ESS_BOOL_T,      fNonUniqueName;
 ESS_USHORT_T     usNumAliasTables;
 ESS_ALIASNAME_T   pAliasTables[1];
 ESS_BOOL_T      fEnableVaryingAttrs;
```

```
 ESS_UCHAR_T,     ucImpliedShareSetting
 ESS_BOOL_T,      fEnableMemberType;
 ESS_CHAR_T,      cSMDateFormatValue;
} ESS_OUTLINEINFOEX_T, *ESS_POUTLINEINFOEX_T, **ESS_PPOUTLINEINFOEX_T;
```

データ型	フィールド	説明
ESS_BOOL_T	fCaseSensitive	大文字と小文字を区別するメンバー名フラグ。
ESS_USHORT_T	usOutlineType	アウトラインのタイプ。次のいずれかにできます: ● ESS_DBTYPE_NORMAL ● ESS_DBTYPE_CURRENCY
ESS_BOOL_T	fAutoConfigure	ESS_TRUE を指定すると、ブロックストレージ・アウトラインが保存されるときに自動的に次元ストレージ(密/疎)が構成されます。
ESS_BOOL_T	fNonUniqueName	重複するメンバー名をアウトラインでサポートするかどうかを指定します。
ESS_USHORT_T	usNumAliasTables	別名テーブルの数。これは読取り専用フィールドで、**EssOtlSetOutlineInfo()**の呼出しでは無視されます。
ESS_ALIASNAME_T	pAliasTables	アウトライン内の別名テーブル名の配列。「usNumAliasTables」フィールドでは、この配列内のエントリ数を定義します。これは読取り専用フィールドで、**EssOtlSetOutlineInfo** の呼出しでは無視されます。
ESS_BOOL_T	fEnableVaryingAttrs	ESS_TRUE は可変属性をサポートするアウトラインを示します。
ESS_UCHAR_T	ucImpliedShareSetting	アウトラインの暗黙の共有設定。可能な値: ● ESS_IMPLIEDSHARE_DEFAULT ● ESS_IMPLIEDSHARE_DEFAULT_ON ● ESS_IMPLIEDSHARE_DEFAULT_OFF ● ESS_IMPLIEDSHARE_FORCE_ON ● ESS_IMPLIEDSHARE_FORCE_OFF
ESS_BOOL_T	fEnableMemberType	
ESS_UCHAR_T	cSMDateFormatValue	

ESS_PERSPECTIVE_T

パースペクティブと妥当性セットに関する情報が含まれます。

```
   typedef struct ESS_PERSPECTIVE_T
{
 ESS_USHORT_T, usValiditySetType;
 ESS_USHORT_T, usFiller;
 ESS_STR_T, szValiditySetExpr;
 ESS_INT32_T, countOfIndepDims;
 ESS_INT32_T, countOfIndepRanges;
 ESS_PVOID_T*, pIndepMbrs;
} ESS_PERSPECTIVE_T; *ESS_PPERSPECTIVE_T
```

データ型	フィールド	説明
ESS_USHORT_T	usValiditySetType	メンバーの指定方法。可能な値: ● ESS_VALIDITYSET_TYPE_MBRHDLS ● ESS_VALIDITYSET_TYPE_MBRNAMS
ESS_USHORT_T	usFiller	ゼロに設定
ESS_STR_T	szValiditySetExpr	MDX タイプによって指定される MDX 式
ESS_INT32_T	countOfIndepDims	各タプルのサイズ
ESS_INT32_T	countOfIndepRanges	タプル範囲の数
ESS_PVOID_T	pIndepMbrs	usValiditySetType の内容によって、メンバー・ハンドル(ESS_HMEMBER_T)またはメンバー名(ESS_STR_T)の配列のいずれか

説明

パースペクティブおよび妥当性セットはいずれも、独立したメンバーの特定の集合を指定します。

- **パースペクティブ**は、独立したメンバーの任意の組合せを指定し、クライアントまたはサーバーで関連付けをクエリーする際に使用します。
- **妥当性セット**は関連付けが TRUE である独立したメンバーの集合を指定します。この用語は関連付け、または関連付け解除に使用する独立したメンバーのセットにも適用されます。

独立したメンバーの指定内容:

- ESS_VALIDITYSET_TYPE_MBRHDLS: 独立したメンバーは一連のメンバー・ハンドル範囲として指定されます(XRange。すなわち Mar 2003-Feb 2004 は、2003 年 3 月から 2004 年の 1 月/2 月までを示します)。
- ESS_VALIDITYSET_TYPE_MBRNAMS: ESS_VALIDITYSET_TYPE_MBRHDLS と同様ですが、メンバー名で範囲が指定される点が異なります。

ESS_PREDICATE_T

クエリー記述に関する情報を含んでいます。

```
   typedef struct ESS_PREDICATE_T
{
 ESS_ULONG_T   ulQuery;
 ESS_ULONG_T   ulOptions;
 ESS_STR_T    pszDimension;
 ESS_STR_T    pszString1;
 ESS_STR_T    pszString2;
} ESS_PREDICATE_T, *ESS_PPREDICATE_T, **ESS_PPPREDICATE_T;
```

データ型	フィールド	説明
ESS_ULONG_T	ulQuery	クエリーのタイプ。詳細は、EssOtlQueryMembers を参照してください。
ESS_ULONG_T	ulOptions	クエリーのタイプに依存するオプション。詳細は、EssOtlQueryMembers を参照してください。
ESS_STR_T	pszDimension	次元名。詳細は、EssOtlQueryMembers を参照してください。
ESS_STR_T	pszString1	入力文字列の値。詳細は、EssOtlQueryMembers を参照してください。
ESS_STR_T	pszString2	入力文字列の値。詳細は、EssOtlQueryMembers を参照してください。

ESS_SVROTLINFO_T

アウトラインに関する情報が含まれます。この構造体は EssGetSrvOutlineInfo で使用できます。

```
   typedef struct ESS_SVROTLINFO_T
{

    ESS_BOOL_T,    fCaseSensitive;
    ESS_USHORT_T,   usOutlineType;
    ESS_BOOL_T,    fNonUniqueName;
    ESS_USHORT_T,   usNumAliasTables;
    ESS_ALIASNAME_T, pAliasTables, 10;
} ESS_SVROTLINFO_T, *ESS_PSVROTLINFO_T;
```

データ型	フィールド	説明
ESS_BOOL_T	fCaseSensitive	大文字と小文字を区別するメンバー名フラグ。
ESS_USHORT_T	usOutlineType	アウトラインのタイプ。次のいずれかになります: ● ESS_DBTYPE_NORMAL ● ESS_DBTYPE_CURRENCY
ESS_BOOL_T	fNonUniqueName	重複するメンバー名をアウトラインでサポートするかどうかを指定します。
ESS_USHORT_T	usNumAliasTables	別名テーブルの数。このフィールドは読取り専用で、**EssOtlSetOutlineInfo()** の呼出しでは無視されます。
ESS_ALIASNAME_T	pAliasTables	アウトライン内の別名テーブル名の配列。「usNumAliasTables」フィールドでは、この配列内のエントリ数を定義します。これは読取り専用フィールドで、**EssOtlSetOutlineInfo()**の呼出しでは無視されます。

ESS_VALIDITYSET_T

パースペクティブと妥当性セットに関する情報が含まれます。

```
   typedef struct ESS_VALIDITYSET_T
```

```
{
 ESS_USHORT_T, usValiditySetType;
 ESS_USHORT_T, usFiller;
 ESS_STR_T, szValiditySetExpr;
 ESS_INT32_T, countOfIndepDims;
 ESS_INT32_T, countOfIndepRanges;
 ESS_PVOID_T*, pIndepMbrs;
} ESS_VALIDITYSET_T; *ESS_PVALIDITYSET_T
```

データ型	フィールド	説明
ESS_USHORT_T	usValiditySetType	メンバーの指定方法。可能な値: ● ESS_VALIDITYSET_TYPE_MBRHDLS ● ESS_VALIDITYSET_TYPE_MBRNAMS
ESS_USHORT_T	usFiller	ゼロに設定
ESS_STR_T	szValiditySetExpr	MDX タイプによって指定される MDX 式
ESS_INT32_T	countOfIndepDims	各タプルのサイズ
ESS_INT32_T	countOfIndepRanges	タプル範囲の数
ESS_PVOID_T	pIndepMbrs	usValiditySetType の内容によって、メンバー・ハンドル(ESS_HMEMBER_T)またはメンバー名(ESS_STR_T)の配列のいずれか

説明

パースペクティブおよび妥当性セットはいずれも、独立したメンバーの特定の集合を指定します。

- **パースペクティブ**は、独立したメンバーの任意の組合せを指定し、クライアントまたはサーバーで関連付けをクエリーする際に使用します。
- **妥当性セット**は関連付けが TRUE である独立したメンバーの集合を指定します。この用語は関連付け、または関連付け解除に使用する独立したメンバーのセットにも適用されます。

独立したメンバーの指定内容:

- ESS_VALIDITYSET_TYPE_MBRHDLS: 独立したメンバーは一連のメンバー・ハンドル範囲として指定されます(XRange。すなわち Mar 2003-Feb 2004 は、2003 年 3 月から 2004 年の 1 月/2 月までを示します)。
- ESS_VALIDITYSET_TYPE_MBRNAMS: ESS_VALIDITYSET_TYPE_MBRHDLS と同様ですが、メンバー名で範囲が指定される点が異なります。

11 CのアウトラインAPI関数

この章の内容

C のアウトライン API 関数のカテゴリ

カテゴリ別の C のアウトライン API 関数:

- 781 ページの「C のアウトライン API 別名テーブル関数」
- 782 ページの「C のアウトライン API 属性の関数」
- 783 ページの「C のアウトライン API 動的時系列関数」
- 783 ページの「C のアウトライン API 世代名関数 」
- 783 ページの「C のアウトライン API レベル名関数」
- 783 ページの「C のアウトライン API メンバー管理関数」
- 784 ページの「C のアウトライン API メンバー別名関数」
- 784 ページの「C のアウトライン API メンバー式関数」
- 785 ページの「C のアウトライン API メンバー走査関数」
- 785 ページの「C のアウトライン API アウトライン管理関数」
- 786 ページの「C のアウトライン API アウトライン・クエリー関数」
- 786 ページの「C のアウトライン API 設定およびクリーンアップ関数」
- 787 ページの「C のアウトライン API ユーザー定義属性関数」
- 787 ページの「C のアウトライン API ユーザー定義ビュー選択関数」
- 788 ページの「C のアウトライン API の可変属性関数」

C のアウトライン API 別名テーブル関数

次の関数は、別名テーブルに対する操作を実行します。

関数	説明
EssOtlCreateAliasTable()	アウトラインに空の別名テーブルを作成します
EssOtlCopyAliasTable	別名テーブルを他の別名テーブルにコピーします

関数	説明
EssOtlRenameAliasTable	既存の別名テーブル名を変更します
EssOtlClearAliasTable	既存の別名テーブルを削除せずに、そのエントリをすべて消去します
EssOtlDeleteAliasTable	別名テーブルをアウトラインから削除し、そのエントリをすべて消去します
EssOtlSetAliasTableLanguage	別名テーブルの言語コードを設定します。1 つの別名テーブルに複数の言語コードを設定できます。
EssOtlGetAliasTableLanguages	別名テーブルに関連付けられた言語コードのセットを取得します。
EssOtlClearAliasTableLanguages	別名テーブルから言語コードのセットを消去します。

C のアウトライン API 属性の関数

次の C アウトライン関数は、属性に関する関数です。

788 ページの「C のアウトライン API の可変属性関数」についての説明も参照してください。

関数	説明
EssOtlAssociateAttributeDimension	属性次元を基本次元に関連付けます
EssOtlAssociateAttributeMember	属性メンバーを基本次元メンバーに関連付けます
EssOtlDisassociateAttributeDimension	属性次元と基本次元との関連付けを解除します
EssOtlDisassociateAttributeMember	属性メンバーと基本次元メンバーとの関連付けを解除します
EssOtlFindAttributeMembers	属性メンバーに関連付けられているすべての基本次元メンバーを戻します
EssOtlFreeStructure	文字列タイプの属性情報用に動的に割り当てられたメモリーを解放します
EssOtlGetAssociatedAttributes	基本次元メンバーまたは基本次元と関連付けられているすべての属性次元メンバーを戻します
EssOtlGetAttributeInfo	指定した属性メンバーまたは属性次元に関する属性情報を戻します
EssOtlGetAttributeSpecifications	アウトラインの属性指定を取得します
EssOtlQueryAttributes	アウトラインに対してメンバー属性情報に関するクエリーを実行します
EssOtlQueryAttributesEx	
EssOtlSetAttributeSpecifications	アウトラインの属性指定を設定します

213 ページの「C のメイン API 属性関数」に関する説明を参照してください。

C のアウトライン API 動的時系列関数

次の関数は、動的時系列メンバーおよび別名を使用可能にして処理します。

関数	説明
EssOtlDeleteDTSMemberAlias	動的時系列メンバーの別名を削除します。
EssOtlEnableDTSMember	アウトラインの新規動的時系列メンバーを使用可能にします。
EssOtlGetEnabledDTSMembers	アウトラインに対して定義されている新規動的時系列メンバーを取得します。
EssOtlGetDTSMemberAlias	動的時系列メンバーの別名を取得します。
EssOtlSetDTSMemberAlias	動的時系列メンバーの別名を設定します。

C のアウトライン API 世代名関数

次の関数は、世代名に対する操作を実行します。

関数	説明
EssOtlGetGenName	指定された次元および世代番号の世代名を取得します
EssOtlGetGenNames	特定の次元に対して指定されたすべての世代名を取得します
EssOtlSetGenName	指定された次元および世代番号の世代名を設定します
EssOtlDeleteGenName	指定された次元およびレベル番号の世代名を削除します

C のアウトライン API レベル名関数

次の関数は、レベル名に対する操作を実行します。

関数	説明
EssOtlGetLevelName	指定された次元のレベル名を取得します
EssOtlGetLevelNames	特定の次元に対して指定されたすべてのレベル名を取得します
EssOtlSetLevelName	指定された次元のレベル名を設定します
EssOtlDeleteLevelName	指定された次元のレベル名を削除します

C のアウトライン API メンバー管理関数

次の関数は、アウトラインのメンバーの管理を支援します。

関数	説明
EssOtlAddMember	メンバーを追加します
EssOtlDeleteMember	メンバーを削除します
EssOtlAddDimension	次元を追加します
EssOtlDeleteDimension	次元を削除します
EssOtlRenameMember	メンバー名を変更します
EssOtlMoveMember	メンバーを移動します
EssOtlFindMember	メンバーを検索します
EssOtlGetMemberCommentEx	指定されたメンバーに対して、拡張コメントを取得します
EssOtlGetMemberInfo	メンバー情報を取得します
EssOtlSetMemberCommentEx	指定されたメンバーに対して、拡張コメントを設定します
EssOtlSetMemberInfo	メンバー情報を設定します
EssOtlGetMemberSolveOrder	メンバーの解決順を取得します
EssOtlSetMemberSolveOrder	メンバーの解決順を設定します
EssOtlGetDimensionSolveOrder	次元の解決順を取得します
EssOtlSetDimensionSolveOrder	次元の解決順を設定します

C のアウトライン API メンバー別名関数

次の関数は、メンバー別名に対する操作を実行します。

関数	説明
EssOtlFindAlias	指定された別名を持つメンバーを検索します
EssOtlGetMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を取得します
EssOtlSetMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を設定します
EssOtlDeleteMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を削除します

C のアウトライン API メンバー式関数

次の関数は、メンバー式に対する操作を実行します。

関数	説明
EssOtlGetMemberFormula	指定されたメンバーの式を取得します
EssOtlGetMemberLastFormula	メンバーの計算に使用された最後の式を戻します
EssOtlSetMemberFormula	指定されたメンバーに対して式を設定します
EssOtlDeleteMemberFormula	指定されたメンバーの式を削除します

C のアウトライン API メンバー走査関数

次の関数は、アウトライン・ツリーの走査に使用されます。

関数	説明
EssOtlGetFirstMember	アウトラインの最初のメンバー、すなわちアウトラインで最初に定義されている次元へメンバーのハンドルを戻します
EssOtlGetChild	メンバーの子へメンバーのハンドルを戻します
EssOtlGetParent	メンバーの親へメンバーのハンドルを戻します
EssOtlGetNextSibling	メンバーの次の兄弟へメンバーのハンドルを戻します
EssOtlGetPrevSibling	メンバーの前の兄弟へメンバーのハンドルを戻します
EssOtlGetNextSharedMember	実メンバーの次の共有メンバーへメンバーのハンドルを戻します
EssOtlQueryGetFirstDimension	アウトラインの最初の次元の次元ハンドルを戻します
EssOtlQueryGetNextDimension()	クエリー・モードで開かれたアウトラインの次元の次の次元ハンドルを戻します

C のアウトライン API アウトライン管理関数

次の関数は、アウトラインの管理を支援します。

関数	説明
EssOtlGetOutlineInfo	アウトライン・ファイルに関する情報を戻します
EssOtlGetUpdateTime	指定されたアウトラインに対するタイムスタンプを戻します
EssOtlSetOutlineInfo	アウトライン情報を設定します
EssOtlVerifyOutline	アウトラインが正しいことを確認します
EssOtlSortChildren	アウトライン・メンバーの子をソートします
EssOtlGenerateCurrencyOutline	既存のアウトラインを基に通貨アウトラインを生成します
EssOtlGetASOCompressionDimension	集約ストレージ圧縮次元を取得します

関数	説明
EssOtlSetASOCompressionDimension	集約ストレージ圧縮次元を設定します

C のアウトライン API アウトライン・クエリー関数

次の関数は、アウトラインのクエリー実行を支援します。

関数	説明
EssOtlGetMemberField	指定されたアウトライン・メンバーの指定されたフィールドのデータを戻します
EssOtlOpenOutlineQuery	既存のアウトラインを開きます
EssOtlQueryMembers	メンバーのハンドルを使用して、アウトラインにクエリーを行います
EssOtlQueryMembersByName	メンバー名文字列を使用して、アウトラインにクエリーを行います
EssOtlQueryMembersEx	特定のメンバーおよびメンバー・フィールドのクエリーを実行し、メンバーのハンドルの配列を戻します
EssOtlQueryAttributes	アウトラインに対し、属性情報についてのクエリーを実行します。
EssOtlQueryAttributesEx	
EssOtlFreeMembers	**EssOtlQueryMembers()**から戻されたメンバー配列を解放します

C のアウトライン API 設定およびクリーンアップ関数

次の関数は、アウトラインの編集操作を開始および終了します。

関数	説明
EssOtlNewOutline	アウトラインを新規に作成します
EssOtlOpenOutline	既存のアウトラインを開きます
EssOtlOpenOutlineEx	既存のアウトラインを開きます(Unicode モード用)
EssOtlWriteOutline	アウトラインをサーバーに書き込みます
EssOtlWriteOutlineEx	アウトラインをサーバーに書き込みます(Unicode モード用)
EssOtlRestructure	新規に保存されたアウトラインに基づいて、データベースを再構築します
EssOtlCloseOutline	アウトラインに関連付けられているリソースを解放します

C のアウトライン API の Unicode モードの関数

次の関数は、Unicode モードで Essbase サーバーのアウトラインを操作するときに役立ちます。

関数	説明
EssOtlWriteOutlineEx	アウトラインをサーバーに書き込み、UTF-8 エンコード方式と非 Unicode エンコード方式のどちらで保存するかを指定します。
EssOtlOpenOutlineEx	Unicode モードのアプリケーションのアウトラインを開きます。

C のアウトライン API ユーザー定義属性関数

次の関数は、ユーザー定義属性(UDA)に対する操作を実行します。

関数	説明
EssOtlGetDimensionUserAttributes	指定された次元の UDA を取得します
EssOtlGetUserAttributes	指定されたメンバーの UDA を取得します
EssOtlSetUserAttribute	指定されたメンバーの UDA を設定します
EssOtlDeleteUserAttribute	指定されたメンバーの UDA を削除します

C のアウトライン API ユーザー定義ビュー選択関数

これらの関数は、集約ストレージ・データベースの集約のビュー選択条件を定義します。

関数	説明
EssOtlSetAggLevelUsage	保管された階層にビュー選択プロパティを適用します
EssOtlGetAggLevelUsage	保管された階層に適用されたビュー選択プロパティを戻します
EssOtlAddQueryHint	ビュー選択を支援するクエリー・ヒントをアウトラインに追加します
EssOtlGetQueryHint	アウトラインに定義され、指定されたクエリー・ヒントを戻します
EssOtlSetQueryHint	クエリー・ヒントを設定します
EssOtlGetNumQueryHints	クエリー・ヒントの数を戻します
EssOtlGetQueryHintSize	クエリー・ヒントのサイズ(メンバー数)を戻します
EssOtlDeleteQueryHint	指定したクエリー・ヒントを削除し、ヒント番号を 1 つ減少させます

C のアウトライン API の可変属性関数

次の C アウトライン関数は、可変属性に関する関数です。

関数	説明
EssOtlQueryVaryingAttributes	アウトラインに対して、メンバー可変属性情報に関するクエリーを実行します
EssOtlDetailQueryVaryingAttributes	EssOtlQueryVaryingAttributes と似ています。
EssOtlVaryingAssociateAttribute	可変属性メンバーを基本次元メンバーに関連付けます
EssOtlVaryingAssociateAttributeDimension	可変属性次元を基本次元に関連付けます
EssOtlVaryingDisassociateAttribute	可変属性次元の基本次元との関連付けを解除します
EssOtlVaryingGetAssociatedAttributes	基本次元メンバーまたは基本次元に関連付けられているすべての可変属性メンバーを戻します
EssOtlVaryingGetAttributeIndepDims	指定された可変属性メンバーを含んでいる次元に対して、独立次元(ある場合)を戻します

次の API は、今後の実装と完全互換でない可能性があります。

213 ページの「C のメイン API 属性関数」に関する説明を参照してください。

可変属性について

属性の関連付けは、外部の要素に左右される場合があります。たとえば

- 時間が経過すると、クライアントに異なる販売担当者が割り当てられる場合があります。
- 時間が経過すると、または市場の担当地域に基づいて、製品の包装が異なる場合があります。

可変属性機能によって、各要素に対する値を追跡できます。たとえば、顧客 A に対する販売担当者属性の関連付けが 5 月に変更される状況を考えます。最初の 6 か月にわたる、顧客販売額および販売担当者の割当ては、次のようになります:

```
   Jan     Feb     Mar     Apr     May     Jun
5540    2190    1580    300     2455    3255
Jones   Jones   Jones   Jones   Smith   Smith
```

可変属性機能を使用すると、取得値には、Jones が顧客 A に$9610 (1 月、2 月および 3 月の合計)販売したこと、および Smith が$5680 (5 月および 6 月の合計)販売したことを反映できます。この機能がなければ、わかっている販売担当者は現在の担当者の Smith のみになり、売上のすべて($15290)が Smith によるものということになってしまいます。

可変属性の用語

用語	定義
独立次元	可変属性が依存する次元。前述の例では Year 次元。
パースペクティブ	関連付けをクエリーするときに使用される独立次元メンバーの組合せ。777 ページの「ESS_PERSPECTIVE_T」に定義されています。
妥当性セット	関連付けが TRUE である独立次元メンバーの集合。779 ページの「ESS_VALIDITYSET_T」に定義されています。

アウトライン構築

可変属性は、次のフローで API で構築されます:

アイテム	アウトライン API の呼出し
1. 可変属性を受け入れるように、アウトライン・タイプを設定します	EssOtlSetOutlineInfo、ここで pOutlineinfo->fEnableVaryingAttrs = ESS_TRUE。
2. 独立次元を識別します	EssOtlSetMemberInfo、ここで pMemberInfo->fIndependentDim = ESS_TRUE
3. 基本次元に属性次元を関連付けて、独立次元を識別します	EssOtlVaryingAssociateAttributeDimension
4. 属性次元メンバーおよび独立次元メンバーを基本次元メンバーと関連付けます	EssOtlVaryingAssociateAttribute
5. アウトラインを保存し再構築します。	アウトラインへの他の変更を行うときと同様です。

保守タスク

アイテム	アウトライン API の呼出し
独立メンバーに新しい関連付けを追加します。	EssOtlVaryingAssociateAttribute
独立メンバーの関連付けを削除します	EssOtlVaryingDisassociateAttribute
既存の独立次元メンバーの関連付けを表示します	EssOtlQueryVaryingAttributes または EssOtlVaryingGetAssociatedAttributes
属性次元の関連付けを基本次元から解除します	EssOtlDisassociateAttributeDimension(属性次元すべての関連付けを解除します)。

C のアウトライン API 関数のリファレンス

「目次」ペインで、EssOtl が前に付いた C のアウトライン API 関数のリストを参照してください

EssOtlAddDimension

アウトラインに次元を追加し、メンバーの属性を設定します。また、この呼出しは、アウトラインが再構成されるときに、データを関連付ける新しい次元メンバーを指定します。

構文

```
ESS_FUNC_M
EssOtlAddDimension
 (
hOutline, pMemberInfo, hPrevSibling, pszDataMbr, phMember
);
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pMemberInfo;	768 ページの「ESS_MBRINFO_T」	メンバーとその属性を定義しているメンバー情報構造体。
hPrevSibling	ESS_HMEMBER_T	前の兄弟のハンドル。このフィールドが ESS_NULL である場合、次元は、アウトラインの最初の次元になります。それ以外の場合、次元は hPrevSibling で指定された次元の後に配置されます。
pszDataMbr;	ESS_STR_T	アウトラインが再構成されるときにデータ値を受領する、新規次元のメンバーのメンバー名。このフィールドが ESS_NULL である場合、次元メンバー自身が使用されます。
phMember	ESS_PHMEMBER_T	API から戻された新規メンバーのハンドル。

備考

- この関数を呼び出す前に、ESS_MBRINFO_T 構造体を作成し、値を入れておく必要があります。
- 属性次元を追加するには、この関数を呼び出す必要があります。
- 属性次元でない次元を追加するには、この関数または **EssOtlAddMember()**を呼び出します。
  - EssOtlAddDimension()によって、追加された次元メンバーを選択し、既存の次元と関連するデータ値にそれを割り当てることができます。
  - **EssOtlAddMember()**を使用する場合は、追加された次元の上位メンバー(次元)が使用されます。
- pszDataMbr フィールドを有効にするには、fKeepTrans フラグを ESS_TRUE に設定した **EssOtlOpenOutline()**を使用して、アウトラインを開いておく必要があります。
- pszDataMbr フィールドで参照されたメンバーは、次元が作成された後、**EssOtlAddMember()**を使用して、新しい次元に追加されます。再構成のときに参照されたメンバーが存在しない場合、次元メンバーがそのかわりに使用されます。
- 属性次元に対して、ESS_MBRINFO_T のフィールドを次のように設定する必要があります:

フィールド	設定
usConsolidation	ESS_UCALC_NOOP
fTwoPass	ESS_FALSE
fExpense	ESS_FALSE
usConversion	ESS_CONV_NONE
usTimeBalance	ESS_TIMEBAL_NONE
usSkip	ESS_SKIP_NONE
usShare	ESS_SHARE_DYNCALCNOSTORE
usStorage	ESS_DIMTYPE_SPARSE
usCategory	ESS_CAT_ATTRIBUTE
usStorageCategory	ESS_STORECAT_ATTRIBUTE
Attribute.usDataType	次の属性メンバー・データ型のいずれかになります: ❍ ESS_ATTRMBRDT_BOOL ❍ ESS_ATTRMBRDT_DATETIME ❍ ESS_ATTRMBRDT_DOUBLE ❍ ESS_ATTRMBRDT_STRING

- 属性次元に基本次元を関連付ける必要があります。
- 属性次元は、基本次元および標準次元の後ろに置く必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME
- OOTLAPI_ERR_ADDNAMEUSEDTLAPI_ERR_ADDDELETEDIMDYNAMICCALC
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE

- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NONATTRDIMFOLLOWED
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_OUTLINEINFO_T  NewInfo;
ESS_HOUTLINE_T   hOutline;
ESS_MBRINFO_T    MbrInfo;
ESS_HMEMBER_T    hDimMeasures;

memset (&NewInfo,'\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo,
   &hOutline);
if (!sts)
{
memset(&MbrInfo, '\0', sizeof(MbrInfo));
strcpy(MbrInfo.szMember, "Measures");
MbrInfo.usStorage = ESS_DIMTYPE_SPARSE;
MbrInfo.usCategory = ESS_CAT_ACCOUNTS;
sts = EssOtlAddDimension(hOutline, &MbrInfo,
ESS_NULL, "Profit",&hDimMeasures);
}
```

関連トピック

- EssOtlAddMember
- EssOtlDeleteDimension
- EssOtlDeleteMember
- EssOtlGetMemberInfo

EssOtlAddQueryHint

ビュー選択を支援するために、アウトラインにクエリー・ヒントを追加します。

ヒントには 1 から n までの番号が付けられます。最初のクエリー・ヒントのヒント番号は 1 です。新しい各クエリー・ヒントがリストの終わりへ追加され、番号は 1 ずつ大きくなります。

構文

```
ESS_FUNC_M EssOtlAddQueryHint (
hOutline, numMembers, pMemberArray
```

```
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
numMembers	ESS_SHORT_T	提供された配列のメンバーの数 – 通常は、アウトラインの実次元の数。(入力)
pMemberArray	ESS_PHMEMBER_T	ヒント用のメンバーの配列。通常、配列には、実次元当たり 1 つのメンバーを持ち、NULL がヒントの一部でない次元に使用されます。この配列は割り当てる必要があります。

備考

- 競合が発生したときに、レベル使用制約がクエリー・ヒントより優先されます(SetAggLevelUsage を参照)。
- ヒントには、動的メンバー、ラベルのみメンバー、または共有メンバーは含まれない場合があります。
- アウトラインが変化すると、ヒントは無効になる場合があります。無効なヒントにより警告メッセージが発生します。
- 共通クエリーのプロファイルについて Essbase に通知することにより、クエリー・ヒントは標準ビュー選択に影響を及ぼすことができます。
- この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。
- クエリー・ヒントは、MDX タプルとして書かれており、指定された各次元からのメンバーを 1 つのみ含みます。
- クエリー・ヒントで使用される各メンバーは、代表的なメンバーと考えられます。Essbase サーバーは、「このメンバーまたは類似した集約レベルのメンバー」として、代表的なメンバーを解釈します。たとえば、Sample Basic で `(Qtr1, Sales, 100, East, Actual)`のクエリー・ヒントを使用すれば、四半期ごとに、レベル 1 マーケットでのレベル 1 製品の実績利益率のメジャーが、クエリーの一般タイプになります。
- ある指定した次元について、Essbase サーバーは、代表的なメンバーの省略を、次元からのメンバーをクエリーで使用できると解釈します。たとえば、Sample Basic で`(Sales, 100, East)`のクエリー・ヒントを使用すると、省略されている Year および Scenario 次元に関係なく、レベル 1 マーケットでのレベル 1 製品の利益率のメジャーが、クエリーの一般タイプになります。ヒント `(Sales, 100, East)`は、`(NULL, Sales, 100, East, NULL)`と同一のものとして扱われます。

戻り値

成功の場合、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_HMEMBER_T   hMember1 = ESS_NULL;
```

```
 ESS_HMEMBER_T   hMember2 = ESS_NULL;
 ESS_HMEMBER_T   hMember3 = ESS_NULL;
 ESS_HMEMBER_T   hMember[3];
 ESS_SHORT_T    nmMembers = 3;

 /* code to assign hOutline variable omitted */
 /* code to assign hMember1 variable to member "Sales" omitted */
 /* code to assign hMember2 variable to member "100" omitted */
 /* code to assign hMember3 variable to member "East" omitted */
 hMember[0] = hMember1;
 hMember[1] = hMember2;
 hMember[2] = hMember3;

 if (hOutline)
 {
   sts = EssOtlAddQueryHint(hOutline, nmMembers, hMember);
if (sts)
 printf("Error (%ld) adding QueryHint\n", sts);
 }
 else
 {
   if (!hOutline)
    printf("Outline not provided\n");
 }
```

関連トピック

- EssOtlSetQueryHint
- EssOtlGetQueryHint
- EssOtlGetNumQueryHints
- EssOtlGetQueryHintSize
- EssOtlDeleteQueryHint

EssOtlAddMember

アウトラインにメンバーを追加し、メンバーの属性を設定します。

構文

```
ESS_FUNC_M
EssOtlAddMember
 (
hOutline, pMemberInfo, hParent, hPrevSibling, phMember
);
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pMemberInfo;	768 ページの「ESS_MBRINFO_T」	メンバーとその属性を定義しているメンバー情報構造体。
hParent;	ESS_HMEMBER_T	親のハンドル。このフィールドは、hPrevSibling フィールドが ESS_NULL の場合にのみ使用されます。

パラメータ	データ型	説明
hPrevSibling;	ESS_HMEMBER_T	前の兄弟のハンドル。
phMember;	ESS_PHMEMBER_T	API から戻された新規メンバーのハンドル。

備考

- この関数を呼び出す前に、ESS_MBRINFO_T 構造体を作成し、値を入れておく必要があります。
- 共有メンバーを作成している場合以外、メンバー名は一意である必要があります。
- 追加されたメンバーの位置は次のようになります:
  - 新規メンバーは、hPrevSibling メンバーの後に挿入されます。
  - hPrevSibling フィールドが ESS_NULL である場合、新しいメンバーは hParent によって指定された親の最初の子になります。
  - hParent および hPrevSibling が ESS_NULL である場合、新しいメンバーはアウトラインの最初の次元になります。
- 共有メンバーを追加するには、次の条件に従います:
  - 共有メンバーはゼロレベル(リーフ・ノード)メンバーである必要があります。(共有メンバーは子を持つことができません。)
  - 実際のメンバーが、次元内にすでに存在している必要があります。
  - 構造体 ESS_MBRINFO_T の usShare フィールドを ESS_SHARE_SHARE に設定します。
- LABEL メンバーを追加するには、次の手順に従います:
  - 最初に、ラベル属性を設定せずにメンバーを追加します。
  - 次に、その子を追加します。(ラベル・メンバーには子が必要です。)
  - 次に、**EssOtlSetMemberInfo()**を使用して、ラベル・メンバーのラベル・タグを設定します。
- 属性メンバーを追加するには、ESS_MBRINFO_T のフィールドを次のように設定します:

フィールド	設定
usConsolidation	ESS_UCALC_NOOP
fTwoPass	ESS_FALSE
fExpense	ESS_FALSE
usConversion	ESS_CONV_NONE
usTimeBalance	ESS_TIMEBAL_NONE
usSkip	ESS_SKIP_NONE
usShare	ESS_SHARE_DYNCALCNOSTORE

フィールド	設定
usStorage	ESS_DIMTYPE_SPARSE
usCategory	ESS_CAT_ATTRIBUTE
usStorageCategory	ESS_STORECAT_ATTRIBUTE
Attribute.usDataType	属性次元またはゼロレベル(リーフ・ノード)の属性メンバーについては、次のデータ型のいずれかを設定します: ❍ ESS_ATTRMBRDT_BOOL ❍ ESS_ATTRMBRDT_DATETIME ❍ ESS_ATTRMBRDT_DOUBLE ❍ ESS_ATTRMBRDT_STRING そのかわりに、ESS_ATTRMBRDT_AUTO にゼロレベル(リーフ・ノード)属性メンバーを設定できます。 0 レベルでない属性メンバーは、ESS_ATTRMBRDT_NONE または ESS_ATTRMBR_AUTO に設定できます。

- ❍ **属性メンバーの追加に関する注意:**
  - ❑ ESS_ATTRMBRDT_STRING 型でないゼロレベルの属性メンバーが追加されると、ESS_MBRINFO_T 構造体の szMember フィールドは、125 ページの「ESS_ATTRSPECS_T」構造体のアウトラインの指定により、属性メンバーのロング名に設定されます。
  - ❑ 属性次元のみでなく属性メンバーにも usCategory および usStorageCategory を設定する必要があります。(基本メンバーには usCategory および usStorageCategory を設定する必要はありません。基本次元のみに対して設定する必要があります。)
  - ❑ ESS_MBRINFO_T 構造体の szDimName フィールドを設定しないでください。
  - ❑ タイプが ESS_ATTRMBRDT_STRING でない 0 レベル属性メンバーには、124 ページの「ESS_ATTRIBUTEVALUE_T」構造体の Attribute.value フィールドを設定しないでください。属性値は、属性メンバーのロング名を変換して内部で導出されます。
  - ❑ 属性メンバーのデータ型を ESS_ATTRMBRDT_AUTO に設定する場合、Essbase によって次のことが行われます:
    - メンバー名がその型の値に変換できる場合、メンバーのデータ型をその次元のデータ型に設定します。
    - メンバー名を次元のデータ型の値に変換できない場合は、メンバーのデータ型を ESS_ATTRMBRDT_NONE に設定します。
    - ESS_ATTRMBRDT_AUTO から ESS_ATTRMBRDT_NONE 以外のデータ型に変換された最初の子メンバーについては、親のロング名をショート名に変換します。
- 次元を追加する場合:

- 属性次元を追加するには、**EssOtlAddDimension()**を呼び出します。**EssOtlAddMember()**を呼び出さないでください。
- 属性次元でない次元を追加するには、**EssOtlAddDimension()**または**EssOtlAddMember()**を呼び出します。
  - **EssOtlAddDimension()**によって、追加された次元メンバーを選択し、既存の次元と関連するデータ値にそれを割り当てることができます。
  - **EssOtlAddMember()**が使用される場合は、追加された次元の上位メンバー(次元)が既存の次元と関連するデータ値に割り当てられます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME
- OTLAPI_ERR_ADDNAMEUSED
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE
- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_MBRINFO_T    MbrInfo;
ESS_HMEMBER_T    hMemberProfit;
ESS_HMEMBER_T    hNewMember;
ESS_APPNAME_T    szAppName;
```

```
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object,
 ESS_TRUE, ESS_TRUE, &hOutline);

if (!sts)
{
  sts = EssOtlFindMember(hOutline, "Profit",
     &hMemberProfit);
}
if (!sts && hMemberProfit)
{
  memset(&MbrInfo, '\0', sizeof(MbrInfo));
  strcpy(MbrInfo.szMember, "Inventory");
  sts = EssOtlAddMember(hOutline, &MbrInfo,
     ESS_NULL, hMemberProfit, &hNewMember);
}
```

関連トピック

- EssOtlAddDimension
- EssOtlDeleteMember
- EssOtlDeleteDimension
- EssOtlSetMemberInfo
- EssOtlFindMember

EssOtlAssociateAttributeDimension

属性次元を標準次元または基本次元に関連付けます。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
hStandardDimension;	ESS_HMEMBER_T	標準次元または基本次元のハンドル
hAttributeDimension;	ESS_HMEMBER_T	属性次元のハンドル

備考

- 属性次元は疎である必要があります。
- 標準または基本次元は疎である必要があります。
- 属性次元を標準次元または基本次元に関連付ける必要があります。
- 複数の属性次元を 1 つの基本次元に関連付けることができます。
- 1 つの属性次元を複数の基本次元に関連付けることはできません。

例

```
    void ESS_OtlAssociateAttributeDimension()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline;
 ESS_HMEMBER_T   hBaseMbr;
 ESS_HMEMBER_T   hAttrMbr;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_PROCSTATE_T pState;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Product", &hBaseMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Color", &hAttrMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlAssociateAttributeDimension(hOutline,hBaseMbr,hAttrMbr);
 printf("EssOtlAssociateAttributeDimension() sts: %ld\n",sts);

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
   sts = EssGetProcessState (hCtx, &pState);
 }
```

```
 sts = EssOtlCloseOutline(hOutline);
 printf("EssOtlCloseOutline() sts: %ld\n",sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlAssociateAttributeMember

属性メンバーを標準または基本メンバーに関連付けます。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
hStandardMember;	ESS_HMEMBER_T	標準メンバーまたは基本メンバーへのハンドル
hAttributeMember;	ESS_HMEMBER_T	属性メンバーのハンドル

備考

- この関数を使用して属性メンバーを標準メンバーまたは基本メンバーと関連付ける前に、**EssOtlAssociateAttributeDimension()**を使用して属性メンバーの次元を標準メンバーまたは基本メンバーの次元に関連付けてください。
- 属性メンバーは基本次元には関連付けられません。
- ゼロレベルの属性メンバーのみを標準または基本メンバーに関連付けられます。
- 与えられた属性次元のメンバーをレベルの異なる基本メンバーには関連付けられません。
- 1 つの属性次元の複数のメンバーを 1 つの基本メンバーに関連付けられません。

- 1 つの基本メンバーに 1 つ以上の属性次元のメンバーを関連付けられます。

例

```
    void ESS_OtlAssociateAttributeMember()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline;
 ESS_HMEMBER_T   hBaseMbr;
 ESS_HMEMBER_T   hAttrMbr;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_PROCSTATE_T pState;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Product", &hBaseMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Color", &hAttrMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlAssociateAttributeMember(hOutline,hBaseMbr,hAttrMbr);
 printf("EssOtlAssociateAttributeMember() sts: %ld\n",sts);

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 sts = EssOtlCloseOutline(hOutline);
 printf("EssOtlCloseOutline() sts: %ld\n",sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlClearAliasTable

既存の別名テーブルからすべてのエントリを消去します。別名テーブルは削除されません。

構文

```
ESS_FUNC_M
EssOtlClearAliasTable
(
hOutline, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable;	ESS_STR_T	消去する別名テーブルの名前。デフォルト・テーブルには ESS_NULL または"デフォルト"を使用します。

備考

別名テーブルから別名を消去すると、その別名テーブルに関連付けられた言語コードは削除されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_ALIASTABLE

例

```
    #include <essapi.h>
#include <essotl.h>
```

```
ESS_STS_T      sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlClearAliasTable(hOutline,
 "Default");
}
```

関連トピック

- EssOtlCreateAliasTable
- EssOtlCopyAliasTable
- EssOtlRenameAliasTable
- EssOtlDeleteAliasTable
- EssOtlSetAliasTableLanguage

EssOtlClearAliasTableLanguages

指定した別名テーブルに関連付けられた言語コードのセットが消去されます。

構文

```
ESS_FUNC_M
EssOtlClearAliasTableLanguages
 (
hOutline
,
pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable	ESS_STR_T	関連付けられた言語コードがすべて削除される別名テーブル名。

戻り値

- 成功の場合、0 が戻されます。
- 処理に失敗すると、エラー OTLAPI_BAD_ALIASTABLE (無効な別名テーブル) が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OUTLINEINFO_T NewInfo;
ESS_HOUTLINE_T   hOutline;
ESS_PALIASLANG_T pLangs=ESS_NULL;
ESS_ULONG_T    nLangs = 0, i=0;

memset(&NewInfo, '\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo, &hOutline);

if (!sts)
{
 sts = EssOtlCreateAliasTable(hOutline,
 "French Alias Table");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr-CA");
}

if (!sts)
{
    sts = EssOtlGetAliasTableLanguages(hOutline, "French Alias Table", &nLangs,
&pLangs);

 if ( !sts == ESS_STS_NOERR && ( pLangs) )
 {
   for (i=0;i<nLangs ;++i)
   {
     if (pLangs[i])
     {
       printf("Language Code:  %s\n", pLangs[i]);
     }
   }
   EssFree(hInst, pLangs);
```

```
 }
}
if (!sts)
{
 sts = EssOtlClearAliasTableLanguages (hOutline,
 "French Alias Table");
}
```

関連トピック

- EssOtlGetAliasTableLanguages
- EssOtlSetAliasTableLanguage

EssOtlCloseOutline

アウトラインに関連するすべての情報を解放します。

構文

```
ESS_FUNC_M
EssOtlCloseOutline
 (
hOutline
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。

備考

- この関数は、**EssOtlNewOutline()**または **EssOtlOpenOutline()**を呼び出す場合には必ず呼び出す必要があります。
- オブジェクトが開かれるときにロックされている場合、この呼出しを行う前に **EssUnlockObject()**を呼び出す必要があります。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
```

```
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/* body of code */
if (!sts)
{
 sts = EssOtlWriteOutline(hOutline, &Object);
}

/* restructure db using EssOtlRestructure() */
if (!sts)
{
 sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlOpenOutline
- EssOtlWriteOutline
- EssOtlRestructure

EssOtlCompactOutline

クライアント側でコンパクト化が必要なアウトライン・ファイルをコンパクト化します。

構文

```
ESS_FUNC_M
EssOtlCompactOutline
 (
hCtx, filename
);
```

パラメータ　データ型　　説明

hCtx	ESS_HCTX_T	ログイン時に取得した API コンテキスト
filename	ESS_STR_T	コンパクト化されるアウトライン・ファイルおよびパス

戻り値

成功の場合、0 が戻されます。コンパクト化されたファイルの名前は、元の名前に拡張子.otn が付加されたものになります。指定されたパスで使用可能です。

例

```
    #include <windows.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#pragma pack(push, api, 1)
#include <essapi.h>
#include <essotl.h>
#pragma pack(pop, api)

/* default names */
ESS_SVRNAME_T  srvrName    =    "localhost";
ESS_USERNAME_T userName    =    "essexer";
ESS_PASSWORD_T pswd      =    "password";
ESS_APPNAME_T  app       =    "ASOSamp";
ESS_DBNAME_T   db        =    "Sample";

int main(int argc, char *argv[ ])
{

    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_HINST_T hInst = NULL;
    ESS_HOUTLINE_T hOutlineQuery = NULL, hOutline = NULL;
    ESS_HCTX_T hCtx = NULL;
    ESS_USHORT_T Items;
    ESS_PAPPDB_T pAppsDbs = NULL;
    ESS_ACCESS_T Access;

    ESS_INIT_T InitStruct =    /* Define init */
                /* structure */
    {
        ESS_API_VERSION,    /* Version of API */
            (ESS_PVOID_T)0,   /* user-defined message context */
            0,     /* max handles   */
            0L,     /* max buffer size */
            NULL, //(ESS_STR_T)"C:\\Hyperion\\AnalyticServices", /* local path */
            /* The following parameters use defaults  */
            NULL,    /* message db path */
            NULL,    /* allocation function pointer */
            NULL,    /* reallocation function pointer */
            NULL,    /* free function pointer */
            NULL, //(ESS_PFUNC_T)MessageFunc,  /* error handling function pointer */
            NULL,    /* path name of user-defined */
            /* Application help file */
            0L    /* Reserved for internal use. */
            /* Set to NULL */
#ifdef AD_UTF8
            , ESS_API_UTF8
#endif
    };

    /* get appname and dbname from the argument list */
    if (argc < 6) {
        puts(" Usage: EssCompactOtl ServerName Userid Password AppName DbName\n");
```

```
        exit (0);
  }

    strcpy(srvrName, argv[1]);
  strcpy(userName, argv[2]);
    strcpy(pswd, argv[3]);
    strcpy(app, argv[4]);
    strcpy(db, argv[5]);

    /* Initialize the Essbase API */
    if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
    {
     printf("EssInit failure: %ld\n", sts);
     exit ((int) sts);
    }

    /* Login to Essbase */
    if ((sts = EssLogin (hInst, srvrName, userName, pswd, &Items, &pAppsDbs, &hCtx)) !
= ESS_STS_NOERR)
    {
     printf("EssLogin failure: %ld\n", sts);
     exit ((int) sts);
    }

    if(pAppsDbs)
        EssFree(hInst, pAppsDbs);

    /* Select the application */
    if ((sts = EssSetActive(hCtx, app, db, &Access)) != ESS_STS_NOERR)
    {
     printf("EssSetActive failure: %ld\n", sts);
     exit ((int) sts);
    }

    /* compact the outline and restructure */
    if ((sts = EssCompactOutline(hCtx)) != ESS_STS_NOERR)
    {
     printf("EssCompactOutline failure: %ld\n", sts);
     exit ((int) sts);
    }

    /* done, logout and terminate the api */
    if ((sts = EssLogout (hCtx)) != ESS_STS_NOERR)
    {
     printf("EssLogout failure: %ld\n", sts);
     exit ((int) sts);
    }

    if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
    {
        /* error terminating API */
        exit((int) sts);
    }

    return(0);
}
```

EssOtlCopyAliasTable

別名テーブルを他の別名テーブルにコピーします。

構文

```
ESS_FUNC_M
EssOtlCopyAliasTable
 (
hOutline, pszSourceAliasTable, pszDestAliasTable, fMerge
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszSourceAliasTable	ESS_STR_T	コピー元の別名テーブル名。このパラメータが ESS_NULL の場合、デフォルトの別名テーブルが使用されます。
pszDestAliasTable	ESS_STR_T	コピー先の別名テーブル名。pszSourceAliasTable 以外を指定する必要があります。
fMerge	ESS_BOOL_T	コピー元のファイルを既存のコピー先別名テーブルにマージする場合は、ESS_TRUE に設定します。コピー前にコピー先の別名テーブルを消去するには ESS_FALSE に設定します。

備考

- コピー先の別名テーブルが存在しない場合は、作成されます。コピー先の別名テーブルが存在する場合、fMerge フラグが ESS_TRUE に設定されていなければ、最初に消去されます。
- 単一のブロック・ストレージまたは集約ストレージ・データベース・アウトライン内の別名テーブルの最大数は(デフォルトのテーブルを含めて)32 です。
- 別名テーブルをコピーすると、別名テーブルに関連付けられている言語コードがコピーされた別名テーブルから削除されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_MAXALIASTABLES
- OTLAPI_ERR_ALIASTABLENAME
- OTLAPI_ERR_COPYALIASTABLE: コピー元とコピー先のテーブルが同一です。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
```

```
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T     szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlCopyAliasTable(hOutline, ESS_NULL,
 "Alias Table 2", ESS_TRUE);
}
```

関連トピック

- EssOtlCreateAliasTable
- EssOtlClearAliasTable
- EssOtlRenameAliasTable
- EssOtlDeleteAliasTable
- EssOtlSetAliasTableLanguage

EssOtlCreateAliasTable

アウトラインに空の別名テーブルを作成します。

構文

```
ESS_FUNC_M
EssOtlCreateAliasTable
 (
hOutline, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable	ESS_STR_T	作成する別名テーブル名。

備考

- Default という名前の別名テーブルは作成できません。これは、この名前を持つデフォルトの別名テーブルが常に存在するためです。

- 単一のブロック・ストレージまたは集約ストレージ・データベース・アウトライン内の別名テーブルの最大数は(デフォルトのテーブルを含めて)32 です。
- EssOtlSetAliasTableLanguage API を使用して、別名テーブルに対して複数の言語コードを指定できます。別名テーブルの作成時には、言語コードは指定されません。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

```
     OTLAPI_ERR_ALIASTABLEEXISTS
OTLAPI_ERR_MAXALIASTABLES
OTLAPI_ERR_ALIASTABLENAME
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OUTLINEINFO_T  NewInfo;
ESS_HOUTLINE_T   hOutline;

memset(&NewInfo, '\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo,
   &hOutline);

if (!sts)
{
 sts = EssOtlCreateAliasTable(hOutline,
 "Alias Table 1");
}
```

関連トピック

- EssOtlCopyAliasTable
- EssOtlRenameAliasTable
- EssOtlDeleteAliasTable
- EssOtlSetAliasTableLanguage

EssOtlCreateObject

指定されたオブジェクト・タイプと名前のオブジェクトを作成し、オブジェクト・ハンドルを戻します。

構文

```
     ESS_FUNC_M EssOtlCreateObject (
     hOutline
     ,
     objType
```

```
,
name
,
phObjHandle
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objType	ESS_OBJECT_TYPES	次のいずれかの値を持つオブジェクト・タイプ: ● OBJECT_SMARTLIST オブジェクト・タイプはテキスト・リスト(スマートリスト)/
name	ESS_STR_T	オブジェクトを特定する文字列
phObjHandle	ESS_PHOBJECT_T	作成されたオブジェクト・ハンドルを戻します。

戻り値

戻り値:

- 0 - 正常終了の場合

  ハンドルが phObjHandle のオブジェクトが作成される
- エラー番号 - 失敗した場合

  オブジェクトは作成されず、phObjHandle は NULL。
- OTLAPI_ERR_OBJTYPE_NOTSUPPORTED

  無効なオブジェクト・タイプが戻された場合。

例

```
    void TestCreateObject()
{
  ESS_STS_T      sts = ESS_STS_NOERR;
  ESS_HOUTLINE_T   hOutline = ESS_NULL;
  ESS_OBJDEF_T    Object;
  ESS_OBJECT_TYPES   objType;
  ESS_STR_T      smartListName;
  ESS_HOBJECT_T    ObjHandle;
  ESS_ULONG_T     Count, i;
  ESS_PHOBJECT_T   ObjHandles;
  ESS_HOBJECT_T       hObjHandle;
  ESS_HSMARTLIST_T  hSmartList;
  ESS_STR_T      objName;

  memset(&Object, '\0', sizeof(Object));
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  Object.AppName = szAppName;
  Object.DbName = szDbName;
  Object.FileName = szFileName;
```

```
  /* Open outline */
  sts = EssOtlOpenOutline(hCtx, &Object,
                                ESS_TRUE, ESS_TRUE, &hOutline);

  /* Create a static SmartList */
  objType = OBJECT_SMARTLIST;
  smartListName = "SList1";
  sts =
    EssOtlCreateObject(hOutline, objType,
                                smartListName, &ObjHandle);


  /* List all SmartList objects */
  objType = OBJECT_SMARTLIST;
  sts = EssOtlListObjects(hOutline, objType,
                                &Count, &ObjHandles);

        /* Free resources */
  if(ObjHandles)
    EssFree (hInst, ObjHandles);

  /* Save */
  SaveOutline(hOutline);

  /* Find objects */
  objName = "SList1";
  sts = EssOtlFindObject(hOutline, objType, objName,
                                          &hObjHandle);

   /* Delete objects */
  hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
  sts = EssOtlDeleteObject(hOutline, hSmartList);
  SaveOutline(hOutline);

  if(ObjHandles)
    EssFree (hInst, ObjHandles);

  /* Unlock objects */
  sts = EssUnlockObject(hCtx, Object.ObjType,
      Object.AppName, Object.DbName, Object.FileName);

        /* Close outline */
  sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList

- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlDeleteAliasTable

アウトラインから指定された別名テーブルを削除し、そのエントリをすべて消去します。

構文

```
ESS_FUNC_M
EssOtlDeleteAliasTable
 (
hOutline, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable	ESS_STR_T	削除する別名テーブル名。

備考

デフォルトの別名テーブルは削除できません。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

```
     OTLAPI_BAD_ALIASTABLE
OTLAPI_ERR_DELETEDEFALIAS
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
```

```
memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlDeleteAliasTable(hOutline,
 " Alias Table 1");
}
```

関連トピック

- EssOtlCreateAliasTable
- EssOtlCopyAliasTable
- EssOtlRenameAliasTable
- EssOtlClearAliasTable

EssOtlDeleteDimension

アウトラインから次元を削除します。また、この呼出しでは、アウトラインの再構築時にデータを保持しておく、削除対象の次元メンバーも指定します。

構文

```
ESS_FUNC_M
EssOtlDeleteDimension
 (
hOutline, hMember, pszDataMbr
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	削除するメンバーのハンドル。
pszDataMbr	ESS_STR_T	アウトラインの再構築時に保存するデータが含まれた次元メンバーの名前。このフィールドが ESS_NULL の場合は、次元が使用されます。

備考

- 次元およびその子孫のすべての共有メンバーが削除されます。
- 次元のすべてのメンバーが削除されます。

- 次元を削減するには、この関数、または **EssOtlDeleteMember()**を呼び出します。**EssOtlDeleteDimension()**では、データベースが再構築されたときにデータ値が別の次元で使用される削除された次元メンバーを選択できるという利点があります。**EssOtlDeleteMember()**を使用する場合、削除された次元の最上位メンバー(次元)のデータ値が使用されます。
- pszDataMbr フィールドを有効にするには、fKeepTrans フラグを ESS_TRUE に設定した **EssOtlOpenOutline()**を使用して、アウトラインを開いておく必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

```
     OTLAPI_ERR_ADDDELETEDIMDYNAMICCALC
OTLAPI_ERR_NOTIMEDIM
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T   hMemberScenario;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Scenario",
 &hMemberScenario);
}

if (!sts && hMemberScenario)
{
 sts = EssOtlDeleteDimension(hOutline,
 hMemberScenario, "Actual");
}
```

関連トピック

- EssOtlDeleteMember
- EssOtlAddDimension
- EssOtlAddMember
- EssOtlFindMember
- EssOtlGetMemberInfo

EssOtlDeleteDTSMemberAlias

動的時系列(Dynamic Time Series: DTS)メンバーの別名を削除します。

構文

```
ESS_STS_T
EssOtlDeleteDTSMemberAlias
 (
hOutline, pszDTSMember, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	EssOtlOpenOutline 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember	ESS_STR_T	別名を提供する DTS メンバー名。
pzsAliasTable	ESS_STR_T	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルを使用します。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_NOALIAS

例

```
    #include "essapi.h"
#include "essotl.h"
#include "esserror.h"

ESS_STS_T ESS_OtlDeleteDTSMemberAlias(ESS_HCTX_T hCtx)
{
    ESS_STS_T        sts =ESS_STS_NOERR;
    ESS_OBJDEF_T   Object;
    ESS_HOUTLINE_T  hOutline;
    ESS_APPNAME_T   szAppName;
    ESS_DBNAME_T   szDbName;
    ESS_OBJNAME_T   szFileName;
```

```
ESS_CHAR_T   pszAliasTable[ESS_ALIASNAMELEN];
ESS_CHAR_T   pszDTSMember[ESS_MBRNAMELEN];
ESS_PROCSTATE_T pState;
ESS_ULONG_T   ulErrors;
ESS_ULONG_T   ulCount;
ESS_POUTERROR_T pMbrErrors = NULL;

strcpy(szAppName, "sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
strcpy(pszDTSMember, "Q-T-D");
strcpy(pszAliasTable, "Default");

Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);

if(sts)
{
    printf("Could not open outline\n");
    return sts;
}

sts = EssOtlDeleteDTSMemberAlias(hOutline, pszDTSMember, pszAliasTable);
if(sts)
{
    printf("Could not get DTS member alias\n");
    return sts;
}

sts = EssOtlWriteOutline(hOutline, &Object);
if(sts)
{
    printf("Could not write outline\n");
    return sts;
}

sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
if(sts)
{
    printf("Could not restructure outline\n");
    return sts;
}

memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
sts = EssGetProcessState(hCtx, &pState);
{
    while ((sts == ESS_STS_NOERR ) && (pState.State != ESS_STATE_DONE))
    {
        memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
        sts = EssGetProcessState(hCtx, &pState);
    }
}
```

```
    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, szAppName, szDbName,
szFileName);
    if (sts)
    {
        printf("Could not unlock outline\n");
        return sts;
    }

    EssOtlCloseOutline(hOutline);
    return sts;
}
```

関連トピック

- EssOtlEnableDTSMember
- EssOtlGetEnabledDTSMembers
- EssOtlGetDTSMemberAlias
- EssOtlSetDTSMemberAlias

EssOtlDeleteGenName

次元内の特定の世代名を削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteGenName
 (
hOutline, pszDimension, usGen
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension;	ESS_STR_T	対象の世代を含む次元の名前。
usGen	ESS_USHORT_T	名前を削除する世代の番号。リーフ・メンバーはレベル 0 です。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

```
    OTLAPI_NO_GENLEVELNAME
OTLAPI_ERR_NOTADIM
```

例

```
    #include <essapi.h>
#include <essotl.h>
```

```
ESS_STS_T      sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T      Dimension;
ESS_USHORT_T    GenNum;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
/***********  Delete Generation Name  ***********/
Dimension = "Year";
GenNum = 2;
if (!sts)
{
 sts = EssOtlDeleteGenName(hOutline, Dimension,
    GenNum);
}
```

関連トピック

- EssOtlGetGenName
- EssOtlSetGenName

EssOtlDeleteLevelName

次元内の特定のレベルの名前を削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteLevelName
 (
hOutline, pszDimension, usLevel
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	レベル名を含む次元の名前。
usLevel	ESS_USHORT_T	名前を削除するレベルの番号。リーフ・メンバーはレベル 0 です。

備考

C プログラムでは、戻りバッファを解放する場合は EssFree()を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

```
     OTLAPI_NO_GENLEVELNAME
OTLAPI_ERR_NOTADIM
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    LevelNum;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName   = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/**********  Delete Level Name  *************/
Dimension = "Year";
LevelNum = 2;
if (!sts)
{
 sts = EssOtlDeleteLevelName(hOutline,
    Dimension, LevelNum);
}
```

関連トピック

- EssOtlGetLevelName
- EssOtlSetLevelName

EssOtlDeleteObject

渡されたオブジェクトを削除します。

構文

```
ESS_FUNC_M EssOtlDeleteObject (
hOutline
,
objHandle
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objHandle	ESS_HOBJECT_T	削除されるオブジェクト

備考

既存の関連付けがあるオブジェクトを削除できません。テキスト・リスト・オブジェクト(スマートリスト・オブジェクト)では、リファレンスを削除せずにスマートリスト・オブジェクトを削除することはできません。このためには、Get Object References API を使用します。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestCreateObject()
{
  ESS_STS_T      sts = ESS_STS_NOERR;
  ESS_HOUTLINE_T    hOutline = ESS_NULL;
  ESS_OBJDEF_T     Object;
  ESS_OBJECT_TYPES   objType;
  ESS_STR_T      smartListName;
  ESS_HOBJECT_T    ObjHandle;
  ESS_ULONG_T     Count, i;
  ESS_PHOBJECT_T    ObjHandles;
  ESS_HOBJECT_T    hObjHandle;
  ESS_HSMARTLIST_T  hSmartList;
  ESS_STR_T      objName;

  memset(&Object, '\0', sizeof(Object));
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  Object.AppName = szAppName;
  Object.DbName = szDbName;
  Object.FileName = szFileName;
```

```
  /* Open outline */
  sts = EssOtlOpenOutline(hCtx, &Object,
                                ESS_TRUE, ESS_TRUE, &hOutline);

  /* Create a static SmartList */
  objType = OBJECT_SMARTLIST;
  smartListName = "SList1";
  sts = EssOtlCreateObject(hOutline, objType,
                                smartListName, &ObjHandle);

  /* List all SmartList objects */
  objType = OBJECT_SMARTLIST;
  sts = EssOtlListObjects(hOutline, objType,
                                &Count, &ObjHandles);

  /* Save */
  SaveOutline(hOutline);

  /* Find objects */
  objName = "SList1";
  sts = EssOtlFindObject(hOutline, objType, objName,
                                          &hObjHandle);

   /* Delete objects */
  hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
  sts =
    EssOtlDeleteObject(hOutline, hSmartList);

  SaveOutline(hOutline);

  if(ObjHandles)
    EssFree (hInst, ObjHandles);

  /* Unlock objects */
  sts = EssUnlockObject(hCtx, Object.ObjType,
      Object.AppName, Object.DbName, Object.FileName);

        /* Close outline */
  sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount

- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlDeleteMember

アウトラインからメンバーを削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteMember
(
hOutline, hMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	削除するメンバーのハンドル。

備考

- メンバーのすべての子孫が削除されます。
- メンバーおよびその子孫のすべての共有メンバーが削除されます。
- 共有メンバーの場合、指定されたメンバーのみ削除されます。
- 次元を削除するには、この呼出しまたは **EssOtlDeleteDimension()**を使用します。**EssOtlDeleteDimension()**では、データベースの再構築時に他の次元に使用されるデータ値を含むメンバーを、削除された次元から選択できるという利点があります。**EssOtlDeleteMember()**を使用する場合、削除された次元の最上位メンバー(次元)のデータ値が使用されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

```
OTLAPI_ERR_LEAFLABEL
OTLAPI_ERR_NOTIMEDIM
```

例

```
#include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
```

```
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hCOGS;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "COGS", &hCOGS);
}

if (!sts && hCOGS)
{
 sts = EssOtlDeleteMember(hOutline, hCOGS);
}
```

関連トピック

- EssOtlDeleteDimension
- EssOtlAddMember
- EssOtlAddDimension
- EssOtlFindMember
- EssOtlGetMemberInfo

EssOtlDeleteMemberAlias

指定された別名テーブルの指定されたメンバーに対する、デフォルトのメンバー別名を削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteMemberAlias
 (
hOutline, hMember, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	別名を削除するメンバーのハンドル。
pszAliasTable	ESS_STR_T	別名を削除する別名テーブル。このパラメータが ESS_NULL の場合、デフォルトのテーブルが使用されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

```
OTLAPI_ERR_NOALIAS
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberJan;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}
if (!sts && hMemberJan)
{
 sts = EssOtlDeleteMemberAlias(hOutline,
    hMemberJan, ESS_NULL);
}
```

関連トピック

- EssOtlGetMemberAlias
- EssOtlSetMemberAlias

EssOtlDeleteMemberFormula

指定されたメンバーの式を削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteMemberFormula
 (
hOutline, hMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	メンバーのハンドル。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

```
OTLAPI_ERR_NOFORMULA
```

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMember;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline,
    "Variance", &hMember);
```

```
}

if (!sts && hMember)
}
 sts = EssOtlDeleteMemberFormula(hOutline,
    hMember);
}
```

関連トピック

- EssOtlSetMemberFormula
- EssOtlGetMemberFormula

EssOtlDeleteQueryHint

入力アウトラインおよびヒント・メンバーによって示されたクエリー・ヒントを削除します。

ヒントには 1 から n までの番号が付けられます。この関数は指定されたクエリー・ヒントを削除し、ヒントの番号を 1 つ下げます。削除されたヒントより大きな hintNum を持つヒントはすべて、hintNum -1 に番号が付け直されます。

構文

```
ESS_FUNC_M EssOtlDeleteQueryHint (
hOutline, hintNum
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hintNum	ESS_SHORT_T	クエリー・ヒント番号(入力)。

備考

- 共通クエリーのプロファイルについて Essbase に通知することにより、クエリー・ヒントは標準ビュー選択に影響を及ぼすことができます。
- この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。

戻り値

正常終了の場合は、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_PMBRINFO_T  pMemberInfo = ESS_NULL;
ESS_SHORT_T    nmHints = 0;
ESS_SHORT_T    i, j, hintNum;
ESS_HMEMBER_T   hMember[10]; /* (nm real dimensions) < 10 */
```

```
/* Code to assign hOutline variable omitted */
/* Code to assign hintNum variable omitted */

sts = EssOtlGetNumQueryHints(hOutline, &nmHints);
if (sts) return sts;  /* error out */

if (hintNum <= nmHints)
{
 sts = EssOtlDeleteQueryHint(hOutline, hintNum);
 if (sts)
  printf("Error [%s] deleting query hint (%d)\n", sts, hintNum);
 else
  printf("Query-Hint number: (%d) deleted\n", hintNum);
}
else
{
 printf("Query-Hint number: (%d) does not exist\n", hintNum);
}
```

関連トピック

- EssOtlAddQueryHint
- EssOtlSetQueryHint
- EssOtlSetQueryHint
- EssOtlGetNumQueryHints
- EssOtlGetQueryHintSize

EssOtlDeleteUserAttribute

メンバーのユーザー定義属性を削除します。

構文

```
ESS_FUNC_M
EssOtlDeleteUserAttribute
 (
hOutline, hMember, pszString
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	削除中の属性を含むメンバーのハンドル
pszString	ESS_STR_T	ユーザー属性の文字列。

備考

呼出し元は、属性を識別するために文字列で値を渡します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_NO_USERATTR。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_HMEMBER_T    hMember;
ESS_STR_T     AttributeList;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/******** Delete User Attributes *********/

AttributeList = "Read Write";

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMember);
}

if (!sts && hMember)
{
 sts = EssOtlDeleteUserAttribute(hOutline,
    hMember, AttributeList);
}
```

関連トピック

- EssOtlGetUserAttributes
- EssOtlSetUserAttribute

EssOtlDetailQueryAttributes

様々な属性に固有ではなく、EssOtlQueryVaryingAttributes と同様ですが、結果は pphDetailMemberArray の特定の関連する属性を提供します。

構文

```
ESS_FUNC_M EssOtlDetailQueryAttributes (
   ESS_HOUTLINE_T              hOutline,
    ESS_PATTRIBUTEQUERY_T     pAttrQuery,
     ESS_PMBRCOUNTS_T              pCount,
     ESS_PPHMEMBER_T             pphReturnedMemberArray,
    ESS_PPHMEMBER_T             pphDetailMemberArray)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)
pAttrQuery	ESS_PATTRIBUTEQUERY_T	クエリーを定義する構造体へのポインタ
pCount	ESS_PMBRCOUNTS_T	戻される基本メンバーの数へのポインタ
pphReturnedMemberArray	ESS_PPHMEMBER_T	戻されたメンバーのハンドルの配列へのポインタ
pphDetailMemberArray	ESS_PPHMEMBER_T	メンバー詳細の配列へのポインタ

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

関連トピック

- EssOtlQueryVaryingAttributes

EssOtlDetailQueryVaryingAttributes

EssOtlQueryVaryingAttributes に似ていますが、結果により pphDetailMemberArray に特定の関連属性が提供される点が異なります。

構文

```
ESS_FUNC_M EssOtlDetailQueryVaryingAttributes(
   ESS_HOUTLINE_T                          hOutline,
   ESS_PVARYING_ATTRIBUTEQUERY_T     pAttrQuery,
   ESS_PPERSPECTIVE_T                      pPerspective,
   ESS_PMBRCOUNTS_T                          pCount,
   ESS_PPHMEMBER_T                         pphMembers,
   ESS_PPHMEMBER_T                         pphDetailMemberArray,
   ESS_USHORT_T                              usValiditySetType,
   ESS_PVALIDITYSET_T                      **pppValiditySets);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
pAttrQuery	ESS_PVARYING_ATTRIBUTEQUERY_T	クエリーを定義する構造体へのポインタ
pPerspective	ESS_PPERSPECTIVE_T	クライアントまたはサーバーで関連付けをクエリーする際に使用される、独立したメンバーの集合へのポインタ
pCount	ESS_PMBRCOUNTS_T	戻される基本メンバーの数へのポインタ
pphMembers	ESS_PPHMEMBER_T	属性メンバーのハンドルの配列へのポインタ
pphDetailMemberArray	ESS_PPHMEMBER_T	属性メンバーの詳細ハンドルの配列へのポインタ
usValiditySetType	ESS_USHORT_T	779 ページの「ESS_VALIDITYSET_T」を参照してください。
**pppValiditySets	ESS_PVALIDITYSET_T	妥当性セットの配列へのポインタ

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestEssOtlDetaileQueryVaringAttributes()
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T    hOutline = ESS_NULL;
    ESS_OBJDEF_T    Object;
    ESS_USHORT_T    i = 0;
    ESS_PMBRINFO_T    pMbrInfo = ESS_NULL;
    ESS_VARYING_ATTRIBUTEQUERY_T    pAttrQuery;
    ESS_MBRCOUNTS_T    Counts;
    ESS_USHORT_T    usValiditySetType;
    ESS_HMEMBER_T    hIndepMbrHandlesArray[4];
    ESS_PERSPECTIVE_T    Perspective;
    ESS_PHMEMBER_T    phMbrHandles = ESS_NULL;
    ESS_PHMEMBER_T    phDetailedMembers = ESS_NULL;
    ESS_PVALIDITYSET_T    *pValiditySets = ESS_NULL;
    ESS_HMEMBER_T    hAttrMbr, hBaseMbr;
    ESS_HMEMBER_T    hAttrDim;
    ESS_PREDICATE_T    Predicate;

    memset(&Object, '\0', sizeof(ESS_OBJDEF_T));
    memset(&Counts, '\0', sizeof(ESS_MBRCOUNTS_T));
    memset(&pAttrQuery, 0x00, sizeof(ESS_ATTRIBUTEQUERY_T));
    memset(&Predicate, '\0', sizeof(ESS_PERSPECTIVE_T));

    Object.hCtx = hCtx;
```

```
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;
    sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
    printf("EssOtlOpenOutlineQuery sts: %ld\n",sts);

    Counts.ulStart = 0;
    Counts.ulMaxCount = 10;

    Predicate.ulQuery = ESS_SEARCH;
    Predicate.ulOptions = ESS_MEMBERSONLY;
    Predicate.pszDimension  = "";
    Predicate.pszString2 = "";

    /* Get handles for attribute member and dimension */
    Predicate.pszString1 = "Type";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hAttrDim = phMbrHandles[0];
    Predicate.pszString1 = "Contractor";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hAttrMbr = phMbrHandles[0];

    Predicate.pszString1 = "Doe,Jane";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hBaseMbr = phMbrHandles[0];

    /* Get handles for independent members */
    Predicate.pszString1 = "Jan";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[0] = phMbrHandles[0];
    hIndepMbrHandlesArray[2] = phMbrHandles[0];

    Predicate.pszString1 = "FY03";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[1] = phMbrHandles[0];
    Predicate.pszString1 = "FY04";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[3] = phMbrHandles[0];

    memset(&Perspective, '\0', sizeof(ESS_PERSPECTIVE_T));
    Perspective.usValiditySetType = ESS_VALIDITYSET_TYPE_MBRHDLS;
    Perspective.countOfIndepDims = 2;
    Perspective.countOfIndepRanges = 1;
    Perspective.pIndepMbrs = hIndepMbrHandlesArray;

    /* Query by handle with InputMemberType of ESS_ATTRIBUTE_MEMBER and
OutputMemberType of ESS_BASE_MEMBER*/
    printf("\n*** Query by handle with InputMemberType of ESS_ATTRIBUTE_MEMBER and
OutputMemberType of ESS_BASE_MEMBER:\n");
    pAttrQuery.bInputMemberIsHandle = ESS_TRUE;
```

```
    pAttrQuery.uInputMember.hMember = hAttrMbr;
    pAttrQuery.usInputMemberType = ESS_ATTRIBUTE_MEMBER;
    pAttrQuery.usOutputMemberType = ESS_BASE_MEMBER;
    pAttrQuery.Attribute.usDataType = ESS_ATTRMBRDT_NONE;
    pAttrQuery.usOperation = ESS_ALL;

    usValiditySetType = ESS_VALIDITYSET_TYPE_MBRHDLS;
    sts = EssOtlDetailQueryVaryingAttributes(hOutline, &pAttrQuery, &Perspective,
&Counts,
                      &phMbrHandles, &phDetailedMembers, usValiditySetType,
&pValiditySets);
    printf("EssOtlDetailQueryVaryingAttributes sts: %d\n", sts);
    if (!sts)
    {
        if (phMbrHandles)
        {
            printf("\tReturned member:\n");
            GetMemberInfo(hOutline, Counts, phMbrHandles);
            if(Counts.ulReturnCount && phMbrHandles)
                sts = EssOtlFreeMembers(hOutline, Counts.ulReturnCount,
phMbrHandles);
        }

        if(phDetailedMembers)
        {
            printf("\tAssociated attribute member:\n");
            GetMemberInfo(hOutline, Counts, phDetailedMembers);
            if (Counts.ulReturnCount && phDetailedMembers)
                sts = EssOtlFreeMembers(hOutline, Counts.ulReturnCount,
phDetailedMembers);
        }
    }

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);

    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlQueryVaryingAttributes

EssOtlDisassociateAttributeDimension

属性次元と基本次元との関連付けを解除します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル

パラメータ	データ型	説明
hBaseDimension;	ESS_HMEMBER_T	基本次元のハンドル
hAttributeDimension;	ESS_HMEMBER_T	属性次元のハンドル

備考

属性次元の基本次元との関連付けを解除すると、属性次元のすべてのメンバーと基本次元のメンバーとの関連付けが解除されます。

例

```
    void ESS_OtlDisassociateAttributeDimension()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline;
 ESS_HMEMBER_T   hBaseMbr;
 ESS_HMEMBER_T   hAttrMbr;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_PROCSTATE_T pState;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Product", &hBaseMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Color", &hAttrMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlDisassociateAttributeDimension(hOutline,hBaseMbr,hAttrMbr);
 printf("EssOtlDisassociateAttributeDimension() sts: %ld\n",sts);

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
```

```
    sts = EssGetProcessState (hCtx, &pState);
  }
  sts = EssOtlCloseOutline(hOutline);
  printf("EssOtlCloseOutline() sts: %ld\n",sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlDisassociateAttributeMember

属性メンバーと基本メンバーとの関連付けを解除します。

構文

```
ESS_FUNC_M
EssOtlDisassociateAttributeMember
 (
hOutline
,
hBaseMember
,
hAttributeMember
);
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
hBaseMember;	ESS_HMEMBER_T	基本メンバーのハンドル
hAttributeMember;	ESS_HMEMBER_T	属性メンバーのハンドル

備考

属性次元の基本次元との関連付けを解除すると、属性次元のすべてのメンバーと基本次元メンバーとの関連付けが解除されます。

例

```
    void ESS_OtlDisassociateAttributeMember()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline;
 ESS_HMEMBER_T   hBaseMbr;
 ESS_HMEMBER_T   hAttrMbr;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_PROCSTATE_T pState;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Product", &hBaseMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "Color", &hAttrMbr);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlDisassociateAttributeMember(hOutline,hBaseMbr,hAttrMbr);
 printf("EssOtlDisassociateAttributeMember() sts: %ld\n",sts);

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 sts = EssOtlCloseOutline(hOutline);
 printf("EssOtlCloseOutline() sts: %ld\n",sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure

- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlEnableDTSMember

アウトラインに対して新規 DTS メンバーを使用可能にします。

構文

```
      ESS_FUNC_M
      EssOtlEnableDTSMember
       (
      hOutline, pszDTSMember,
usGen, bEnable
      );
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	EssOtlOpenOutline 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember;	ESS_STR_T	DTS メンバーの名前
usGen;	ESS_USHORT _T	DTS メンバーに割り当てる世代
bEnable;	ESS_BOOL_T	DTS メンバーを使用可能にするフラグ

備考

この関数は、渡された ESS_DTSMBRNAME_T 構造体にも値を入れます。

戻り値

正常終了の場合は 0 が戻されます。

例

```
    #include "essapi.h"
#include "essotl.h"
#include "esserror.h"
ESS_STS_T ESS_OtlEnableDTSMember(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
```

```
ESS_STS_T    sts =ESS_STS_NOERR;
ESS_HOUTLINE_T hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_PROCSTATE_T pState;
ESS_ULONG_T   ulErrors;
ESS_ULONG_T   ulCount;
ESS_POUTERROR_T pMbrErrors = NULL;

strcpy(szAppName, "1Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");

memset(&Object, '\0', sizeof(ESS_OBJDEF_T));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
if(sts)
{
    printf("Could not open outline\n");
    return sts;
}

sts = EssOtlEnableDTSMember(hOutline, "H-T-D", 1, ESS_TRUE);
if(sts)
{
    printf("Could not enable DTS member alias\n");
}

sts = EssOtlVerifyOutline(hOutline, &ulErrors, &ulCount, &pMbrErrors);
if(sts)
{
    printf("Could not verify outline\n");
    return sts;
}

sts = EssOtlWriteOutline(hOutline, &Object);
if(sts)
{
    printf("Could not write outline\n");
    return sts;
}

sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
if(sts)
{
    printf("Could not restructure outline\n");
    return sts;
}

memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
```

```
    sts = EssGetProcessState(hCtx, &pState);
    {
        printf("sts from Proc State is %d and ProcState is %d\n", sts, pState.State);
        while ((sts == ESS_STS_NOERR) && (pState.State != ESS_STATE_DONE))
        {
            memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
            sts = EssGetProcessState(hCtx, &pState);
        printf("sts from Proc State is %d and ProcState is %d\n", sts, pState.State);
        }
    }

    EssUnlockObject(hCtx, Object.ObjType, Object.AppName, Object.DbName,
Object.FileName);
    EssOtlCloseOutline(hOutline);
    return sts;
}
```

関連トピック

- EssOtlDeleteDTSMemberAlias
- EssOtlGetEnabledDTSMembers
- EssOtlGetDTSMemberAlias
- EssOtlSetDTSMemberAlias

EssOtlFindAlias

指定した別名を持つメンバーを検索し、そのメンバーにハンドルを戻します。

構文

```
ESS_FUNC_M
EssOtlFindAlias
 (
hOutline, pszAlias, pszAliasTable, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAlias	ESS_STR_T	検索対象の別名。簡単な別名または修飾された別名(同じ名前のメンバーと区別)です。修飾された別名を指定するために使用する構文についての情報は、『Oracle Essbase データベース管理者ガイド』の重複するメンバー・アウトラインの作成および使用に関する項を参照してください。
pszAliasTable	ESS_STR_T	検索する別名テーブル。ESS_NULL を使用すると、すべての別名テーブルが検索されます。デフォルトの別名テーブルを検索するには、"Default"を使用します。
phMember	ESS_PHMEMBER_T	メンバー・ハンドルを戻す変数。メンバーが検出されない場合は ESS_NULL になります。

備考

別名の組合せで使用されている別名も検索されます。

戻り値

成功の場合、0 が戻されます。メンバーが見つからなかった場合は、*phMember が ESS_NULL に設定され、呼出しから 0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>
ESS_STS_T  sts = 0;
ESS_OBJDEF_T  Object;
ESS_HOUTLINE_T  hOutline;
ESS_HMEMBER_T  hMemberAlias;
ESS_APPNAME_T     szAppName;
ESS_DBNAME_T     szDbName;
ESS_OBJNAME_T     szFileName;
memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
  /* search all alias tables */
  sts = EssOtlFindAlias(hOutline, "Colas",
     ESS_NULL, &hMemberAlias);
}
```

関連トピック

- EssOtlGetOutlineInfo
- EssOtlGetMemberAlias

EssOtlFindAttributeMembers

指定されたショート名を持つすべての属性メンバーを戻します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
pszMember;	ESS_STR_T	属性のショート名

パラメータ	データ型	説明
pszDimName;	ESS_STR_T	属性次元名(オプション)
pusCount;	ESS_PUSHORT_T	戻された基本メンバーの数
pphMembers;	ESS_PPHMEMBER_T	基本メンバーのハンドルの配列へのポインタ

備考

- pszMember は、ショート名である必要があります。
- pszDimName はオプションです。NULL を入力できます。

例

```
    void  ESS_OtlFindAttributeMembers()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_SHORT_T   index;
 ESS_USHORT_T   count;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T  szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T  szFileName;
 ESS_HOUTLINE_T  hOutline;
 ESS_PPHMEMBER_T  phMember;
 ESS_PPMBRINFO_T  phMemberInfo;
 ESS_MBRNAME_T  mbrName;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 /* Returning an array of member handles? */
 sts = EssOtlFindAttributeMembers(hOutline,"12", "", &count, &phMember);
 /* sts = EssOtlFindAttributeMembers(hOutline,"10-01-1996", "", &count, &phMember);
*/
 printf("EssOtlFindAttributeMembers() sts: %ld\n",sts);
 /* Allocate memory for an array of memberinfo struct handles */
 sts = EssAlloc(hInst,count * (sizeof(ESS_HMEMBER_T)), (ESS_PPVOID_T)&phMemberInfo);
 if (!sts)
 {
  for(index = 0; index < count; index++)
  {
   /* Step through array of member handles, and assign member */
   sts = EssOtlGetMemberInfo(hOutline,phMember[index],&phMemberInfo[index]);
```

```
    printf("EssOtlGetMemberInfo() sts: %ld\n",sts);
    strcpy(mbrName,phMemberInfo[index]->szMember);
    printf("Attribute member name #%d is: %s\n",(index + 1),mbrName);
   }
   EssFree(hInst, phMember);
   EssFree(hInst, phMemberInfo);
  }
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlFindMember

指定した名前を持つメンバーを検索し、そのメンバーにハンドルを戻します。

構文

```
ESS_FUNC_M
EssOtlFindMember
 (
hOutline, pszMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszMember	ESS_STR_T	検索対象のメンバー名。簡単なメンバー名または修飾メンバー名(同じ名前のメンバーと区別)です。修飾メンバー名を指定するために使用する構文についての情報は、『Oracle Essbase データベース管理者ガイド』の重複するメンバー・アウトラインの作成および使用に関する項を参照してください。
phMember	ESS_PHMEMBER_T	メンバー・ハンドルを戻す変数。メンバーが検出されない場合は ESS_NULL になります。

備考

- ターゲット・メンバーに共有メンバーがある場合、実メンバーへのハンドルのみが戻されます。
- 実メンバーへのメンバーのハンドルを持つと、**EssOtlGetNextSharedMember()**を使用して共有メンバー情報を取得できます。
- メンバーが見つからなかった場合は、*phMember が ESS_NULL に設定され、呼出しから 0 が戻されます。
- **EssOtlFindMember()**を使用する場合は、必ず次の 2 つを確認します:
  1. 戻されるステータス番号を確認します。
  2. ハンドルが戻されたかどうかを確認します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hDimProduct;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
  sts = EssOtlFindMember(hOutline, "Product",
    &hDimProduct);
}
```

関連トピック

- EssOtlMoveMember
- EssOtlRenameMember
- EssOtlAddMember

- EssOtlDeleteMember
- EssOtlGetNextSharedMember

EssOtlFindObject

指定されたタイプと名前のオブジェクト・ハンドルを戻します。

構文

```
ESS_FUNC_M EssOtlFindObject(
hOutline, objType, objName, pObjHandle
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objType	ESS_OBJECT_TYPES	次のいずれかの値を持つオブジェクト・タイプ: ● OBJECT_SMARTLIST オブジェクト・タイプはテキスト・リスト(スマートリスト)
objName	ESS_STR_T	オブジェクトを特定する文字列
phObjHandle	ESS_PHOBJECT_T	見つかったオブジェクト・ハンドルを戻します。

戻り値

戻り値:

- 0 - 正常終了の場合

  phObjHandle はオブジェクト・ハンドルを保持します。
- エラー番号 - 失敗した場合

  phObjHandle は NULL です。

例

```
    void TestCreateObject()
{
  ESS_STS_T       sts = ESS_STS_NOERR;
  ESS_HOUTLINE_T     hOutline = ESS_NULL;
  ESS_OBJDEF_T     Object;
  ESS_OBJECT_TYPES   objType;
  ESS_STR_T       smartListName;
  ESS_HOBJECT_T         ObjHandle;
  ESS_ULONG_T     Count, i;
  ESS_PHOBJECT_T    ObjHandles;
  ESS_HOBJECT_T    hObjHandle;
  ESS_HSMARTLIST_T   hSmartList;
  ESS_STR_T       objName;

  memset(&Object, '\0', sizeof(Object));
```

```
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  Object.AppName = szAppName;
  Object.DbName = szDbName;
  Object.FileName = szFileName;

  /* Open outline */
  sts = EssOtlOpenOutline(hCtx, &Object,
                                ESS_TRUE, ESS_TRUE, &hOutline);

  /* Create a static SmartList */
  objType = OBJECT_SMARTLIST;
  smartListName = "SList1";
  sts = EssOtlCreateObject(hOutline, objType,
                                smartListName, &ObjHandle);

  /* List all SmartList objects */
  objType = OBJECT_SMARTLIST;
  sts = EssOtlListObjects(hOutline, objType,
                                &Count, &ObjHandles);

  /* Save */
  SaveOutline(hOutline);

  /* Find objects */
  objName = "SList1";
  sts =
    EssOtlFindObject(hOutline, objType, objName,
                                            &hObjHandle);

   /* Delete objects */
  hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
  sts = EssOtlDeleteObject(hOutline, hSmartList);
  SaveOutline(hOutline);

  if(ObjHandles)
    EssFree (hInst, ObjHandles);

  /* Unlock objects */
  sts = EssUnlockObject(hCtx, Object.ObjType,
      Object.AppName, Object.DbName, Object.FileName);

        /* Close outline */
  sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo

- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlFreeMembers

EssOtlQueryMembers()から戻されたメンバー配列を解放します。

構文

```
ESS_FUNC_M
EssOtlFreeMembers
 (
hOutline, ulCount, phMembers
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。これは **EssOtlOpenOutlineQuery()**から戻されている必要があります。
ulCount	ESS_ULONG_T	phMember 配列内の要素の数。
phMembers	ESS_PHMEMBER_T	解放されるメンバーのハンドルの配列。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

EssOtlQueryMembers の例を参照してください。

関連トピック

- EssOtlOpenOutlineQuery
- EssOtlQueryMembers
- EssOtlQueryMembersByName

EssOtlFreeSmartListInfo

EssOtlGetSmartListInfo によって取得されたテキスト・リスト(SmartList)オブジェクトを解放します。

構文

```
ESS_FUNC_M EssOtlFreeSmartListInfo(hOutline, pSmartListInfo);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	テキスト・リスト(SmartList)のソース Essbase アウトライン。
pSmartListInfo	ESS_PSMARTLISTINFO_T	テキスト・リスト(SmartList)情報。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    DisplaySmartListInfo(ESS_HOUTLINE_T hOutline, ESS_PHOBJECT_T ObjHandles)
{
    ESS_STS_T                          sts = ESS_STS_NOERR;
    ESS_PSMARTLISTINFO_T    SmartListInfo;
    ESS_ULONG_T                       i;

    sts = EssOtlGetSmartListInfo(hOutline, ObjHandles,
                                &SmartListInfo);
    if(!sts)
    {
        printf("\n");
        printf("\tName: %s\n", SmartListInfo->szName);
        printf("\tMissing Name: %s\n",
            SmartListInfo->szMissingName);
        printf("\tOut of Range Name: %s\n",
            SmartListInfo->szOutOfRangeName);
        printf("\tusLen: %d\n", SmartListInfo->usLen);
        for (i = 0; i < SmartListInfo->usLen; i++)
        {
            printf("\tpIDs: %d, \tpszText[%d]: %s\n",
                        SmartListInfo->pIDs[i], i,
                        SmartListInfo->ppszText[i]);
        }
        printf("\n");
    }
    else
        printf("\t\tEssOtlGetSmartListInfo  sts: %d\n",sts);

    if(SmartListInfo)
        sts =

    EssOtlFreeSmartListInfo(hOutline, SmartListInfo);
```

```
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlFreeObjectArray

オブジェクト・ハンドルの配列の割当てを解除します。

構文

```
ESS_FUNC_M EssOtlFreeObjectArray(
hOutline, count, objHandles
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(クエリー・モードのみ)
count	ESS_ULONG_T	オブジェクト・ハンドルのカウント
objHandles	ESS_PHOBJECT_T	割り当てるオブジェクト・ハンドルの配列

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestFreeObjectArray()
{
    ESS_STS_T                       sts = ESS_STS_NOERR;
```

```
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T                Object;
    ESS_STR_T                    objNames[1];
    ESS_OBJECT_TYPES        objType;
    ESS_ULONG_T                count;
    ESS_PHOBJECT_T            hObjHandles = ESS_NULL;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =             hCtx;
    Object.ObjType =         ESS_OBJTYPE_OUTLINE;
    Object.AppName =         szAppName;
    Object.DbName =         szDbName;
    Object.FileName =     szFileName;

    /* Set up */
    sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
     count = 2;
    objType = OBJECT_SMARTLIST;
    objNames[0] = "Smartlist1";
    objNames[1] = "Smartlist2";

    /* Query objects */
    sts = EssOtlQueryObjects(hOutline, objType,
                    objNames, &Count, &hObjHandles);

    /* Free object array */
    if(hObjHandles)
    {
        sts =
    EssOtlFreeObjectArray(hOutline, count,
                        hObjHandles);

        }

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects

- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlFreeStructure

EssOtlGetAttributeInfo())および **EssOtlGetMemberInfo()**により文字列型の属性情報用に動的に割り当てられたメモリーを解放します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
structId;	ESS_ULONG_T	構造体に対する次の定数識別子のいずれかになります: ● ESS_DT_STRUCT_ATTRIBUTEINFO ● ESS_DT_STRUCT_ATTRSPECS ● ESS_DT_STRUCT_MBRINFO ● ESS_DT_STRUCT_TIGENINFO
count;	ESS_ULONG_T	構造体の数
structPtr;	ESS_PVOID_T	メモリーへのポインタ

備考

EssOtlGetMemberInfo()を呼び出した後に、必ず関数 **EssOtlFreeStructure()**を呼び出してください。

例

```
    void ESS_OtlGetAssociatedAttributes()
{
ESS_STS_T    sts = ESS_STS_NOERR;
ESS_SHORT_T   index;
ESS_USHORT_T   count;
ESS_OBJDEF_T   Object;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T   szDbName;
ESS_OBJNAME_T   szFileName;
ESS_HOUTLINE_T  hOutline;
ESS_PPHMEMBER_T  hMember;
ESS_PPHMEMBER_T  phMember;
ESS_PPMBRINFO_T  phMemberInfo;
ESS_MBRNAME_T   mbrName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
```

```
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
printf("EssOtlOpenOutline() sts: %ld\n",sts);

sts = EssOtlFindMember(hOutline, "100-10", &hMember);
printf("EssOtlFindMember() sts: %ld\n",sts);

sts = EssOtlGetAssociatedAttributes(hOutline, hMember, &count, &phMember);
printf("EssOtlGetAssociatedAttributes() sts: %ld\n",sts);

/* Allocate memory for an array of memberinfo structs */
sts = EssAlloc(hInst,count * (sizeof(ESS_MBRINFO_T)), (ESS_PPVOID_T)&phMemberInfo);
if (!sts)
{
 for(index = 0; index < count; index++)
 {
  /* Step through array of member handles, and assign member */
  sts = EssOtlGetMemberInfo(hOutline,phMember[index],&phMemberInfo[index]);
  printf("EssOtlGetMemberInfo() sts: %ld\n",sts);
  strcpy(mbrName,phMemberInfo[index]->szMember);
  printf("Associated attribute member name #%d is: %s\n",(index + 1),mbrName);
 }
 EssFree(hInst, phMember);
 EssOtlFreeStructure(hOutline, ESS_DT_STRUCT_MBRINFO, 1, phMemberInfo);
}

printf("\n Attributes associated :%ld\n\n", count);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlGenerateCurrencyOutline

既存のアウトラインを基に通貨アウトラインを生成します。

構文

```
ESS_FUNC_M
EssOtlGenerateCurrencyOutline
 (
hOutline, phCurOutline
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
phCurOutline	ESS_PHOUTLINE_T	通貨アウトラインの戻り値のための、アウトラインのコンテキスト・ハンドルを指すポインタ。

備考

- ソースのアウトラインには、時間、会計および国の次元が含まれている必要があります。
- 時間次元およびすべての子孫は、ソースのアウトラインから新しいアウトラインの時間次元へ直接コピーされます。
- CurCategory(Dense, Category = Accounts)という名前の次元が、新規アウトラインに作成されます。ソース・アカウント次元内のすべての通貨カテゴリが、新規アウトラインの CurCategory 次元の子になります。
- CurName(Dense, Category = Country)という名前の次元が、新規アウトラインに作成されます。ソースの国次元のすべての通貨名が、新規アウトラインの CurName 次元の子になります。
- CurType(Sparse, Category = Type)という名前の次元が、新規アウトラインに子なしで作成されます。
- **EssOtlWriteOutline()**の後に **EssOtlRestructure()**を呼び出して通貨アウトラインを保存し、**EssOtlCloseOutline()**を呼び出して通貨アウトラインを終了する必要があります。
- 新規アウトラインには、次の属性があります:
  - 自動構成は、ESS_TRUE に設定されます
  - 大文字と小文字の区別は、元のアウトラインと同じです

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_ALREADYCURRENCY
- OTLAPI_CUR_NOACCOUNTS
- OTLAPI_CUR_NOTIME
- OTLAPI_CUR_NOCOUNTRY

例

```
#include <essapi.h>
```

```
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HOUTLINE_T   hCurOutline;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Interntl");
strcpy(szFileName, "Interntl");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlGenerateCurrencyOutline(hOutline,
    &hCurOutline);
}
```

関連トピック

- EssOtlOpenOutline
- EssOtlWriteOutline
- EssOtlRestructure

EssOtlGetAggLevelUsage

保管された階層に適用されたビュー選択プロパティを戻します。

構文

```
ESS_FUNC_M EssOtlGetAggLevelUsage (
hOutline, hMember, pAgglevelUsage
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	メンバーのハンドル(入力)。
pAgglevelUsage	ESS_PSHORT_T	EssOtlSetAggLevelUsage マニュアル(出力)にリストされたレベル使用定数の 1 つ。

備考

この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。

戻り値

正常終了の場合は、0 が戻されます。

例

```
     ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline = ESS_NULL;
 ESS_HMEMBER_T   hMember = ESS_NULL;
 ESS_SHORT_T    sAggLevelUsage = 0;

 /* code to assign hOutline variable omitted */
 /* code to assign hMember variable omitted */

 if (hOutline && hMember)
 {
   sts = EssOtlGetAggLevelUsage (hOutline, hMember, &sAggLevelUsage);
if (sts)
 printf("Error (%ld) getting AggLevelUsage\n", sts);
else
    printf("AggLevelUsage is: %d ", sAggLevelUsage);
    switch (sAggLevelUsage)
    {
    case ESS_AGGLEVELUSAGE_NOTSET :
        printf("(not set)\n");
        break;
    case ESS_AGGLEVELUSAGE_DEFAULT :
        printf("(Default)\n");
        break;
    case ESS_AGGLEVELUSAGE_ALL :
        printf("(All levels considered)\n");
        break;
    case ESS_AGGLEVELUSAGE_NOAGGREGATION :
        printf("(Do not aggregate)\n");
        break;
    case ESS_AGGLEVELUSAGE_BOTTOMONLY :
        printf("(Bottom level only considered)\n");
        break;
    case ESS_AGGLEVELUSAGE_TOPONLY :
        printf("(Top level only considered)\n");
        break;
    case ESS_AGGLEVELUSAGE_BOTTOMTOP :
        printf("(Never aggregate intermediate levels)\n");
        break;
    case ESS_MULTIPLE_HIERARCHY_IS_ENABLED :
printf("(Error: Multiple hierarchies - hierarchy members are gen=2)\n");
        break;
    case ESS_MULTIPLE_HIERARCHY_NOT_ENABLED :
   printf("(Error: Single hierarchy - hierarchy member is gen=1)\n");
        break;
    case ESS_NOT_HIERARCHY_MEMBER :
  printf("(Error: This member does not carry agglevel information)\n");
```

```
        break;
    default: printf("(Unrecognized response)\n");
   }
 }
 else
 {
   if (!hOutline)
    printf("Outline not provided\n");
   if (!hMember)
    printf("Member not provided\n");
 }
```

関連トピック

- EssOtlSetAggLevelUsage

EssOtlGetAliasTableLanguages

指定した別名テーブルに関連付けられている言語コードの配列とその配列内の言語コードの数が戻されます。

構文

```
ESS_FUNC_M
EssOtlGetAliasTableLanguages
 (
hOutline
,
pszAliasTable
,
pulCount
,
ppLangArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのハンドル。
pszAliasTable	ESS_STR_T	関連付けられている言語コードを取得する別名テーブル名。
pulCount	ESS_PULONG_T	別名テーブルに関連付けられた言語コードの数が戻される変数のアドレス。
ppLangArray	ESS_PPALIASLANG_T	pszAliasTable で指定された別名テーブルに関連付けられている言語コードの配列。 ppLangArray に対して割り当てられているメモリーは、EssFree()を使用して解放する必要があります。

戻り値

- 成功の場合、0 が戻されます。
- 処理に失敗すると、エラー OTLAPI_BAD_ALIASTABLE (無効な別名テーブル) が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OUTLINEINFO_T  NewInfo;
ESS_HOUTLINE_T   hOutline;
ESS_PALIASLANG_T  pLangs=ESS_NULL;
ESS_ULONG_T    nLangs = 0, i=0;

memset(&NewInfo, '\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo, &hOutline);

if (!sts)
{
 sts = EssOtlCreateAliasTable(hOutline,
 "French Alias Table");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr-CA");
}

if (!sts)
{
    sts = EssOtlGetAliasTableLanguages(hOutline, "French Alias Table", &nLangs,
&pLangs);

 if ( !sts == ESS_STS_NOERR && ( pLangs) )
 {
   for (i=0;i<nLangs ;++i)
   {
     if (pLangs[i])
     {
       printf("Language Code:  %s\n", pLangs[i]);
     }
   }
   EssFree(hInst, pLangs);
 }
}
if (!sts)
{
 sts = EssOtlClearAliasTableLanguages (hOutline,
 "French Alias Table");
```

```
}
```

関連トピック

- EssOtlClearAliasTableLanguages
- EssOtlSetAliasTableLanguage

EssOtlGetAltHierarchyEnabled

次元の複合階層使用可能設定を戻します。

構文

```
ESS_FUNC_M EssOtlGetAltHierarchyEnabled(
hOutline
,
hDimMember
,
pEnabled
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDimMember	ESS_HMEMBER_T	次元メンバー(入力)。
pEnabled	ESS_BOOL_T	次元が複合階層使用可能に設定されている場合は TRUE を戻し、それ以外の場合は FALSE を戻します。

戻り値

- 0 - 正常終了の場合
- hDimMember が次元メンバーでない場合、エラー OTLAPI_ERR_BADDIM を戻します。

関連トピック

- EssOtlSetAltHierarchyEnabled
- EssOtlGetHierarchyType
- EssOtlSetHierarchyType

EssOtlGetASOCompressionDimension

圧縮のタグが付けられた集約ストレージ次元のハンドルを戻します。

構文

```
ESS_FUNC_M
EssOtlGetASOCompressionDimension
 (
hOutline, phDim
```

```
    );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
phDim	ESS_PHMEMBER_T	次元ハンドル(出力)へのポインタ。

備考

デフォルトでは、集約ストレージ・データベースの圧縮次元は会計次元です。圧縮次元を変更するには、EssOtlSetASOCompressionDimension を使用します。圧縮次元を変更すると、データベース全体の再構築がトリガーされます。

戻り値

正常終了の場合は、0 が戻されます。

例

```
     ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T   hOutline = ESS_NULL;
 ESS_PMBRINFO_T   pMemberInfo = ESS_NULL;
 ESS_HMEMBER_T    hMember = ESS_NULL;

 /* code to assign hOutline variable omitted */

 if (hOutline)
 {
  sts = EssOtlGetASOCompressionDimension(hOutline, &hMember);
  if (!sts)
  {
   if (hMember)
   {
    sts = EssOtlGetMemberInfo(hOutline, hMember, &pMemberInfo);
    printf("\The ASO compression dimension is: %s\n", pMemberInfo->szMember);
   }
   else
   {
    printf("Outline has no dimension selected for compression\n");
   }
  }
  else
  {
   printf("Error returned\n");
  }
 }
 else
 {
   printf("NULL outline selected");
 }
```

関連トピック

- EssOtlSetASOCompressionDimension

EssOtlGetAssociatedAttributes

基本メンバーまたは基本次元に関連付けられているすべての属性メンバーを戻します。

構文

```
ESS_FUNC_M EssOtlGetAssociatedAttributes(hOutline,hMember,
pusCount,pphMemberArray);
```

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
hMember;	ESS_HMEMBER_T	基本メンバーまたは基本次元のハンドル
pusCount;	ESS_PUSHORT_T	戻された属性メンバー数
pphMemberArray;	ESS_PPHMEMBER_T	属性メンバーのハンドルの配列へのポインタ

例

```
    void  ESS_OtlGetAssociatedAttributes()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_SHORT_T    index;
 ESS_USHORT_T   count;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_HOUTLINE_T  hOutline;
 ESS_PPHMEMBER_T  hMember;
 ESS_PPHMEMBER_T  phMember;
 ESS_PPMBRINFO_T  phMemberInfo;
 ESS_MBRNAME_T   mbrName;

 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline() sts: %ld\n",sts);

 sts = EssOtlFindMember(hOutline, "100-10", &hMember);
 printf("EssOtlFindMember() sts: %ld\n",sts);

 sts = EssOtlGetAssociatedAttributes(hOutline, hMember, &count, &phMember);
```

```
 printf("EssOtlGetAssociatedAttributes() sts: %ld\n",sts);

 /* Allocate memory for an array of memberinfo struct handles */
 sts = EssAlloc(hInst,count * (sizeof(ESS_HMEMBER_T)), (ESS_PPVOID_T)&phMemberInfo);
 if (!sts)
 {
  for(index = 0; index < count; index++)
  {
   /* Step through array of member handles, and assign member */
   sts = EssOtlGetMemberInfo(hOutline,phMember[index],&phMemberInfo[index]);
   printf("EssOtlGetMemberInfo() sts: %ld\n",sts);
   strcpy(mbrName,phMemberInfo[index]->szMember);
   printf("Associated attribute member name #%d is: %s\n",(index + 1),mbrName);
  }
  EssFree(hInst, phMember);
  EssFree(hInst, phMemberInfo);
 }

 printf("\n Attributes associated :%ld\n\n", count);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlGetAttributeAssocLevel

属性またはリンク属性次元の関連付けレベルを取得します。

属性それぞれには関連付けレベルと属性次元定義に関連付けられた添付レベルがあります。リンク属性次元について、関連付けレベルは常にリンク属性次元が示す期別比較の 2 つの期間のうち、短いほうです。たとえば、リンク属性次元 Quarter by Year では Quarter は関連付けレベルで、Year は添付レベルです。

構文

```
ESS_FUNC_M
EssOtlGetAttributeAssocLevel
```

```
(
hOutline, hDimMember, psLevel
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDimMember	ESS_HMEMBER_T	属性またはリンク属性次元メンバー・ハンドル(入力)。
psLevel	ESS_PUSHORT_T	属性の関連付けレベル(出力)。

備考

- この関数を呼び出す前に、EssOtlOpenOutline または EssOtlOpenOutlineQuery のいずれかを使用して、アウトラインを編集モードまたはクエリー・モードで開きます。
- この関数は hDimMember のタイプが属性次元の場合に適用できます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER    hDimMember;
ESS_USHORT_T    usAssocLevel;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Quarter By Year",
    &hDimMember);
}
```

```
if (!sts  && hMemberJan)
{
 sts = EssOtlGetAttributeAssocLevel(hOutline,
    hDimMember, &usAssocLevel);
```

関連トピック

- EssOtlGetLinkedAttributeAttachLevel
- EssOtlQueryGenerationInfo

EssOtlGetAttributeInfo

指定した属性メンバーまたは属性次元に関する属性情報を戻します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
hAttribute;	ESS_HMEMBER_T	属性メンバーまたは属性次元のハンドル
pAttributeInfo;	124 ページの「ESS_ATTRIBUTEINFO_T」	属性情報

備考

- この関数は **EssGetAttributeInfo()** と同様の関数です。
- この関数を呼び出した後、**EssOtlFreeStructure()** を呼び出して、**EssOtlGetAttributeInfo()** によって文字列タイプの属性情報用に動的に割り当てられたメモリーを解放してください。

例

```
    void ESS_GetAttributeInfo()
{
 ESS_PPATTRIBUTEINFO_T  pAttributeInfo = ESS_NULL;
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_OBJDEF_T   Object;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_HOUTLINE_T   hOutline;
 ESS_PPHMEMBER_T   phMember;
 ESS_PPMBRINFO_T   phMemberInfo;
 ESS_MBRNAME_T   mbrName;
 ESS_HMEMBER_T   hMember;

 memset(&Object, '\0', sizeof(Object));
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  strcpy(szAppName, "Sample");
  strcpy(szDbName, "Basic");
```

```
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
printf("EssOtlOpenOutline() sts: %ld\n",sts);

sts = EssOtlFindMember(hOutline, "100-10", &hmember);
printf("EssOtlFindMember() sts: %ld\n",sts);

sts = EssOtlGetAttributeInfo(hOutline, hMember, &pAttributeInfo);
if (sts == ESS_STS_NOERR && pAttributeInfo)
{
 printf("\n------Attribute Information------\n");
 printf("Member name:        %s\n", pAttributeInfo->MbrName);
 printf("Dim name:         %s\n", pAttributeInfo->DimName);

  switch(pAttributeInfo->Attribute.usDataType)
  {
   case (ESS_ATTRMBRDT_STRING):
    printf("Attribute data type:    Text\n");
    if(pAttributeInfo->Attribute.value.strData)
    printf("Attribute value:      %s\n",pAttributeInfo->Attribute.value.strData);
    break;

   case (ESS_ATTRMBRDT_BOOL):
    printf("Attribute data type:    Boolean\n");
    printf("Attribute value:      %d\n",pAttributeInfo->Attribute.value.bData);
    break;

   case (ESS_ATTRMBRDT_DOUBLE):
    printf("Attribute data type:    Numeric\n");
    printf("Attribute value:      %f\n",pAttributeInfo->Attribute.value.dblData);
    break;

   case (ESS_ATTRMBRDT_DATETIME):
    printf("Attribute data type:    Date\n");
    printf("Attribute value:      %s\n",ctime(&pAttributeInfo-
>Attribute.value.dtData));
    break;

   case (ESS_ATTRMBRDT_NONE):
    printf("Attribute data type:    None\n");
    break;

   default:
    printf("Attribute data type:    \n");
    break;
  }
 }
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure

- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlGetAttributeSpecifications

アウトラインの属性指定を取得します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
pAttrSpecs;	125 ページの「ESS_ATTRSPECS_T」	属性指定

備考

- この関数は、開かれたアウトラインから情報を戻すということを除けば、**EssGetAttributeSpecifications()**と同様の関数です。
- **EssOtlSetAttributeSpecifications()**を使用して、アウトラインの属性指定を設定します。
- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供

例

```
    void ESS_OtlGetAttributeSpecifications()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PATTRSPECS_T AttrSpecs;
  ESS_OBJDEF_T   Object;
  ESS_HOUTLINE_T  hOutline;
  ESS_APPNAME_T   szAppName;
  ESS_DBNAME_T   szDbName;
```

```
 ESS_OBJNAME_T  szFileName;
 ESS_PROCSTATE_T pState;

memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
printf("EssOtlOpenOutline() sts: %ld\n",sts);

sts = EssOtlGetAttributeSpecifications(hOutline,&AttrSpecs);
printf("EssOtlGetAttributeSpecifications() sts: %ld\n",sts);

switch(AttrSpecs->usGenNameBy)
{
 case ESS_GENNAMEBY_PREFIX:
  printf("\n Prefix/Suffix  : Prefix");
  break;
 case ESS_GENNAMEBY_SUFFIX:
  printf("\n Prefix/Suffix  : Suffix");
  break;
 default:
  printf("\n Prefix/Suffix  : None");
  break;
}
switch(AttrSpecs->usUseNameOf)
{
 case ESS_USENAMEOF_PARENT:
  printf("\n Use Name of   : Parent");
  break;
 case ESS_USENAMEOF_GRANDPARENTANDPARENT:
  printf("\n Use Name of   : Grand Parent and Parent");
  break;
 case ESS_USENAMEOF_ALLANCESTORS:
  printf("\n Use Name of   : All Ancestors");
  break;
 case ESS_USENAMEOF_DIMENSION:
  printf("\n Use Name of   : Dimension");
  break;
 case ESS_USENAMEOF_NONE:
  printf("\n Use Name of   : None");
  break;
 default:
  printf("\n Use Name of   : Invalid setting");
  break;
}
switch(AttrSpecs->cDelimiter)
{
 case ESS_DELIMITER_PIPE:
  printf("\n Delimiter    : '|'");
  break;
```

```
 case ESS_DELIMITER_UNDERSCORE:
  printf("\n Delimiter    : '_'");
  break;
 case ESS_DELIMITER_CARET:
  printf("\n Delimiter    : '^'");
  break;
 default:
  printf("\n Delimiter    : Invalid setting");
  break;
}

switch(AttrSpecs->usDateFormat)
{
 case ESS_DATEFORMAT_DDMMYYYY :
  printf("\n Date Format   : DD-MM-YYYY");
  break;
 case ESS_DATEFORMAT_MMDDYYYY :
  printf("\n Date Format   : MM-DD-YYYY");
  break;
 default:
  printf("\n Date Format   : Invalid setting");
  break;

}
switch(AttrSpecs->usBucketingType)
{
 case ESS_UPPERBOUNDINCLUSIVE :
  printf("\n Bucketing Type  : Upper Bound inclusive");
  break;
 case ESS_UPPERBOUNDNONINCLUSIVE :
  printf("\n Bucketing Type  : Upper Bound non-inclusive");
  break;
 case ESS_LOWERBOUNDINCLUSIVE :
  printf("\n Bucketing Type  : Lower Bound inclusive");
  break;
 case ESS_LOWERBOUNDNONINCLUSIVE :
  printf("\n Bucketing Type  : Lower Bound non-inclusive");
  break;
 default:
  printf("\n Bucketing Type  : Invalid setting");
  break;
}

printf("\n Default for TRUE    : %s",
    AttrSpecs->pszDefaultTrueString);

printf("\n Default for FALSE   : %s",
    AttrSpecs->pszDefaultFalseString);

printf("\n Default for Attr Calc  : %s",
    AttrSpecs->pszDefaultAttrCalcDimName);

printf("\n Default for Sum    : %s",
    AttrSpecs->pszDefaultSumMbrName);

printf("\n Default for Count   : %s",
    AttrSpecs->pszDefaultCountMbrName);
```

```
 printf("\n Default for Average   : %s",
     AttrSpecs->pszDefaultAverageMbrName);

 printf("\n Default for Min     : %s",
     AttrSpecs->pszDefaultMinMbrName);

 printf("\n Default for Max     : %s",
     AttrSpecs->pszDefaultMaxMbrName);

 printf("\n");

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
 sts = EssOtlCloseOutline(hOutline);
 printf("EssOtlCloseOutline() sts: %ld\n",sts);

 EssOtlFreeStructure(hInst, ESS_DT_STRUCT_ATTRSPECS, 1,&AttrSpecs);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlQueryAttributes
- EssOtlSetAttributeSpecifications

EssOtlGetChild

メンバーの子を戻します。

構文

```
ESS_FUNC_M
EssOtlGetChild
 (
hOutline, hMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	子を取得するメンバーのハンドル。
phMember	ESS_PHMEMBER_T	パラメータ hMember の子のメンバーのハンドルの戻り値を指すポインタ。

備考

- 子が存在しない場合は、*phMember が ESS_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberParent;
ESS_HMEMBER_T    hMemberChild;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Year",
    &hMemberParent);
}
if (!sts && hMemberParent)
```

```
{
 sts = EssOtlGetChild(hOutline, hMemberParent,
    &hMemberChild);
}
```

関連トピック

- EssOtlGetParent
- EssOtlGetNextSibling
- EssOtlGetPrevSibling
- EssOtlGetFirstMember

EssOtlGetCountOfDupMemberNameInDim

クエリー・モードで開かれたアウトラインの次元内で名前が重複しているメンバーの数を戻します。

構文

```
ESS_FUNC_M EssOtlGetDimensionNameUniqueness (
hOutline, hDim, *pulDupCount
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDim	ESS_HMEMBER_T	入力の次元。これは、EssOtlQueryGetFirstDimension()またはEssOtlQueryGetNextDimension()によって戻されます。
*pulDupCount	ESS_ULONG_T	名前が重複しているメンバーの数(出力)。

備考

- 次元内の共有メンバーは、カウントされません。
- この関数を呼び出す前に、EssOtlOpenOutlineQuery を呼び出し、クエリー・モードでアウトラインを開いてください。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_GetCount()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hDim;
ESS_LONG_T   pulDupCount;
```

```
memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;

sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);


if (!sts)
{
 sts = EssOtlQueryGetFirstDimension(hOutline, &hDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
}

if (!sts)
{
 // returns pulDupCount which gives the number of members in a dimension
 // whose names are duplicate
  sts = EssOtlGetCountOfDupMemberNameInDim (hOutline, hDim, &pulDupCount);

 if (sts)
  printf("EssOtlGetCountOfDupMemberNameInDim failed sts %ld\n",sts);
}

return sts;
}
```

関連トピック

- EssOtlQueryGetFirstDimension
- EssOtlQueryGetNextDimension

EssOtlGetDateFormatString

この関数は、アウトライン・プロパティ日付フォーマット文字列を取得します。

構文

```
  ESS_FUNC_M EssOtlGetDateFormatString(
ESS_HOUTLINE_T hOutline,
ESS_PSTR_T  formatString)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル
formatString	ESS_PSTR_T	この引数にアウトライン日付フォーマット文字列を戻します。

戻り値

戻り値:

- 0 - 正常終了の場合

  formatString にはアウトライン日付フォーマットが含まれます。

- エラー番号 - 失敗した場合

例

```
    void TestGetSetDateFormatString()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T                Object;
    ESS_SHORT_T                length = 80;
    ESS_STR_T                    dateFormatString = "";
    ESS_STR_T                    localeStr;
    ESS_USHORT_T                count, i;
    ESS_STR_T*                pdateStrings;
    ESS_STR_T*                pformatStrings;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =          hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =         szDbName;
    Object.FileName =     szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Get current value */
    sts =
    EssOtlGetDateFormatString(hOutline, &dateFormatString);

    printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
    printf("\tDate format string: %s\n", dateFormatString);

    printf("\n");
    localeStr = "English_UnitedStates.Latin1@Binary";
    sts = EssOtlGetServerDateFormats(hCtx, localeStr,
                &Count, &pdateStrings, &pformatStrings);
    printf("EssOtlGetServerDateFormats sts: %d \n", sts);

    for (i = 0; i < count; i++)
    {
        printf("\nCase with %s:\n", pformatStrings[i]);
        sts = EssOtlSetDateFormatString(hOutline,
                    pformatStrings[i]);
```

```
        printf("EssOtlSetSMDateFormatString sts: %d \n", sts);
        SaveOutline(hOutline);

        sts =
    EssOtlGetDateFormatString(hOutline,
                    &dateFormatString);

        printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
        printf("\tDate format string: %s\n", dateFormatString);

    }

    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline  sts: %d\n",sts);
}
```

関連トピック

- EssOtlGetServerDateFormats
- EssOtlSetDateFormatString

EssOtlGetDimensionNameUniqueness

その次元メンバー名の一意性の設定を戻します。

構文

```
ESS_FUNC_M EssOtlGetDimensionNameUniqueness (
hOutline, hDim, pbNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDim	ESS_HMEMBER_T	次元ルート・メンバーのメンバー・ハンドル(入力)。
pbNameUnique	ESS_PBOOL_T	次元メンバー名の一意の設定(出力)。TRUE の場合、次元に重複するメンバー名を持たせることはできません。

備考

EssOtlFindMember を呼び出して、ESS_HMEMBER_T (hDim)変数を設定してください。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_GetSetDimNameUniq()
{
```

```
ESS_STS_T  sts = 0;
ESS_POUTLINEINFO_T pInfo = ESS_NULL;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_BOOL_T  pbNameUnique;
ESS_HMEMBER_T  hDim = ESS_NULL;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Year",&hDim);

 if (sts)
  printf("EssOtlFindMember failed sts %ld\n",sts);
}

/*Get the dimension's, Year, member-name uniqueness setting */
if (!sts)
{
 sts = EssOtlGetDimensionNameUniqueness (hOutline, hDim, &pbNameUnique);

 if (sts)
  printf("EssOtlGetDimensionNameUniqueness failed sts %ld\n",sts);
 else
  printf("Dimension Year has Member Name Uniqueness value: %ld\n", pbNameUnique);
}

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Product",&hDim);

 if (sts)
  printf("EssOtlFindMember failed sts %ld\n",sts);
}

if (!sts)
{
 /*set Product to prohibit duplicate (non-unique) member names*/
 pbNameUnique = ESS_TRUE;
 sts = EssOtlSetDimensionNameUniqueness (hOutline, hDim, pbNameUnique);
```

```
 if (sts)
  printf("EssOtlSetDimensionNameUniqueness failed sts %ld\n",sts);
 else
  printf("Dimension Product has Member Name Uniqueness value: %ld\n", pbNameUnique);
}

return sts;

}
```

関連トピック

- EssOtlSetDimensionNameUniqueness

EssOtlGetDimensionSolveOrder

次元の解決順を戻します。

構文

```
ESS_FUNC_M EssOtlGetDimensionSolveOrder (
hOutline, hMember, pOrder
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_PHMEMBER_T	次元ハンドル(入力)。
pOrder	ESS_PUCHAR_T	解決順(出力)。

備考

解決順は集約ストレージ・データベースにのみ適用できます。

戻り値

成功の場合、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T  hOutline = ESS_NULL;
 ESS_HMEMBER_T   hMember = ESS_NULL;
 ESS_UCHAR_T    ucOrder = 0;

 /* code to assign hOutline variable omitted */
 /* code to assign hMember variable omitted */


 if (hOutline && hMember)
 {
  sts = EssOtlGetDimensionSolveOrder(hOutline, hMember, &ucOrder);
```

```
  if (sts)
   printf("Error [%ld] returned\n", sts);
  else
   printf("Solve Order: %d\n", ucOrder);
 }
 else
  printf("Both hOutline and hMember must have values\n");
```

関連トピック

- EssOtlSetDimensionSolveOrder
- EssOtlGetMemberSolveOrder
- EssOtlSetMemberSolveOrder

EssOtlGetDimensionUserAttributes

指定された次元で使用されるユーザー定義属性を戻します。

構文

```
ESS_FUNC_M
EssOtlGetDimensionUserAttributes
 (
hOutline
,
pPredicate
,

pCounts
,
ppAttributeNames
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。これは **EssOtlOpenOutlineQuery()** から戻されている必要があります。
pPredicate	778 ページの「ESS_PREDICATE_T」	クエリーを定義している構造体。この構造体のフィールドの用法: • **ulQuery** - 実行する操作を定義する値。有効な唯一の値は ESS_DIMUSERATTRIBUTES です。 • **pszDimension**- クエリー範囲を制限する次元。有効な次元名を指定します。

パラメータ	データ型	説明
pCounts	768 ページの「ESS_MBRCOUNTS_T」	カウントに関する情報を定義している構造体。次のフィールドが含まれます: ● ulStart- 戻す最初の番号 ● ulMaxCount- 戻されるメンバー名の最大数 ● ulTotalCount- クエリーの実行結果において定義されるメンバーの合計数 ● pulReturnCount- このクエリーで戻されたメンバー名の数
ppAttributeNames	ESS_PPMBRNAME_T	このクエリーから戻される属性名の配列。

備考

- この関数は、特定の次元でユーザーが定義した属性を取得するために使用します。したがって、Predicate に有効な唯一の値は ESS_DIMUSERATTRIBUTES_T です。
- メンバーまたは次元の解決順プロパティは、計算順序を指定します。
- メンバーの解決順は次元の解決順よりも優先されます。解決順は 0 から 127 までにできます。デフォルトは 0 です。
- 解決順が指定されていない式を持たないメンバーは、その次元の解決順を継承します。解決順が指定されていない式を持つメンバーは、ゼロの解決順を持ちます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_PREDICATE_T   Predicate;
ESS_MBRCOUNTS_T   Counts;
ESS_MBRNAME_T    pAttribNames;
ESS_ULONG_T     i;
ESS_ACCESS_T    Access;
ESS_STR_T     AppName;
ESS_STR_T     DbName;

AppName = "Sample";
DbName = "Basic";

sts = EssSetActive(hCtx, AppName, DbName, &Access);

if ( sts == 0)
{
 memset(&Object, '\0', sizeof(Object));
```

```
 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);

 memset(&Predicate, '\0', sizeof(Predicate));
 Predicate.ulQuery    = ESS_DIMUSERATTRIBUTES;
 Predicate.pszDimension  = "Market";

 memset(&Counts, '\0', sizeof(Counts));
 Counts.ulStart   = 0;
 Counts.ulMaxCount = 10;

 if(!sts)
 {
  sts = EssOtlGetDimensionUserAttributes(hOutline,
      &Predicate, &Counts, &pAttribNames);

  if (!sts && Counts.ulReturnCount)
  {
   sts = EssFree(hInstance, pAttribNames);
  }
 }
}
```

関連トピック

- EssFree
- EssOtlOpenOutlineQuery
- EssOtlQueryMembers
- EssOtlQueryMembersByName

EssOtlGetDTSMemberAlias

動的時系列(DTS)メンバーの別名を取得します。

構文

```
ESS_STS_T
EssOtlGetDTSMemberAlias
 (
hOutline, pszDTSMember, pszAliasTable, ppszAlias
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	EssOtlOpenOutline 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember	ESS_STR_T	別名を提供する DTS メンバー名。
pszAliasTable	ESS_STR_T	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルが使用されます。
ppszAlias	ESS_PSTR_T	DTS メンバーの別名を含む C 文字列を指すポインタへのポインタ。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE

例

```
    #include "essapi.h"
#include "essotl.h"
#include "esserror.h"

ESS_STS_T ESS_OtlGetDTSMemberAlias(ESS_HCTX_T hCtx)
{
    ESS_STS_T       sts =ESS_STS_NOERR;
    ESS_OBJDEF_T  Object;
    ESS_HOUTLINE_T hOutline;
    ESS_APPNAME_T  szAppName;
    ESS_DBNAME_T  szDbName;
    ESS_OBJNAME_T  szFileName;
    ESS_CHAR_T   pszAliasTable[ESS_ALIASNAMELEN];
    ESS_STR_T    pszAlias;
    ESS_CHAR_T   pszDTSMember[ESS_MBRNAMELEN];

    strcpy(szAppName, "sample");
    strcpy(szDbName, "Basic");
    strcpy(szFileName, "Basic");
    strcpy(pszDTSMember, "Q-T-D");
    strcpy(pszAliasTable, "Default");

    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object, ESS_FALSE, ESS_TRUE, &hOutline);

    if(sts)
    {
        printf("Could not open outline\n");
        return sts;
    }

    sts = EssOtlGetDTSMemberAlias(hOutline, pszDTSMember, pszAliasTable, &pszAlias);
    if(sts)
    {
        printf("Could not get DTS member alias\n");
        return sts;
    }
    printf("MEMBER %s is aliased to %s\n", pszDTSMember, pszAlias);
    EssOtlCloseOutline(hOutline);
    return sts;
```

```
}
```

関連トピック

- EssOtlDeleteDTSMemberAlias
- EssOtlEnableDTSMember
- EssOtlGetEnabledDTSMembers
- EssOtlSetDTSMemberAlias

EssOtlGetEnabledDTSMembers

指定されたアウトラインにある使用可能な動的時系列(DTS)メンバーのメンバー情報構造体を取得します。

構文

```
ESS_STS_T
EssOtlGetEnabledDTSMembers
 (
hOutline, pusCount, ppEnabledDTSMemberList
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	EssOtlOpenOutline 呼出しから戻される Essbase アウトライン・ハンドル。
pusCount	ESS_PUSHORT_T	使用可能な DTS メンバーの数。
ppEnabledDTSMemberList	ESS_PPDTSMBRINFO_T	(アウトラインに対して使用可能な DTS メンバーの)DTS メンバー情報構造体の配列へのポインタ。

備考

この関数は、この関数に渡された ESS_DTSMBRNAME_T 構造体にも値を入れます。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、**EssOtlQueryMembers()**呼出しのステータスを戻します。

例

```
    #include "essapi.h"
#include "essotl.h"
#include "esserror.h"

ESS_STS_T ESS_OtlGetEnabledDTSMembers(ESS_HCTX_T hCtx)
{
    ESS_STS_T    sts =ESS_STS_NOERR;
    ESS_HOUTLINE_T  hOutline;
    ESS_OBJDEF_T   Object;
    ESS_APPNAME_T  szAppName;
```

```
    ESS_DBNAME_T   szDbName;
    ESS_OBJNAME_T  szFileName;
    ESS_USHORT_T   usCount, i;
    ESS_PDTSMBRNAME_T    pEnabledDTSMbrList;

    strcpy(szAppName, "Sample");
    strcpy(szDbName, "Basic");
    strcpy(szFileName, "Basic");

    memset(&Object, '\0', sizeof(ESS_OBJDEF_T));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object, ESS_FALSE, ESS_TRUE, &hOutline);
    if(sts)
    {
        printf("Could not open outline\n");
        return sts;
    }

    sts = EssOtlGetEnabledDTSMembers(hOutline, &usCount, &pEnabledDTSMbrList);
    if(sts)
    {
        printf("Could not get enabled DTS member alias\n");
    }
    else
    {
        printf("No of enabled DTS members is %u\n", usCount);
        for (i = 0; i < usCount; i++)
        {
            printf("%s\n", pEnabledDTSMbrList[i]);
        }
    }
    EssOtlCloseOutline(hOutline);
    return sts;
}
```

関連トピック

- EssOtlDeleteDTSMemberAlias
- EssOtlEnableDTSMember
- EssOtlGetDTSMemberAlias
- EssOtlSetDTSMemberAlias

EssOtlGetFirstMember

アウトラインの最初のメンバーのハンドルを戻します。最初のメンバーはアウトラインで最初に定義されている次元です。

構文

```
ESS_FUNC_M
EssOtlGetFirstMember
 (
hOutline, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
phMember	ESS_PHMEMBER_T	アウトラインの最初のメンバーのハンドルの戻りを指すポインタ。このパラメータはアウトラインをたどる後続の呼出しに渡されます。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_MEMBER_T    hMemberFirst
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlGetFirstMember(hOutline,
    &hMemberFirst);
}
```

関連トピック

- EssOtlGetParent
- EssOtlGetNextSibling
- EssOtlGetPrevSibling

- EssOtlGetChild

EssOtlGetGenName

次元内の特定の世代の名前を取得します。

構文

```
ESS_FUNC_M
EssOtlGetGenName
 (
hOutline, pszDimension, usGen, ppszName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象の世代を含む次元の名前。
usGen	ESS_USHORT_T	名前を取得する世代の番号。次元は世代 1 です。
ppszName	ESS_PSTR_T	世代名が戻されるバッファ。API によって割り当てられます。

備考

- 世代名はメンバー名と同じルールに従い、メンバー名全体で一意性が必要です。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- 世代名は自動的には付与されません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EssOtlSetGenName で割り当てられます
- 戻りバッファを解放するには、**EssFree()**を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_NO_GENLEVELNAME
- OTLAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
```

```
ESS_STR_T      Dimension;
ESS_USHORT_T    GenNum;
ESS_STR_T      GenName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/************ Get Gen Name ***************/
Dimension = "Year";
GenNum = 3;

if (!sts)
{
 sts = EssOtlGetGenName(hOutline, Dimension,
    GenNum, &GenName);
}

if (!sts && GenName)
{
 printf("Gen Name: %s\n",GenName);
    EssFree(hInst, GenName);
}
```

関連トピック

- EssFree
- EssOtlDeleteGenName
- EssOtlSetGenName

EssOtlGetGenNameEx

次元内の特定の世代について、名前とメンバーの一意性の設定を取得します。

構文

```
ESS_FUNC_M
EssOtlGetGenName
 (
hOutline, pszDimension, usGen, ppszName, pbNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象の世代を含む次元の名前。
usGen	ESS_USHORT_T	名前を取得する世代の番号。次元は世代 1 です。
ppszName	ESS_PSTR_T	世代名が戻されるバッファ。API によって割り当てられます。
pbNameUnique	ESS_PBOOL_T	メンバー名の一意性の設定。

備考

- 世代名はメンバーの名前スペース全体で一意の必要があります。他の世代、レベル、メンバー名、または別名と重複できません。重複した世代名を追加しようとすると、エラーが発生します。
- 世代名は自動的には付与されません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EssOtlSetGenName で割り当てられます
- 戻りバッファを解放するには、**EssFree()**を呼び出します。

戻り値

正常終了の場合は 0 が戻されます。それ以外はエラー・コードが戻されます。

例

```
    void ESS_GetGenNameEx()

{

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    GenNum;
ESS_STR_T     GenName;
ESS_BOOL_T     bUnique= ESS_FALSE;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline sts: %ld\n",sts);
```

```
/*************** Set and Get GenName **************/
 Dimension = "Year";
 GenNum = 1;
 GenName = "Gen 1 Year";

 //SetGenNameEx() so that Gen 1 members of Year cannot be non-unique
 if (!sts)
 {
  sts = EssOtlSetGenNameEx(hOutline, Dimension,
   GenNum, GenName, ESS_TRUE);
 }



 // GetGenNameEx() to see if the gen is able to be non-unique
 if (!sts)
 {
  sts = EssOtlGetGenNameEx(hOutline, Dimension,
     GenNum, &GenName, &bUnique);
  printf("Generation 1 members of Year have bUnique value of %ld\n", bUnique);
  printf("EssOtlGetGenNameEx sts: %ld\n",sts);
 }

 if (!sts && GenName)
 {
  printf("Gen Name: %s\n",GenName);
     EssFree(hInst, GenName);
 }

}
```

関連トピック

- EssOtlGetGenName
- EssFree
- EssOtlDeleteGenName
- EssOtlSetGenNameEx

EssOtlGetGenNames

特定の次元に対して指定されたすべての世代名を取得します。

構文

```
ESS_FUNC_M
EssOtlGetGenNames
 (
hOutline, pszDimension, ulOptions, pulCount, pNameArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。
pszDimension	ESS_STR_T	世代名を取得する次元。
ulOptions	ESS_ULONG_T	これは、次のいずれかの値にできます: ● ESS_GENLEV_ALL – デフォルト値と実際の世代名が戻されます。 ● ESS_GENLEV_ACTUAL – 実際に定義されている世代名のみ戻されます。 ● ESS_GENLEV_DEFAULT – すべてのデフォルト世代名が戻されます。これには、実際の名前がある世代のデフォルト名も含まれます。 ● ESS_GENLEV_NOACTUAL – デフォルト世代名が戻されます。これには、実際の名前がない世代のみが含まれます。
pulCount	ESS_PULONG_T	pNameArray に要素数が戻されます。指定したメンバーの世代名の数です。
pNameArray	767 ページの「ESS_GENLEVELNAME_T」	指定された次元に対する世代名の構造体の配列。

備考

- 呼出し元は、**EssFree()**を呼び出して、使用後に pNameArray 構造体を解放する必要があります。
- この呼出しは、**EssOtlOpenOutline()**および **EssOtlOpenOutlineQuery()**の両方に機能します。情報は **EssOtlOpenOutlineQuery()**呼出し中にサーバーから戻されるため、情報は両方に対してローカルに存在します。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_STR_T     Dimension;
ESS_ULONG_T    GenOpt;
ESS_ULONG_T    pCount = 0, i;
ESS_PGENLEVELNAME_T pNameArray = ESS_NULL;
ESS_ACCESS_T    Access;
ESS_STR_T     AppName;
ESS_STR_T     DbName;

AppName = "Sample";
DbName = "Basic";

sts=EssSetActive(hCtx, AppName, DbName, &Access);

if (sts == 0)
```

```
{
 memset(&Object, '\0', sizeof(Object));

 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);

 Dimension = "Year";
 GenOpt = ESS_GENLEV_ALL;

 if (!sts)
 {
  sts = EssOtlGetGenNames(hOutline, Dimension,
     GenOpt, &Count, &pNameArray);

  if(!sts && Count )
  {
   for(i = 0; i<Count; i++)
   {
    printf("\nNumber %ld, Name %s ",
    pNameArray[i].usNumber, pNameArray[i].szName);
   }
   EssFree(hInst, pNameArray);
  }
 }
}
```

関連トピック

- EssFree
- EssOtlGetGenName
- EssOtlGetLevelName
- EssOtlGetLevelNames
- EssOtlOpenOutline
- EssOtlOpenOutlineQuery

EssOtlGetHierarchyType

次元の階層タイプの指定(複合階層使用可能、動的階層、または保管階層)を取得します。

構文

```
ESS_FUNC_M EssOtlGetHierarchyType(
hOutline
,
hMember
,
pType
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。

パラメータ	データ型	説明
hMember	ESS_HMEMBER_T	次元メンバー(入力)。
pType	ESS_UCHAR_T	hMember が次元メンバーである場合は次のいずれかの値(出力): ● ESS_STORED_HIERARCHY – 次元は単一の保管階層です。 ● ESS_DYNAMIC_HIERARCHY – 次元は単一の動的階層です。 ● ESS_MULTIPLE_HIERARCHY_IS_ENABLED – 次元は複合階層使用可能に設定されます。 「注意」を参照してください。

備考

- 次元が複合階層使用可能になると、階層タイプは世代 2 のメンバーによって決定されます。hMember が世代 2 のメンバーである場合、pType は次の値を戻す可能性があります:
  - ESS_STORED_HIERARCHY - hMember が最上位である階層は単一の保管階層です。
  - ESS_DYNAMIC_HIERARCHY - hMember が最上位である階層は単一の動的階層です。
  - ESS_MULTIPLE_HIERARCHY_NOT_ENABLED - 次元は複合階層使用可能ではありません。
- hMember が 2 より大きい世代である場合、pType は ESS_NOT_HIERARCHY_MEMBER を戻します。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

関連トピック

- EssOtlSetHierarchyType
- EssOtlSetAltHierarchyEnabled
- EssOtlGetAltHierarchyEnabled

EssOtlGetImpliedShare

アウトラインの暗黙の共有設定を戻します。

構文

```
ESS_FUNC_M EssOtlGetImpliedShare(
hOutline
,
&impliedShareSetting
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
&impliedShareSetting	ESS_USHORT	暗黙の共有設定のアドレス。

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

暗黙の共有設定値。109 ページの「暗黙の共有設定(C)」を参照してください。

戻り値のパラメータ

ESS_USHORT impliedShareSetting

関連トピック

- EssOtlSetImpliedShare

EssOtlGetLevelName

次元内の特定のレベルの名前を取得します。

構文

```
ESS_FUNC_M
EssOtlGetLevelName
(
hOutline, pszDimension, usLevel, pszName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象の世代を含む次元の名前。
usLevel	ESS_USHORT_T	名前を取得するレベル番号の番号。リーフ・メンバーはレベル 0 です。
pszName	ESS_PSTR_T	指定された次元のレベル名が戻されるバッファ。API によって割り当てられます。

備考

- C プログラムでは、戻りバッファを解放する場合は EssFree()を呼び出します。
- レベル名は自動的には割り当てられません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EssOtlSetLevelName で割り当てられます

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_NO_GENLEVELNAME
- OTLAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    LevelNum;
ESS_STR_T     LevelName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/***************  Get Level Name **************/
Dimension = "Year";
LevelNum = 0;

if (!sts)
{
 sts = EssOtlGetLevelName(hOutline, Dimension,
    LevelNum, &LevelName);
}

if (!sts && LevelName)
{
 printf("Level Name: %s\n",LevelName);
    EssFree(hInst, LevelName);
}
```

関連トピック

- EssOtlSetLevelName
- EssOtlDeleteLevelName
- EssOtlSetGenName

EssOtlGetLevelNameEx

次元内の特定のレベルについて、メンバー名の一意性の設定を戻します。

構文

```
ESS_FUNC_M
EssOtlGetLevelNameEx
(
hOutline, pszDimension, usLevel, pszName, pbNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象の世代を含む次元の名前。
usLevel	ESS_USHORT_T	名前を取得するレベル番号の番号。リーフ・メンバーはレベル 0 です。
pszName	ESS_PSTR_T	指定された次元のレベルを戻すためのバッファ。API によって割り当てられます(出力)。
pbNameUnique	ESS_PBOOL_T	メンバー名の一意性の設定(出力)。

備考

- C プログラムでは、戻りバッファを解放する場合は EssFree()を呼び出します。
- レベル名は自動的には割り当てられません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EssOtlSetLevelName で割り当てられます
- この関数はレベルのメンバー名の一意性情報を取得します。メンバー名の一意性設定を変更するには、EssOtlSetLevelNameEx を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_GetLevelNameEx()
{
ESS_STS_T      sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T      Dimension;
ESS_USHORT_T    LevelNum;
ESS_STR_T      LevelName;
ESS_BOOL_T     bUnique= ESS_FALSE;

memset(&Object, '\0', sizeof(Object));
```

```
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/*************** Set and Get Level Name **************/
 Dimension = "Year";
 LevelNum = 0;
 LevelName = "Level 0 Year";

 //SetLevelNameEx() so that level 0 member of Year cannot be non-unique
 if (!sts)
 {
  sts = EssOtlSetLevelNameEx(hOutline, Dimension,
   LevelNum, LevelName, ESS_TRUE);
 }

 // GetLevelNameEx() to see if the level is able to be non-unique
 if (!sts)
 {
  sts =
     EssOtlGetLevelNameEx
     (hOutline, Dimension,
     LevelNum, &LevelName, &bUnique);
  printf("Level 0 members of Year have bUnique value of %ld\n", bUnique);

 }

 if (!sts && LevelName)
 {
  printf("Level Name: %s\n",LevelName);
     EssFree(hInst, LevelName);
 }

return (sts);
}
```

関連トピック

- EssOtlSetLevelNameEx

EssOtlGetLevelNames

特定の次元に対して指定されたすべてのレベル名を取得します。

構文

```
     ESS_FUNC_M
```

```
EssOtlGetLevelNames
 (
hOutline, pszDimension, ulOptions, pulCount, pNameArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。
pszDimension	ESS_STR_T	レベル名を取得する次元。
ulOptions	ESS_ULONG_T	これは、次のいずれかの値にできます: ● ESS_GENLEV_ALL – デフォルト値と実際のレベル名が戻されます ● ESS_GENLEV_ACTUAL – 実際に定義されたレベル名のみが戻されます ● ESS_GENLEV_DEFAULT – すべてのデフォルト・レベル名が戻されます。これには、実際の名前があるレベルのデフォルト名も含まれます。 ● ESS_GENLEV_NOACTUAL – デフォルト・レベル名が戻されます。これには、実際のレベル名前がないレベルのみが含まれます。
pulCount	ESS_PULONG_T	pNameArray に要素数が戻されます。指定したメンバーのレベル名の数です。
pNameArray	767 ページの「ESS_GENLEVELNAME_T」	指定した次元に対するレベル名の構造体の配列。

備考

- 呼出し元は、**EssFree()**を呼び出して、使用後に pNameArray 構造体を解放する必要があります。
- この呼出しは、**EssOtlOpenOutline()**および **EssOtlOpenOutlineQuery()**の両方に機能します。情報は **EssOtlOpenOutlineQuery()**呼出し中にサーバーから戻されるため、情報は両方に対してローカルに存在します。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_STR_T      Dimension;
ESS_ULONG_T     LevOpt;
ESS_ULONG_T     pCount = 0, i;
ESS_PGENLEVELNAME_T pNameArray = ESS_NULL;
ESS_ACCESS_T    Access;
ESS_STR_T      AppName;
ESS_STR_T      DbName;
```

```
AppName = "Sample";
DbName = "Basic";

sts=EssSetActive(hCtx, AppName, DbName, &Access);

if (sts == 0)
{
 memset(&Object, '\0', sizeof(Object));

 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);

 Dimension = "Year";
 LevOpt = ESS_GENLEV_ALL;

 if (!sts)
 {
  sts = EssOtlGetLevelNames(hOutline, Dimension,
     LevOpt, &Count, &pNameArray);

  if(!sts && Count )
  {
   for(i = 0; i<Count; i++)
   {
    printf("\nNumber %ld, Name %s ",
    pNameArray[i].usNumber, pNameArray[i].szName);
   }
   EssFree(hInst, pNameArray);
  }
 }
}
```

関連トピック

- EssFree
- EssOtlGetGenName
- EssOtlGetGenNames
- EssOtlGetLevelName
- EssOtlOpenOutline
- EssOtlOpenOutlineQuery

EssOtlGetLinkedAttributeAttachLevel

リンク属性次元の添付レベルを取得します。

リンク属性次元は、日時次元メンバー間での期別比較を使用可能にします。それぞれのリンク属性には関連付けレベルと、属性次元定義と関連付けられた添付レベルがあります。

添付レベルは常にリンク属性次元が示す期別比較の 2 つの期間のうち、長いほうです。たとえば、リンク属性次元 Quarter by Year では、Year が添付レベルで、Quarter が関連付けレベルです。

構文

```
ESS_FUNC_M
EssOtlGetLinkedAttributeAttachLevel
 (
hOutline, hDimMember, psLevel
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDimMember	ESS_HMEMBER_T	リンク属性次元メンバー・ハンドル(入力)。
psLevel	ESS_PUSHORT_T	リンク属性添付レベル(出力)。

備考

- この関数を呼び出す前に、EssOtlOpenOutline または EssOtlOpenOutlineQuery のいずれかを使用して、アウトラインを編集モードまたはクエリー・モードで開きます。
- この関数は hDimMember のタイプがリンク属性次元の場合にのみ適用できます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER    hDimMember;
ESS_USHORT_T    usAttachLevel;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
```

```
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Quarter By Year",
    &hDimMember);
}

if (!sts  && hMemberJan)
{
 sts = EssOtlGetLinkedAttributeAttachLevel(hOutline,
    hDimMember, &usAttachLevel);
}
```

関連トピック

- EssOtlGetAttributeAssocLevel
- EssOtlQueryGenerationInfo

EssOtlGetMemberAlias

指定の別名テーブルにおける指定メンバーに対する、デフォルトのメンバーの別名を取得します。

構文

```
ESS_FUNC_M
EssOtlGetMemberAlias
 (
hOutline, hMember, pszAliasTable, ppszAlias
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	別名を取得するメンバーのハンドル。
pszAliasTable	ESS_STR_T	別名を取得する別名テーブル。このパラメータが ESS_NULL の場合、デフォルトの別名テーブルが使用されます。
ppszAlias	ESS_PSTR_T	別名の戻り値が格納されるバッファ。バッファは API によって割り当てられます。

備考

- 別名バッファを解放するには、**EssFree()**を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_ALIASTABLE

例

```
#include <essapi.h>
```

```
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMember;
ESS_STR_T      pszAlias;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "100",
    &hMember);}

if (!sts && hMember)
{
 sts = EssOtlGetMemberAlias(hOutline,
    hMember, ESS_NULL, &pszAlias);
}

if (pszAlias)
{
 EssFree(hInst, pszAlias);
}
```

関連トピック

- EssOtlSetMemberAlias
- EssOtlDeleteMemberAlias

EssOtlGetMemberCommentEx

指定されたメンバーに対する拡張コメントを取得します。

構文

```
ESS_FUNC_M
EssOtlGetMemberCommentEx
 (
hOutline, hMember, pszCommentEx
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	メンバーのハンドル。
pszCommentEx	ESS_PSTR_T	拡張コメントが戻される変数。このバッファは API によって割り当てられます。

備考

- 拡張メンバー・コメントを含むバッファを解放する場合は、**EssFree()**を使用します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T   Object;
ESS_HOUTLINE_T  hOutline;
ESS_HMEMBER_T   hMember;
ESS_STR_T     pszCommentEx = ESS_NULL;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T   szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx   = hCtx;
Object.ObjType  = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName  = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Variance", &hMember);
}

if (!sts  && hMember)
{
 sts = EssOtlGetMemberCommentEx(hOutline, hMember, &pszCommentEx);
}

if (pszCommentEx)
{
 EssFree(hInst, pszCommentEx);
```

```
}
```

関連トピック

- EssFree
- EssOtlOpenOutline
- EssOtlSetMemberCommentEx

EssOtlGetMemberField

指定されたアウトライン・メンバーの指定のフィールドのデータを戻します。

構文

```
ESS_FUNC_M
EssOtlGetMemberField
(
hOutline, hMember, MbrFieldID, ppFieldElement
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。これは EssOtlOpenOutlineQuery()から戻されている必要があります。
hMember	ESS_HMEMBER_T	EssOtlQueryMembersEx()によって戻されるメンバーのハンドル。
MbrFieldID	ESS_ULONG_T	メンバー・フィールド識別子定数。「注意」を参照してください。
ppFieldElement	ESS_PPVOID_T	必須フィールド要素を指すポインタの戻り値。

備考

- EssOtlGetMemberField()は、メンバーのハンドルおよびフィールド識別子を取得し、指定されたフィールドのデータへのポインタを戻します。
- MbrFieldID に対して EssOtlQueryMembersEx()の fieldSelection 文字列内にない定数を指定した場合、EssOtlGetMemberField()はエラー OTLAPI_ERR_MBRINVALID を戻します。
- EssOtlGetMemberField()の呼出し側は、指定されたフィールド・データ用にメモリー・セットを解放するために EssFree()を呼び出す必要があります。
- 次のメンバー・フィールド識別子定数は、MbrFieldID に対して有効な値です:
  - ESS_OTLQRYMBR_NONE
  - ESS_OTLQRYMBR_NAME
  - ESS_OTLQRYMBR_LEVEL
  - ESS_OTLQRYMBR_GENERATION
  - ESS_OTLQRYMBR_CONSOLIDATION
  - ESS_OTLQRYMBR_TWOPASS
  - ESS_OTLQRYMBR_EXPENSE

- ESS_OTLQRYMBR_CURRENCYCONVTYPE
- ESS_OTLQRYMBR_CURRENCYCONVNAME
- ESS_OTLQRYMBR_TIMEBALANCE
- ESS_OTLQRYMBR_SKIP
- ESS_OTLQRYMBR_SHARE
- ESS_OTLQRYMBR_STORAGE
- ESS_OTLQRYMBR_CATEGORY
- ESS_OTLQRYMBR_STORAGECATEGORY
- ESS_OTLQRYMBR_COMMENT
- ESS_OTLQRYMBR_CHILDCOUNT
- ESS_OTLQRYMBR_NUMBER
- ESS_OTLQRYMBR_DIMNAME
- ESS_OTLQRYMBR_DIMNUMBER
- ESS_OTLQRYMBR_ALIASNAME
- ESS_OTLQRYMBR_NEXTNAME
- ESS_OTLQRYMBR_PREVNAME
- ESS_OTLQRYMBR_PARENTNAME
- ESS_OTLQRYMBR_CHILDNAME
- ESS_OTLQRYMBR_UDA
- ESS_OTLQRYMBR_FORMULA
- ESS_OTLQRYMBR_LASTFORMULA
- ESS_OTLQRYMBR_EXTCOMMENT
- ESS_OTLQRYMBR_ALIASCOMBO
- ESS_OTLQRYMBR_VALID
- ESS_OTLQRYMBR_CURRENCYCONVDB
- ESS_OTLQRYMBR_STATUS
- ESS_OTLQRYMBR_ATTRIBUTED

  True - 属性が関連付けられている場合
- ESS_OTLQRYMBR_ASSOCATTRDIMNAME

  関連付けられている属性次元の名前
- ESS_OTLQRYMBR_ASSOCATTRMEMNAME

  関連付けられている属性メンバーの名前
- ESS_OTLQRYMBR_ASSOCATTRVALUE

  関連付けられている属性値
- ESS_OTLQRYMBR_ATTRVALUE

メンバーの属性値

- ESS_OTLQRYMBR_UNIQUENAME

  一意のメンバー名

- ESS_OTLQRYMBR_FORMATSTRING

  メンバーのフォーマット文字列

- ESS_OTLQRYTIDIM_TIMEPERIODS

  期間リストのクエリー時間次元

- ESS_OTLQRYMBR_MBRINFO

戻り値

関数の呼出しが正常終了の場合、戻り値は 0 になります。

例

EssOtlQueryMembersEx()、**EssOtlGetMemberField()**および **ESS_OTLQUERYERRORLIST_T** を使用し、**EssOtlFreeMembers()**および **EssFree()**の呼出しを含んでいる例については、1070 ページの「拡張メンバーのクエリー・コードの例」を参照してください。

関連トピック

- EssFree
- EssOtlGetDimensionUserAttributes
- EssOtlOpenOutlineQuery
- EssOtlQueryMembers
- EssOtlQueryMembersByName
- EssOtlQueryMembersEx

EssOtlGetMemberFormula

指定されたメンバーの式を取得します。

構文

```
ESS_FUNC_M
EssOtlGetMemberFormula
(
hOutline, hMember, pszFormula
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	メンバーのハンドル。
ppszFormula	ESS_PSTR_T	メンバー式が戻される変数。このバッファは API によって割り当てられます。

備考

- 式バッファを解放するには、**EssFree()**を使用します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMember;
ESS_STR_T     pszFormula = ESS_NULL;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Variance",
    &hMember);
}

if (!sts  && hMember)
{
 sts = EssOtlGetMemberFormula(hOutline,
    hMember, &pszFormula);
}
if (pszFormula)
{
 EssFree(hInst, pszFormula);
}
```

関連トピック

- EssFree
- EssOtlDeleteMemberFormula
- EssOtlGetMemberLastFormula

- EssOtlOpenOutline
- EssOtlOpenOutlineQuery
- EssOtlSetMemberFormula

EssOtlGetMemberInfo

指定されたメンバーの情報を取得します。

構文

```
ESS_FUNC_M
EssOtlGetMemberInfo
 (
hOutline, hMember, pInfo
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	メンバーのハンドル。
pInfo	768 ページの「ESS_MBRINFO_T」	メンバー情報構造体へのポインタ。API によって割り当てられます。

備考

- メンバーのハンドルを取得する場合は、**EssOtlFindMember()**を呼び出します。
- 情報構造体を解放するには、**EssFreeStructure()**を呼び出します。
- 768 ページの「ESS_MBRINFO_T」構造体の 2 つのフィールドは、属性用としてのみ使用されます:
  - fAttributed
  - Attribute

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER    hMemberJan;
ESS_PMBRINFO_T   pMbrInfo;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
```

```
memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}

if (!sts  && hMemberJan)
{
 sts = EssOtlGetMemberInfo(hOutline,
    hMemberJan, &pMbrInfo);
}

if (pMbrInfo)
{
 EssOtlFreeStructure(hOutline, ESS_DT_STRUCT_MBRINFO, 1, pMbrInfo);
}
```

EssOtlGetMemberInfoArray

指定されたメンバー配列のメンバー情報を取得します。

構文

```
ESS_FUNC_M
EssOtlGetMemberInfoArray
 (
hOutline, memberCount, hMemberArr, pInfoArr, pStsArr
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル。
memberCount	ESS_SHORT_T	入力配列のメンバー数。
hMemberArr	ESS_HMEMBER_T	memberCount メンバーのハンドルの配列。
pInfoArr	ESS_PPMBRINFO_T	memberCount メンバーの情報ポインタの配列。

パラメータ	データ型	説明
pStsArr	ESS_STS_T	memberCount ステータスの戻りコードの配列。 複数のエラーが発生した場合は、関数により、最初に発生したエラーの値が戻されます。

備考

- メンバーのハンドルを取得する場合は、**EssOtlFindMember()**を呼び出します。
- 情報構造体を解放するには、**EssFreeStructure()**を呼び出します。
- 768 ページの「ESS_MBRINFO_T」構造体の 2 つのフィールドは、属性用としてのみ使用されます:
  - fAttributed
  - Attribute

戻り値

成功の場合、0 が戻されます。失敗の場合、各メンバーの戻りコードが pStsArr に設定されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER    hMemberArr[3];
ESS_PMBRINFO_T   pMbrInfoArr[3];
ESS_STS_T     stsArr[3];
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_SHORT_T    i;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
ESS_TRUE, &hOutline);
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan", &hMemberArr[0]);
}
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Feb", &hMemberArr[1]);
}
```

```
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Mar", &hMemberArr[2]);
}
if (!sts)
{
 sts = EssOtlGetMemberInfoArray(hOutline, 3, hMemberArr, pMbrInfoArr, stsArr);
}
for (i = 0; i < 3; i++)
{
 if (pMbrInfoArr[i])
 {
  EssOtlFreeStructure(hOutline, ESS_DT_STRUCT_MBRINFO, 1, pMbrInfoArr[i]);
 }
}
```

EssOtlGetMemberLastFormula

メンバーの計算に使用された最後の式を戻します。

構文

```
ESS_FUNC_M
EssOtlGetMemberLastFormula
 (
hOutline, hMember, ppszFormula
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	メンバーのハンドル。
ppszFormula	ESS_PSTR_T	メンバー式が戻される変数。このバッファは API によって割り当てられます。

備考

- 式バッファを解放するには、**EssFree()**を使用します。
- この呼出しは、**EssOtlOpenOutline()**および **EssOtlOpenOutlineQuery()**の両方に機能します。
- **EssOtlGetMemberLastFormula()**は、選択されたメンバーに前回適用された式を戻します。その式は、そのメンバーと関連付けられたデータベース・アウトライン式とは異なる場合があります。
- 最後の式は、そのメンバーに対して最後に実行された計算(アウトラインまたは計算スクリプト)から導出されます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <ESSAPI.H>
#include <ESSOTL.H>

ESS_STS_T     sts = 0 ;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_HMEMBER_T    hMember;
ESS_STR_T     pszFormula = ESS_NULL;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

 if (!sts)
 {
   sts = EssOtlFindMember(hOutline, "Margin", &hMember);
 }

 if (!sts && hMember)
 {
   sts = EssOtlGetMemberLastFormula(hOutline, hMember, &pszFormula);
      printf("Member Last Formula: %s\n",pszFormula);
 }

 if (pszFormula)
 {
   EssFree(hInst, pszFormula);
 }
```

関連トピック

- EssFree
- EssOtlDeleteMemberFormula
- EssOtlGetMemberFormula
- EssOtlOpenOutline
- EssOtlOpenOutlineQuery
- EssOtlSetMemberFormula

EssOtlGetMemberSmartList

入力アウトライン・メンバーと関連付けられたテキスト・リスト(スマートリスト)を戻します。

構文

```
ESS_FUNC_M EssOtlGetMemberSmartList(
hOutline
,
hMember
,
*phSmartlist
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
hMember	ESS_HMEMBER_T	アウトライン・メンバー・ハンドル
*phSmartlist	ESS_HSMARTLIST_T	関連するテキスト・リスト(スマートリスト)ハンドルを戻します

戻り値

戻り値:

- 0 - 正常終了の場合

  *phSmartlist は戻り値を含みます。

- エラー番号 - 失敗した場合

  *phSmartlist は NULL です。

例

```
    void TestGetMemberSmartList()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T             hOutline = ESS_NULL;
    ESS_OBJDEF_T                 Object;
    ESS_HMEMBER_T              hMember;
    ESS_HSMARTLIST_T         hSmartList;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =               hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =      szFileName;

    /* Open outline */
    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);
```

```
    /* Find member */
    sts = EssOtlFindMember(hOutline, "Original Price",
                &hMember);

    /* Return SmartList associated with member */
    sts =
    EssOtlGetMemberSmartList(hOutline, hMember,
                &hSmartList);



    /* Unlock object */
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlGetMemberSolveOrder

メンバーの解決順を戻します。

構文

```
ESS_FUNC_M EssOtlGetMemberSolveOrder (
hOutline, hMember, pOrder
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。

パラメータ	データ型	説明
hMember	ESS_HMEMBER_T	メンバーのハンドル(入力)。
pOrder	ESS_PUCHAR_T	解決順(出力)。

備考

- 解決順は集約ストレージ・データベースにのみ適用できます。
- メンバーまたは次元の解決順プロパティは、計算順序を指定します。
- メンバーの解決順は次元の解決順よりも優先されます。解決順は 0 から 127 までにできます。デフォルトは 0 です。
- 解決順が指定されていない式を持たないメンバーは、その次元の解決順を継承します。解決順が指定されていない式を持つメンバーは、ゼロの解決順を持ちます。

戻り値

成功の場合、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_HMEMBER_T   hMember = ESS_NULL;
ESS_UCHAR_T    ucOrder = 0;

/* code to assign hOutline variable omitted */
/* code to assign hMember variable omitted */


if (hOutline && hMember)
{
 sts = EssOtlGetMemberSolveOrder(hOutline, hMember, &ucOrder);

 if (sts)
  printf("Error [%ld] returned\n", sts);
 else
  printf("Solve Order: %d\n", ucOrder);
}
else
 printf("Both hOutline and hMember must have values\n");
```

EssOtlSetMemberSolveOrder

EssOtlSetDimensionSolveOrder

EssOtlGetDimensionSolveOrder

EssOtlGetMemberType

入力アウトライン・メンバーのメンバー・タイプを戻します。

構文

```
ESS_FUNC_M EssOtlGetMemberType(
hOutline, hMember, *pusType
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル
hMember	ESS_HMEMBER_T	アウトライン・メンバー・ハンドル
*pusType	ESS_USHORT_T	アウトライン・メンバーのタイプ: • ESS_MEMBERTYPE_NUMERIC メンバー・タイプは数値です。 • ESS_MEMBERTYPE_SMARTLIST メンバー・タイプはテキストで、関連するテキスト・リスト(SmartList)オブジェクトがあります。 • ESS_MEMBERTYPE_DATE メンバー・タイプは日付型です。

戻り値

戻り値:

- 0 - 正常終了の場合

  pusType には値が含まれます。
- エラー番号 - 失敗した場合

  pusType は NULL です。

例

```
    void TestGetSetMemberType()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T                Object;
    ESS_HMEMBER_T            hMember;
    ESS_USHORT_T                usMemberType;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =              hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =          szFileName;

    /* Open outline */
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
                ESS_TRUE, &hOutline);
```

```
    /* Find a member */
    sts = EssOtlFindMember(hOutline, "Original Price", &hMember);

    /* Get Member Type of an outline that is not member
        type enabled */
    /* Get original type */
    sts =
    EssOtlGetMemberType(hOutline, hMember, &usMemberType);

        DisplayMemberType(usMemberType); /* a display function */

    /* Get Member Type of an outline that is member
        type enabled */
    EnableSmartList(hOutline);

    /* Get original type */
    sts =
    EssOtlGetMemberType(hOutline, hMember, &usMemberType);

        printf("EssOtlGetMemberType sts: %d\n", sts);
    DisplayMemberType(usMemberType);

    /* Set type to NUMERIC */
    usMemberType = ESS_MEMBERTYPE_NUMERIC;
    sts = EssOtlSetMemberType(hOutline, hMember, usMemberType);
    printf("EssOtlSetMemberType sts: %d\n",sts);

    sts =
    EssOtlGetMemberType(hOutline, hMember, &usMemberType);

    printf("EssOtlGetMemberType sts: %d\n", sts);
    DisplayMemberType(usMemberType);


    /* Clean up */
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount

- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlGetMemberUniqueName

メンバー名が一意である場合にはメンバー名を戻し、メンバー名が重複している場合には、メンバーの区別に必要な最小限の修飾名を戻します。

構文

```
ESS_FUNC_M EssOtlGetMemberUniqueName (
hOutline, hMember, *szFullName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	メンバーのハンドル(入力)。
*szFullName	ESS_STR_T	戻されたメンバー名または修飾されたメンバー名(出力)。

備考

- この関数を呼び出す前に、EssOtlOpenOutline を呼び出して編集モードでアウトラインを開くか、EssOtlOpenOutlineQuery を呼び出してクエリー・モードでアウトラインを開いてください。
- この関数の 2 番目の引数のメンバー・ハンドルを取得するには、メンバー走査関数を使用します。
- 重複するメンバー名が許可されるアウトラインで、渡されたメンバー・ハンドルが拡張された共有メンバー、または標準の共有メンバーである場合には、この関数は一意の名前を戻します。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

例 1

次の例では、この関数の出力は、Qtr1 の完全修飾メンバー名:[2004].[Qtr1]です

```
    ESS_FUNC_M ESS_GetMemberUniq()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
```

```
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
 ESS_STR_T  szFullName;
ESS_HMEMBER_T  hMemberParent;
ESS_HMEMBER_T  hMemberChild;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "2004", &hMemberParent);
}

if (!sts && hMemberParent)
{
 sts = EssOtlGetChild(hOutline, hMemberParent, &hMemberChild);
}


/*Get the qualified name of the first child of 2004, Qtr1*/
if (!sts)
{
 sts = EssOtlGetMemberUniqueName (hOutline, hMemberChild, &szFullName);

 if (sts)
  printf("EssOtlGetMemberUniqueName failed sts %ld\n",sts);
 else
  printf("Qtr1's qualified name is: %s\n", szFullName);
}


return sts;
}
```

例 2

次の例は、クエリー・モードでのこの関数の使用を示しています。

```
 member_fields   = "<SelectMbrInfo (membername, uniquename) ";
 member_selection  = "@SHARE(@DESCENDANTS(product))";
```

```
MaxCount     = -1;
phMemberArray   = ESS_NULL;
pqryErrorList   = ESS_NULL;

status = EssOtlQueryMembersEx(hOutline,
           member_fields,
           member_selection,
           &MaxCount,
           &phMemberArray,
           &pqryErrorList);

if (status) goto exit;

for (int i = 0; i < MaxCount; i++)
{
 status = EssOtlGetMemberField(hOutline, phMemberArray[i], ESS_OTLQRYMBR_NAME,
  (ESS_PPVOID_T) &pName);
 if (status) goto exit;

 status = EssOtlGetMemberUniqueName(hOutline, phMemberArray[i], &pUniqueName2);
 if (status) goto exit;
}
```

EssOtlGetNextSharedMember

指定されたメンバーの次の共有メンバーにメンバー・ハンドルを戻します。

構文

```
ESS_FUNC_M
EssOtlGetNextSharedMember
 (
hOutline, hMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	次の共有メンバーを検索するメンバーのハンドル。
phMember	ESS_PHMEMBER_T	アウトラインの次の共有メンバーのメンバー・ハンドルを戻すポインタ。 共有メンバーがもうない場合は、このパラメータは ESS_NULL です。

備考

- hmember が実際のメンバーの場合、最初の共有メンバーが phMember パラメータに戻されます。hmember が共有メンバーの場合、次の共有メンバーが phMember パラメータに戻されます。
- 共有メンバーが(それ以上)存在しない場合には、phMember が ESS_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T   hMember;
ESS_HMEMBER_T   hMemberShared;
ESS_HMEMBER_T   hNextShared;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "200-20",
    &hMember);
}

if (!sts  && hMember)
{
/* get first shared member of actual member */
 sts = EssOtlGetNextSharedMember(hOutline, hMember, &hMemberShared);

/* do something with hMemberShared */
/* get next shared member, if any*/

 while(!sts && hMemberShared)
 {
  sts = EssOtlGetNextSharedMember(hOutline,
     hMemberShared, &hNextShared);
      hMemberShared = hNextShared;

/* do something with hMemberShared */
 }
}
```

関連トピック

- EssOtlFindMember

EssOtlGetNextSibling

メンバーの次の兄弟を戻します。

構文

```
ESS_FUNC_M
EssOtlGetNextSibling
(
hOutline, hMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	取得する兄弟を持つメンバーのハンドル
phMember	ESS_PHMEMBER_T	hMember パラメータの兄弟のメンバーのハンドルの戻り値を指すポインタ

備考

- 次の兄弟が存在しない場合は、*phMember が ESS_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberJan;
ESS_HMEMBER_T    hMemberSibling;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
```

```
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}

if (!sts  && hMemberJan)
{
 sts = EssOtlGetNextSibling(hOutline,
    hMemberJan, &hMemberSibling);
}
```

関連トピック

- EssOtlGetPrevSibling
- EssOtlGetParent
- EssOtlGetChild
- EssOtlGetFirstMember

EssOtlGetNumQueryHints

アウトラインのすべてのクエリー・ヒントのヒント・メンバーを戻します。

構文

```
ESS_FUNC_M EssOtlGetNumQueryHints (
hOutline, pNumHints
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
pNumHints	ESS_PSHORT_T	クエリー・ヒント・メンバー(出力)の配列へのポインタ。

備考

- 共通クエリーのプロファイルについて Essbase に通知することにより、クエリー・ヒントは標準ビュー選択に影響を及ぼすことができます。
- この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。

戻り値

正常終了の場合は、0 が戻されます。

例

EssOtlSetQueryHint を参照してください。

関連トピック

- EssOtlAddQueryHint
- EssOtlSetQueryHint
- EssOtlGetQueryHint
- EssOtlGetQueryHintSize
- EssOtlDeleteQueryHint

EssOtlGetObjectReferenceCount

入力オブジェクト・ハンドルを参照するアウトライン・メンバーの数を戻します。

構文

```
ESS_FUNC_M EssOtlGetObjectReferenceCount(
hOutline, objHandle, pCount
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objHandle	ESS_HOBJECT_T	インポートまたはエクスポートされるオブジェクト・ハンドル
pCount	ESS_ULONG_T*	アウトライン・メンバーのカウント

戻り値

戻り値:

- 0 - 正常終了の場合

  pCount は値を含みます。

- エラー番号 - 失敗した場合

  pCount は NULL です。

例

```
    void TestGetObjectReferenceCount()
{
    ESS_STS_T         sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T         hOutline = ESS_NULL;
    ESS_OBJDEF_T         Object;
    ESS_HOBJECT_T         hObjHandle = ESS_NULL;
    ESS_ULONG_T             Count = 0;
    ESS_OBJECT_TYPES     objType;
    ESS_STR_T             objName;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
```

```
    Object.FileName = szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Get count of an object that is referenced */
    objType = OBJECT_SMARTLIST;
    objName = "Smartlist1";
    sts = EssOtlFindObject(hOutline, objType,
                objName, &hObjHandle);
    printf("EssOtlFindObject sts: %ld\n",sts);

    sts =
    EssOtlGetObjectReferenceCount(hOutline,
                hObjHandle, &Count);

    printf("EssOtlGetObjectReferenceCount sts: %ld\n",sts);
    printf("\tCount returned: %d\n", Count);

    sts = EssUnlockObject(hCtx, Object.ObjType,
        Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlGetObjectReferences

入力オブジェクト・ハンドルを参照するアウトライン・メンバーの配列を戻します。この関数に続けて、EssFree を使用して phMembers の割当てを解除する必要があります。

構文

```
    ESS_FUNC_M EssOtlGetObjectReferences(
```

```
hOutline, objHandle, ulMaxCount, phMembers, pulNumMembers
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objHandle	ESS_HOBJECT_T	オブジェクト・ハンドル
ulMaxCount	ESS_ULONG_T	クライアントが処理できるアウトライン・メンバーの最大カウント
phMembers	ESS_HMEMBER_T*	アウトライン・メンバーの出力配列
pulNumMembers	ESS_ULONG_T*	戻されたアウトライン・メンバーの数。

戻り値

戻り値:

- 0 - 正常終了の場合

  ulMaxCount、phMembers、および pulNumMembers には値が含まれます。
- エラー番号 - 失敗した場合

  ulMaxCount、phMembers、および pulNumMembers は NULL です。

例

```
    void TestGetObjectReferences()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T                Object;
    ESS_HOBJECT_T            hObjHandle = ESS_NULL;
    ESS_ULONG_T                ulMaxCount;
    ESS_HMEMBER_T            hMembers[256];
    ESS_ULONG_T                ulNumMembers, i;
    ESS_OBJECT_TYPES        objType;
    ESS_STR_T                    objName;
    ESS_PMBRINFO_T       pMbrInfo;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =             hCtx;
    Object.ObjType =         ESS_OBJTYPE_OUTLINE;
    Object.AppName =         szAppName;
    Object.DbName =         szDbName;
    Object.FileName =         szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Get the member(s) of the object that is referenced */
    objType = OBJECT_SMARTLIST;
    objName = "SmartList1";
    sts = EssOtlFindObject(hOutline, objType, objName, &hObjHandle);
```

```
    ulMaxCount = 256;
    sts =
    EssOtlGetObjectReferences(hOutline, hObjHandle,
                ulMaxCount, hMembers, &ulNumMembers);

    printf("EssOtlGetObjectRefrences sts: %ld\n",sts);

    for(i = 0; i < ulNumMembers; i++)
    {
        sts = EssOtlGetMemberInfo(hOutline, hMembers[i], &pMbrInfo);
        if(pMbrInfo)
            printf("\tMember: %s\n", pMbrInfo->szMember);
    }

    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlGetOriginalMember

共有または拡張共有メンバーの元のメンバー名が戻されます。共有メンバーでない場合は、戻り値は NULL です。この関数は完全に修飾された元のメンバー名が戻されます。

構文

```
ESS_FUNC_M EssOtlGetOriginalMember (
hOutline, hMember, ppOriMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	メンバー名(入力)。
ppOriMember	ESS_PSTR_T	元のメンバー名(出力)。

備考

- この関数は、編集モードとクエリー・モードのどちらでも動作します。
- すべてのメンバー名が一意であるアウトラインでこの関数を使用した場合は、無効です。
- 重複するメンバー名が許可されるアウトラインで、渡されたメンバー・ハンドルが拡張の共有メンバーまたは標準の共有メンバーである場合には、この関数はパス式として元のメンバーを戻します。
- 次の階層では、[Diet].[100-10]に対応するメンバー・ハンドルをこの関数に渡すと、[200].[100-10]が戻されます。

```
        100
  100-10
200
  100-10 (duplicate)
Diet
  100-10 (shared with [200.100-10])
```

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

Sample Basic 共有メンバー 100-10 で戻される「元のメンバー」は[100].[100-20]であるとします。

```
    ESS_FUNC_M ESS_GetOrigMember()
{

ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hMember = ESS_NULL, ChildMember = ESS_NULL;
ESS_STR_T   OriMember;



memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
```

```
strcpy(szFileName, "Basic");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);

// sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Diet", &hMember);
}

//Get member handle for shared member "100-10"
if (!sts && hMember)
{
 sts = EssOtlGetChild(hOutline, hMember, &ChildMember);
}


 if (!sts && ChildMember)
 {
  sts = EssOtlGetOriginalMember (hOutline, ChildMember, &OriMember);
  printf("Original member for shared member \"100-10\" is: %s", OriMember);
 }

return sts;
}
```

関連トピック

- EssOtlSetOriginalMember

EssOtlGetOutlineInfo

アウトライン・ファイルに関する情報を戻します。

構文

```
ESS_FUNC_M
EssOtlGetOutlineInfo
 (
hOutline, ppInfo
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
ppInfo	775 ページの「ESS_OUTLINEINFO_T」	アウトライン情報を保管するために、API によって割り当てられた構造体へのポインタ。

備考

- 情報構造体を解放するには、EssFree()を使用します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T  sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_POUTLINEINFO_T pInfo = ESS_NULL;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T     szAppName;
ESS_DBNAME_T     szDbName;
ESS_OBJNAME_T     szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlGetOutlineInfo(hOutline, &pInfo);
}

if(pInfo)
{
 EssFree(hInst, pInfo);
}
```

関連トピック

- EssFree
- EssOtlSetOutlineInfo

EssOtlGetParent

メンバーの親を戻します。

構文

```
ESS_FUNC_M
EssOtlGetParent
 (
hOutline, hMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	親を取得するメンバーのハンドル。
phMember	ESS_PHMEMBER_T	hMember パラメータの親のメンバーのハンドルの戻り値を指すポインタ。

備考

- 親が存在しない場合は、*phMember が ESS_NULL に設定され、呼出しは 0 を戻します。(hMember は次元です。)

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T   hMemberChild;
ESS_HMEMBER_T   hMemberParent;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberChild);
```

```
}

if (!sts  && hMemberChild)
{
 sts = EssOtlGetParent(hOutline,
    hMemberChild, &hMemberParent);
}
```

関連トピック

- EssOtlGetChild
- EssOtlGetNextSibling
- EssOtlGetPrevSibling
- EssOtlGetFirstMember

EssOtlGetPrevSibling

メンバーの前の兄弟を戻します。

構文

```
ESS_FUNC_M
EssOtlGetPrevSibling
 (
hOutline, hMember, phMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	前の兄弟を取得するメンバーのハンドル。
phMember	ESS_PHMEMBER_T	hMember パラメータの前の兄弟のメンバーのハンドルの戻り値を指すポインタ。

備考

- 前の兄弟が存在しない場合は、*phMember が ESS_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberFeb;
ESS_HMEMBER_T    hMemberSibling;
```

```
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

if (!sts)
{
sts = EssOtlFindMember(hOutline, "Feb",
   &hMemberFeb);
}

if (!sts  && hMemberFeb)
{
  sts = EssOtlGetPrevSibling(hOutline,
     hMemberFeb, &hMemberSibling);
}
```

関連トピック

- EssOtlGetNextSibling
- EssOtlGetParent
- EssOtlGetChild
- EssOtlGetFirstMember

EssOtlGetQueryHint

入力アウトラインとヒント番号で示されたクエリーが戻されます。

ヒントには 1 から n までの番号が付けられます。最初のクエリー・ヒントのヒント番号は 1 です。新しい各クエリー・ヒントがリストの終わりへ追加され、番号は 1 ずつ大きくなります。

構文

```
ESS_FUNC_M EssOtlGetQueryHint (
hOutline, hintNum, numMembers, pMemberArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hintNum	ESS_SHORT_T	クエリー・ヒント番号(入力)。
numMembers	ESS_SHORT_T	提供された配列が保持できるメンバー数 - 通常はアウトライン内の実際の次元数です(入力)
pMemberArray	ESS_PHMEMBER_T	ヒントのメンバー配列。通常、配列には実際の次元ごとに 1 メンバーがあり、ヒントに含まれていない次元には NULL が使用されます。この配列には numMembers のサイズを割り当てる必要があります。(出力)

備考

- 共通クエリーのプロファイルについて Essbase に通知することにより、クエリー・ヒントは標準ビュー選択に影響を及ぼすことができます。
- この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。

戻り値

正常終了の場合は、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_SHORT_T    nmHints = 0;
ESS_SHORT_T    i, j, hintNum;
ESS_HMEMBER_T   hMember[10]; /* (nm real dimensions) < 10 */

/* clear array just to be safe */
memset(hMember, 0x00, 10*sizeof(ESS_HMEMBER_T));

/* Code to assign hOutline variable omitted */

sts = EssOtlGetNumQueryHints(hOutline, &nmHints);
if (sts) return sts;  /* error out */

for (i = 0; i < nmHints; i++)
{
 hintNum = i+1;
 sts = EssOtlGetQueryHint(hOutline, hintNum, 10, hMember);
 if (sts) return sts;  /* error out */

 for (j = 0; j < 10; j++)
 {
  if (hMember[j] != AD_NULL)
  {
   sts = EssOtlGetMemberInfo(hOutline, hMember[j], &pMemberInfo);
   if (sts) return sts;  /* error out */
   printf("Hint (%d), member (%d): [%s]\n",
             hintNum, j, pMemberInfo->szMember);
   /* Code to free pMemberInfo omitted */
  }
  else
```

```
 {
  printf("Hint (%d), member (%d): [NULL]\n", hintNum, j);
 }
}
}
```

関連トピック

- EssOtlAddQueryHint
- EssOtlSetQueryHint
- EssOtlGetNumQueryHints
- EssOtlGetQueryHintSize
- EssOtlDeleteQueryHint

EssOtlGetQueryHintSize

アウトラインで定義されたクエリー・ヒントのサイズ(メンバー数)を戻します。

ヒントには 1 から n までの番号が付けられます。最初のクエリー・ヒントのヒント番号は 1 です。新しい各クエリー・ヒントがリストの終わりへ追加され、番号は 1 ずつ大きくなります。

構文

```
ESS_FUNC_M EssOtlGetQueryHintSize (
hOutline, pHintSize
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
pHintSize	ESS_SHORT_T	クエリー・ヒントのサイズ(出力)。

備考

通常、クエリー・ヒントのメンバー数は、実際の次元の数と同じです。ただし、ヒントを追加した後に次元を追加または削除すると、hMember 配列のメンバー数が実際の次元の数と異なってくることがあります。この関数は、GetQueryHint にメンバー配列の大きさを戻します。

戻り値

正常終了の場合は、0 が戻されます。

関連トピック

- EssOtlAddQueryHint
- EssOtlSetQueryHint
- EssOtlGetQueryHint
- EssOtlGetNumQueryHints
- EssOtlDeleteQueryHint

EssOtlGetSmartListInfo

hSmartList ハンドルに渡されたテキスト・リスト(スマートリスト)のテキスト・リスト(スマートリスト)情報を戻します。この後、ppSmartListInfo で EssOtlFreeSmartListInfo を呼び出す必要があります。

構文

```
ESS_FUNC_M EssOtlGetSmartListInfo(hOutline, hSmartList, **ppSmartListInfo);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル
hSmartlist	ESS_HSMARTLIST_T	テキスト・リスト(スマートリスト)ハンドル
**ppSmartListInfo	ESS_SMARTLISTINFO_T	テキスト・リスト(スマートリスト)情報構造体を含みます。

戻り値

戻り値:

- 0 - 正常終了の場合

  ppSmartListInfo はテキスト・リスト(スマートリスト)情報を含みます。

- エラー番号 - 失敗した場合

  ppSmartListInfo は NULL です。

例

```
    DisplaySmartListInfo(ESS_HOUTLINE_T hOutline, ESS_PHOBJECT_T ObjHandles)
{
    ESS_STS_T                          sts = ESS_STS_NOERR;
    ESS_PSMARTLISTINFO_T    SmartListInfo;
    ESS_ULONG_T                      i;

    sts =
    EssOtlGetSmartListInfo(hOutline, ObjHandles,
                                &SmartListInfo);

    if(!sts)
    {
        printf("\n");
        printf("\tName: %s\n", SmartListInfo->szName);
        printf("\tMissing Name: %s\n",
            SmartListInfo->szMissingName);
        printf("\tOut of Range Name: %s\n",
            SmartListInfo->szOutOfRangeName);
        printf("\tusLen: %d\n", SmartListInfo->usLen);
        for (i = 0; i < SmartListInfo->usLen; i++)
        {
            printf("\tpIDs: %d, \tpszText[%d]: %s\n",
                        SmartListInfo->pIDs[i], i,
                        SmartListInfo->ppszText[i]);
```

```
        }
        printf("\n");
    }
    else
        printf("\t\tEssOtlGetSmartListInfo  sts: %d\n",sts);

    if(SmartListInfo)
        sts = EssOtlFreeSmartListInfo(hOutline, SmartListInfo);

}
```

EssOtlGetServerDateFormats

この関数は、サポートされているサーバーの日付フォーマットのリストを戻します。

構文

```
 ESS_FUNC_M EssOtlGetServerDateFormats(
ESS_HCTX_T hCtx,
ESS_STR_T localeStr,
ESS_USHORT_T* pcount,
ESS_STR_T** ppdateStrings,
ESS_STR_T** ppformatStrings)
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	サーバーのコンテキスト・ハンドル
localeStr	ESS_STR_T	サンプルの日付文字列が生成されるロケール。 ● localeStr が空の場合、デフォルトの環境ロケールを生成して日付文字列を生成します ● localeStr が無効の場合、無効を示すエラー・メッセージが戻されます ● localeStr が NULL の場合、エラー・メッセージが戻されます
pcount	ESS_USHORT_T*	サポートされる日付フォーマットの数
ppdateStrings	ESS_STR_T**	配列として、異なる日付フォーマットでサンプルの現在の日付を戻します(割当て解除される)。
ppformatStrings	ESS_STR_T**	サポートされるフォーマットの配列を戻します(割当て解除される)。

戻り値

戻り値:

- 0 - 正常終了の場合

  値は ppdateStrings および ppformatStrings に含まれます。
- エラー番号 - 失敗した場合

例

```
    void TestGetSetDateFormatString()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T                Object;
    ESS_SHORT_T                length = 80;
    ESS_STR_T                    dateFormatString = "";
    ESS_STR_T                    localeStr;
    ESS_USHORT_T                count, i;
    ESS_STR_T*                 pdateStrings;
    ESS_STR_T*                 pformatStrings;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =          hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =      szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Get current value */
    sts = EssOtlGetDateFormatString(hOutline, &dateFormatString);
    printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
    printf("\tDate format string: %s\n", dateFormatString);

    printf("\n");
    localeStr = "English_UnitedStates.Latin1@Binary";
    sts =
    EssOtlGetServerDateFormats(hCtx, localeStr,
                &Count, &pdateStrings, &pformatStrings);

    printf("EssOtlGetServerDateFormats sts: %d \n", sts);

    for (i = 0; i < count; i++)
    {
        printf("\nCase with %s:\n", pformatStrings[i]);
        sts = EssOtlSetDateFormatString(hOutline,
                    pformatStrings[i]);
        printf("EssOtlSetSMDateFormatString sts: %d \n", sts);
        SaveOutline(hOutline);

        sts = EssOtlGetDateFormatString(hOutline,
                    &dateFormatString);
        printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
        printf("\tDate format string: %s\n", dateFormatString);

    }
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline  sts: %d\n",sts);
```

```
}
```

関連トピック

- EssOtlSetDateFormatString
- EssOtlGetDateFormatString

EssOtlGetUpdateTime

指定したアウトラインのタイムスタンプが戻されます。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトライン・ハンドル
pOtlTimeStamp;	ESS_PTIME_T	アウトラインのタイムスタンプへのポインタ

備考

- 時刻の値はタイプ ESS_ULONG_T で、00:00:00 1/1/1970 GMT からの秒数で示されます。
- 時刻の値には永続性はありません。したがって、サーバーがデータベースをロードするとリセットされます。

戻り値

指定したアウトラインのタイムスタンプが戻されます。

例

```
     ESS_HOUTLINE_T hOutline;
ESS_TIME_T   TimeStamp;

sts = EssOtlGetUpdateTime(hOutline, &TimeStamp);
```

関連トピック

- EssOtlGetOutlineInfo
- EssOtlSetOutlineInfo
- EssOtlVerifyOutline
- EssOtlSortChildren
- EssOtlGenerateCurrencyOutline

EssOtlGetUserAttributes

メンバーのユーザー定義属性をすべて取得します。

構文

```
      ESS_FUNC_M
      EssOtlGetUserAttributes
       (
      hOutline, hMember, pusCount,
ppAttributeList
      );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	ユーザー定義属性を取得するメンバーのハンドル。
pusCount	ESS_PUSHORT_T	戻されるユーザー属性の数。ppAttributeList 配列の要素数を定義します。
ppAttributeList	ESS_PPMBRNAME_T	*pusCount メンバーの配列。配列の要素には 1 つのユーザー定義属性文字列が含まれます。

備考

- 呼出し元は **EssOtlSetUserAttribute()**呼出しを使用してメンバーの任意の数のユーザー定義属性を設定できます。各属性は、メンバー名と同じ表記規則に従った一意の文字列として定義されます。
- ユーザー属性は、メンバー名、別名、世代名またはレベル名と同じであっても構いません。
- 属性リストを解放するには、**EssFree()**を呼び出します。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_HMEMBER_T    hMember;
ESS_USHORT_T    Count, ind;
ESS_PMBRNAME_T   AttributeList = ESS_NULL;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
```

```
Object.AppName  = szAppName;
Object.DbName   = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/************ Get User Attributes ***********/

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMember);
}

if (!sts && hMember)
{
 sts = EssOtlGetUserAttributes(hOutline,
    hMember, &Count, &AttributeList);
}

if (!sts && AttributeList)
{
 printf("User Attribute:\n");
    for(ind = 0; ind < Count; ind++)
 {
 printf("%s\n",AttributeList[ind]);
 }
 EssFree(hInst, AttributeList);
}
```

関連トピック

- EssOtlDeleteUserAttribute
- EssOtlSetUserAttribute

EssOtlImportExportObject

bImport が TRUE か FALSE かに応じて、入力オブジェクトのコンテンツを入力ファイルにインポートまたはエクスポートします。

構文

```
ESS_FUNC_M EssOtlImportExportObject(
hOutline, objHandle, FileName, bImport
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objHandle	ESS_HOBJECT_T	インポートまたはエクスポートされるオブジェクト・ハンドル

パラメータ	データ型	説明
FileName	ESS_STR_T	オブジェクトをエクスポートまたはインポートする必要があるファイル名
bImport	ESS_BOOL_T	• true インポート • false エクスポート

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestImportExportObject()
{
    ESS_STS_T                sts = ESS_STS_NOERR;
    ESS_OBJDEF_T            Object;
    ESS_HOUTLINE_T        hOutline = ESS_NULL;
    ESS_HOBJECT_T        hObjHandle = ESS_NULL;
    ESS_PHOBJECT_T        hObjHandles;
    ESS_STR_T               sFileName;
    ESS_BOOL_T            bImport;
    ESS_OBJECT_TYPES    objType;
    ESS_STR_T                objName = "";
    ESS_ULONG_T            Count, i;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =         hCtx;
    Object.ObjType =     ESS_OBJTYPE_OUTLINE;
    Object.AppName =     szAppName;
    Object.DbName =     szDbName;
    Object.FileName = szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Create an object for the test */
    objType = OBJECT_SMARTLIST;
    objName = "CSRatings";

    sts = EssOtlCreateObject(hOutline, objType,
                    objName, &hObjHandle);

    /* Import a SmartList */
    sFileName = "F:\\testArea\\Smartlist\\ImpCSRatingsSL.txt";
    bImport = ESS_TRUE;
     sts =
    EssOtlImportExportObject(hOutline, hObjHandle,
                    sFileName, bImport);
```

```
    printf("EssOtlImportExportObject sts: %ld\n",sts);

    /* Verify import results */
    sts = EssOtlListObjects(hOutline, objType,
                    &Count, &hObjHandles);
    for (i = 0; i < Count; i++)
        DisplaySmartListInfo(hOutline, hObjHandles[i]);

    SaveOutline(hOutline);

    printf("\n");
    objName = "CSRatings";
    sts = EssOtlFindObject(hOutline, objType,
                objName, &hObjHandle);
    printf("EssOtlFindObject sts: %ld\n",sts);

    /* Export a SmartList */
    bImport = ESS_FALSE;
    sFileName = "F:\\testArea\\Smartlist\\ExpCSRatingsSL.txt";
    sts =
    EssOtlImportExportObject(hOutline, hObjHandle,
                sFileName, bImport);

    /* Unlock objects */
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);

    /* Close */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlIsMemberNameNonUnique

メンバー名が重複しているかどうかを判定します。

構文

```
ESS_FUNC_M EssOtlIsMemberNameNonUnique (
hOutline, hMember, fNameNonUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	非一意性をクエリーするメンバー(入力)。
*fNameNonUnique	ESS_BOOL_T	TRUE の場合は、クエリーされたメンバーが重複メンバー名です(出力)。

備考

- この関数を呼び出す前に、EssOtlOpenOutline を呼び出して編集モードでアウトラインを開いてください。
- この関数の 2 番目の引数のメンバー・ハンドルを取得するには、メンバー走査関数を使用します。
- この関数はメンバー名が重複しているかどうかを確認します。メンバー名が重複している場合には、使用プログラム内で別の関数が固有でない名前を使用して重複しているメンバー名を参照したときに、予期しない動作が起こる可能性があるため、メンバーの完全修飾名(そのメンバーの一意名またはキー)の取得が必要な場合もあります。
- ただし、すべての名前が一意である場合には、完全修飾の名前やキーを使用するためのリソースを消費する必要はありません。
- この関数を使用してからメンバー名が重複していることが判明した場合には、EssOtlGetMemberUniqueName を使用して、完全修飾名を取得してその名前を指定場所に保存できます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_ISUniqMemberName()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hMemberParent, hMemberChild;
ESS_BOOL_T   pbNameUnique;
```

```
memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_FALSE,
 ESS_TRUE, &hOutline);


if (!sts)
{
 sts = EssOtlFindMember(hOutline, "2004", &hMemberParent);
}

if (!sts && hMemberParent)
{
 sts = EssOtlGetChild(hOutline, hMemberParent, &hMemberChild);
}


if (!sts)
{
 //Check whether Qtr1 is unique member name, returns 0 if unique and 1 if non-unique
  sts =  EssOtlIsMemberNameNonUnique (hOutline, hMemberChild, &pbNameUnique);
  if (sts)
   printf("EssOtlIsMemberNameNonUnique failed sts %ld\n",sts);
}


return sts;
}
```

関連トピック

- EssOtlIsMemberNameUniqueWithinDim
- EssOtlIsMemberNameUniqueWithinDimAtGenLevel

EssOtlIsMemberNameUniqueWithinDim

次元内でメンバー名がすべて一意であるかどうかを判定します。

構文

```
ESS_FUNC_M EssOtlIsMemberNameUniqueWithinDim (
hOutline, hDim, *pbNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDim	ESS_HMEMBER_T	入力の次元。これは、EssOtlQueryGetFirstDimension()または EssOtlQueryGetNextDimension()によって戻されます。
*pbNameUnique	ESS_BOOL_T	クエリーされた次元に重複するメンバー名が含まれていない場合には TRUE が戻され、それ以外の場合は FALSE が戻されます。

備考

- この関数は、メンバー名の一意性または非一意性のクエリーを行う次の 3 つの関数のいずれかです。
  - EssOtlIsMemberNameNonUnique は、アウトライン内でメンバー名が重複しているかどうかを判定します。
  - EssOtlIsMemberNameUniqueWithinDim は、次元内でメンバー名がすべて一意であるかどうかを判定します。
  - EssOtlIsMemberNameUniqueWithinDimAtGenLevel は、指定した世代またはレベルにある次元内ですべてのメンバー名が一意であるかどうかを判定します。
- この関数を呼び出す前に、EssOtlOpenOutlineQuery()を呼び出し、クエリー・モードでアウトラインを開いてください。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_ISUniq()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hDim = ESS_NULL;
ESS_BOOL_T   pbNameUnique = 0;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);
```

```
if (!sts)
{
 sts = EssOtlQueryGetFirstDimension(hOutline, &hDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
}



if (!sts)
{
 sts =
    EssOtlIsMemberNameUniqueWithinDim
     (hOutline, hDim, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDim failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}


return sts;
}
```

関連トピック

- EssOtlGetCountOfDupMemberNameInDim
- EssOtlIsMemberNameNonUnique
- EssOtlIsMemberNameUniqueWithinDimAtGenLevel

EssOtlIsMemberNameUniqueWithinDimAtGenLevel

指定した世代またはレベルにある次元内ですべてのメンバー名が一意であるかどうかを判定します。

構文

```
ESS_FUNC_M EssOtlIsMemberNameUniqueWithinDimAtGenLevel (
hOutline, hDim, bGen, usGenLevel, *pbNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hDim	ESS_HMEMBER_T	入力の次元。これは、EssOtlQueryGetFirstDimension()またはEssOtlQueryGetNextDimension()によって戻されます。
bGen	ESS_BOOL_T	入力。TRUE の場合は、usGenLevel が世代番号とみなされます。FALSE の場合は、usGenLevel はレベル番号とみなされます。

パラメータ	データ型	説明
usGenLevel	ESS_USHORT_T	入力された世代またはレベルの番号。
*pbNameUnique	ESS_BOOL_T	出力。クエリーされた次元に、指定した世代またはレベルで重複するメンバー名が含まれている場合には TRUE が戻され、それ以外の場合は FALSE が戻されます。

備考

- この関数は、メンバー名の一意性または非一意性のクエリーを行う次の 3 つの関数のいずれかです。
  - EssOtlIsMemberNameNonUnique は、アウトライン内でメンバー名が重複しているかどうかを判定します。
  - EssOtlIsMemberNameUniqueWithinDim は、次元内でメンバー名がすべて一意であるかどうかを判定します。
  - EssOtlIsMemberNameUniqueWithinDimAtGenLevel は、指定した世代またはレベルにある次元内ですべてのメンバー名が一意であるかどうかを判定します。
- この関数を呼び出す前に、EssOtlOpenOutlineQuery()を呼び出し、クエリー・モードでアウトラインを開いてください。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_ISUniqMemberNameWithinDimatGenLev()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hDim,hNextDim;
ESS_BOOL_T   pbNameUnique, bGen = ESS_TRUE;
ESS_USHORT_T  usGenLevel = 3;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);
```

```
if (!sts)
{
 sts = EssOtlQueryGetFirstDimension(hOutline, &hDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
}


if (!sts)
{
 sts =  EssOtlIsMemberNameUniqueWithinDimAtGenLevel (hOutline, hDim, bGen,
usGenLevel, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDimAtGenLevel failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}

  if (!sts)
 {
 sts = EssOtlQueryGetNextDimension (hOutline, hDim, &hNextDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
  }


if (!sts)
{
 sts =  EssOtlIsMemberNameUniqueWithinDimAtGenLevel (hOutline, hNextDim, bGen,
usGenLevel, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDimAtGenLevel failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}



return sts;
}
```

関連トピック

- EssOtlGetCountOfDupMemberNameInDim
- EssOtlIsMemberNameNonUnique
- EssOtlIsMemberNameUniqueWithinDim

EssOtlListObjects

指定されたタイプのすべてのオブジェクト・ハンドルの配列が戻されます。

構文

```
ESS_FUNC_M EssOtlListObjects(
hOutline, objType, pCount, pObjHandles
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
objType	ESS_OBJECT_TYPES	次のいずれかの値を持つオブジェクト・タイプ: ● OBJECT_SMARTLIST オブジェクト・タイプはテキスト・リスト(スマートリスト)
pCount	ESS_ULONG_T*	オブジェクト・ハンドルのカウント
pObjHandles	ESS_PPHOBJECT_T	オブジェクト・ハンドルの配列が戻されます。EssFree を使用して割当てを解除する必要があります。

戻り値

戻り値:

- 0 - 正常終了の場合

  pCount および pObjHandles には値が含まれます。

- エラー番号 - 失敗した場合

  pCount および pObjHandles は NULL です。

例

```
    void TestCreateObject()
{
  ESS_STS_T      sts = ESS_STS_NOERR;
  ESS_HOUTLINE_T   hOutline = ESS_NULL;
  ESS_OBJDEF_T    Object;
  ESS_OBJECT_TYPES  objType;
  ESS_STR_T      smartListName;
  ESS_HOBJECT_T    ObjHandle;
  ESS_ULONG_T     Count, i;
  ESS_PHOBJECT_T   ObjHandles;
  ESS_HOBJECT_T    hObjHandle;
  ESS_HSMARTLIST_T  hSmartList;
  ESS_STR_T      objName;

  memset(&Object, '\0', sizeof(Object));
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  Object.AppName = szAppName;
  Object.DbName = szDbName;
  Object.FileName = szFileName;

  /* Open outline */
  sts = EssOtlOpenOutline(hCtx, &Object,
```

```
                                ESS_TRUE, ESS_TRUE, &hOutline);

  /* Create a static SmartList */
  objType = OBJECT_SMARTLIST;
  smartListName = "SList1";
  sts = EssOtlCreateObject(hOutline, objType,
                                smartListName, &ObjHandle);
  /* List all SmartList objects */
  objType = OBJECT_SMARTLIST;
  sts =
    EssOtlListObjects(hOutline, objType,
                                &Count, &ObjHandles);


  /* Save */
  SaveOutline(hOutline);

  /* Find objects */
  objName = "SList1";
  sts = EssOtlFindObject(hOutline, objType, objName,
                                            &hObjHandle);

   /* Delete objects */
  hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
  sts = EssOtlDeleteObject(hOutline, hSmartList);
  SaveOutline(hOutline);

  if(ObjHandles)
    EssFree (hInst, ObjHandles);

  /* Unlock objects */
  sts = EssUnlockObject(hCtx, Object.ObjType,
      Object.AppName, Object.DbName, Object.FileName);

        /* Close outline */
  sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects

- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlMoveMember

メンバーを移動します。

構文

```
ESS_FUNC_M
EssOtlMoveMember
 (
hOutline, hMember, hNewParent, hNewPrevSibling
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	移動するメンバーのハンドル
hNewParent	ESS_HMEMBER_T	新しい親のハンドル。このフィールドは、hNewPrevSibling フィールドが ESS_NULL の場合にのみ使用されます。
hNewPrevSibling	ESS_HMEMBER_T	新しい前の兄弟のハンドル

備考

- 移動したメンバーは、hPrevSibling メンバーの後に挿入されます。このフィールドが ESS_NULL である場合、移動したメンバーは hParent によって指定した親の最初の子になります。
- hParent と hPrevSibling の両方が ESS_NULL の場合、移動されたメンバーはアウトラインの最初の次元になります。
- 型が ESS_ATTRMBRDT_STRING でないゼロレベル(リーフ・ノード)の属性メンバーを移動すると、125 ページの「ESS_ATTRSPECS_T」構造体のアウトラインの定義によって、メンバーのロング名がリセットされます。
- 祖先を移動すると、ゼロレベルの属性メンバーのロング名に影響する場合があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_MOVE

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberJan;
```

```
ESS_HMEMBER_T    hMemberMar;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}

if (!sts && hMemberJan)
{
 sts = EssOtlFindMember(hOutline, "Mar",
    &hMemberMar);
}

if (!sts && hMemberMar)
{
 sts = EssOtlMoveMember(hOutline, hMemberJan,
    ESS_NULL, hMemberMar);
}
```

関連トピック

- EssOtlFindMember
- EssOtlRenameMember
- EssOtlAddMember
- EssOtlDeleteMember

EssOtlNewOutline

ファイルを作成せずにアウトラインを作成します。この呼出しは**EssOtlOpenOutline()**の代替として使用されます。

構文

```
ESS_FUNC_M
EssOtlNewOutline
 (
```

```
    hCtx, pNewInfo, phOutline
    );
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	Essbase コンテキスト・ハンドル。
pNewInfo	775 ページの「ESS_OUTLINEINFO_T」	新規アウトラインを記述する構造体。
phOutline	ESS_PHOUTLINE_T	ESS_HOUTLINE_T 変数へのポインタ。このハンドルは API によって設定され、後続のアウトライン API 関数に渡される必要があります。

備考

- この関数では、メモリーに空のアウトラインが作成されます。
- この呼出しが使用された場合はトランザクションは維持されません。トランザクションの維持の詳細は、**EssOtlOpenOutline()**を参照してください。

戻り値

成功の場合、0 が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_OUTLINEINFO_T  NewInfo;
ESS_HOUTLINE_T   hOutline;

memset(&NewInfo, '\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo,
  &hOutline);
```

関連トピック

- EssOtlOpenOutline
- EssOtlWriteOutline
- EssOtlRestructure
- EssOtlCloseOutline
- EssOtlVerifyOutline

EssOtlOpenOutline

既存のアウトラインを開いて読み取ります。アウトラインに対する操作を行う前に、この関数(または **EssOtlNewOutline()**)を呼び出す必要があります。

構文

```
    ESS_FUNC_M
    EssOtlOpenOutline
```

```
(
hCtx, pObject, fLock, fKeepTrans, phOutline
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	Essbase コンテキスト・ハンドル。
pObject	ESS_POBJDEF_T	アウトライン・オブジェクトを開くように定義している、オブジェクト構造体へのポインタ。
fLock	ESS_BOOL_T	開いたときにアウトラインをロックするかどうかを特定するフラグ。サーバー・アウトラインの場合にのみ有効です。
fKeepTrans	ESS_BOOL_T	トランザクションを保持するかどうかを特定するフラグ。 変更するために既存のアウトラインを開き、データベースを再構成するがデータを保持する場合、このフラグを ESS_TRUE に設定してください。ESS_TRUE に設定すると、アウトラインに行われたアクティビティのログが保存されます。 空のアウトラインから始める場合や、再構築の際にデータを保存しない場合には、このフィールドを ESS_FALSE に設定することをお薦めします。ESS_FALSE に設定すると、ログが保持されないため、時間やメモリーを節約できます。
phOutline	ESS_PHOUTLINE_T	ESS_HOUTLINE_T 変数へのポインタ。このハンドルは API によって設定され、後続のアウトライン API 関数に渡される必要があります。

備考

- Unicode モードのアウトラインの場合、EssOtlOpenOutlineEx を使用します。
- アウトライン・ファイルがサーバー上に存在する場合、この呼出しは、クライアントのアクセスのためにそのファイルをローカルにコピーします。
- 集約ストレージ・データベースのアウトラインについては、EssOtlCloseOutline が呼び出されるまで、EssOtlOpenOutline はアウトラインを開いておきます。集約ストレージ・アウトラインは、(完全にメモリーに読み込まれるかわりに)メモリーにページ・インされるので、アウトラインは開いておきます。その結果、EssOtlCloseOutline が呼び出されるまで、一時ファイルはコンピュータの Temp フォルダに残ります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_FILEOPEN
- OTLAPI_ERR_FILEIO

アクセス

この関数を使用するには、アウトライン・オブジェクトを含む指定されたアプリケーションやデータベースへの、適切なレベルのアクセス権が必要です。アウトライン・オブジェクトをロックするには(ロック・フラグは ESS_TRUE)、アウトラインを含む指定されたアプリケーションまたはデータベースに対して、アプリケー

ション・デザイナまたはデータベース・デザイナの権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);
```

関連トピック

- EssOtlOpenOutlineEx
- EssOtlCloseOutline
- EssOtlGetMemberCommentEx
- EssOtlNewOutline
- EssOtlRestructure
- EssOtlSetMemberCommentEx
- EssOtlVerifyOutline
- EssOtlWriteOutline

EssOtlOpenOutlineEx

既存のアウトラインを開き、読み取り、正しいロケールを特定します。アウトラインに対する操作を実行する前に、この関数(または **EssOtlNewOutline()**)を呼び出す必要があります。

構文

```
ESS_FUNC_M
EssOtlOpenOutlineEx
(
hCtx
,
pObject
,
```

```
fLock
,
fKeepTrans
,
pLocaleDescription
,
phOutline
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	Essbase コンテキスト・ハンドル。
pObject	ESS_POBJDEF_T	アウトライン・オブジェクトを開くように定義している、オブジェクト構造体へのポインタ。
fLock	ESS_BOOL_T	開いたときにアウトラインをロックするかどうかを特定するフラグ。サーバー・アウトラインの場合にのみ有効です。
fKeepTrans	ESS_BOOL_T	トランザクションを保持するかどうかを特定するフラグ。 変更するために既存のアウトラインを開き、データベースを再構成するがデータを保持する場合、このフラグを ESS_TRUE に設定してください。ESS_TRUE に設定すると、アウトラインに行われたアクティビティのログが保存されます。 空のアウトラインから始める場合や、再構築の際にデータを保存しない場合には、このフィールドを ESS_FALSE に設定することをお薦めします。ESS_FALSE に設定すると、ログが保持されないため、時間やメモリーを節約できます。
pLocaleDescription		ロケールを識別するために GlobalC で使用される識別子。 LocaleDescription の構造は、次のとおりです: `[language]_[territory].[codepage]@[sort]` 例: `Japaness_japan.MS932@binary` は、アウトライン・ファイルの言語のロケールを説明します。現在のフォーマットで pLocaleDescription を渡すのは、プログラムの責任です。
phOutline	ESS_PHOUTLINE_T	ESS_HOUTLINE_T 変数へのポインタ。このハンドルは API によって設定され、後続のアウトライン API 関数に渡される必要があります。

備考

- この関数は、EssOtlOpenOutline()と同じように機能しますが、Unicode 固有の LocaleDescription 引数が追加されている点が異なります。
- アウトライン・ファイルがサーバー上に存在する場合、この呼出しは、クライアントのアクセスのためにそのファイルをローカルにコピーします。
- 集約ストレージ・データベースのアウトラインについては、EssOtlCloseOutline が呼び出されるまで、EssOtlOpenOutline はアウトラインを開いておきます。集約ストレージ・アウトラインは、(完全にメモリーに読み込まれるかわりに)メモリーにページ・インされるので、アウトラインは開いておきます。その

結果、EssOtlCloseOutline が呼び出されるまで、一時ファイルはコンピュータの Temp フォルダに残ります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_FILEOPEN
- OTLAPI_ERR_FILEIO

アクセス

この関数を使用するには、アウトライン・オブジェクトを含む指定されたアプリケーションやデータベースへの、適切なレベルのアクセス権が必要です。アウトライン・オブジェクトをロックするには(ロック・フラグは ESS_TRUE)、アウトラインを含む指定されたアプリケーションまたはデータベースに対して、アプリケーション・デザイナまたはデータベース・デザイナの権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)を持っている必要があります。

関連トピック

- EssOtlCloseOutline
- EssOtlGetMemberCommentEx
- EssOtlNewOutline
- EssOtlRestructure
- EssOtlSetMemberCommentEx
- EssOtlVerifyOutline
- EssOtlWriteOutlineEx

EssOtlOpenOutlineQuery

既存のアウトラインを開きます。

構文

```
ESS_FUNC_M
EssOtlOpenOutlineQuery
 (
hCtx, pObject, phOutline
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	アウトラインのコンテキスト・ハンドル。有効なサーバー・ログイン・コンテキストである必要があります。
pObject	ESS_POBJDEF_T	開く対象のアウトライン・オブジェクトを定義しているオブジェクト構造体を指すポインタ。現在これは無視されています。アクセス先のデータベースに対して **EssSetActive()** を呼び出す必要があります。

パラメータ	データ型	説明
phOutline	ESS_PHOUTLINE_T	ESS_HOUTLINE_T 変数へのポインタ。API によって設定され、以降の API 関数に渡されます。

備考

- **EssOtlQueryMembers()**を使用したアウトラインにアクセスする場合は、この関数を使用します。
- この関数を呼び出してもアウトラインはダウンロードされず、ファイル全体がメモリーにロードされます。
- したがって、多くのアウトライン API 関数は、この関数呼出しから戻される hOutline を処理できません。
- この呼出しの後で次の呼出しにアクセスできます。他のすべてのアウトライン API 呼出しはエラーを戻します。
  - EssOtlCloseOutline
  - EssOtlGetMemberAlias
  - EssOtlGetMemberFormula
  - EssOtlGetMemberInfo
  - EssOtlGetNextAliasCombination
  - EssOtlGetOutlineInfo
  - EssOtlGetUserAttributes
  - EssOtlGetGenName
  - EssOtlGetGenNames
  - EssOtlGetLevelName
  - EssOtlGetLevelNames
  - EssOtlGetMemberLastFormula

戻り値

関数が正常終了した場合、戻り値は 0 になります。

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_FILEOPEN
- OTLAPI_ERR_FILEIO

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_ACCESS_T    Access;
ESS_STR_T     AppName;
ESS_STR_T     DbName;
```

```
AppName = "Sample";
DbName = "Basic";

sts = EssSetActive(hCtx, AppName, DbName, &Access);

if ( sts == 0)
{
memset(&Object, '\0', sizeof(Object));
sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
}
```

関連トピック

- EssOtlCloseOutline
- EssOtlOpenOutline
- EssOtlQueryMembers
- EssOtlQueryMembersByName
- EssSetActive

EssOtlPutSmartList

EssOtlCreateObject によって作成されるテキスト・リスト(SmartList)ハンドルのコンテンツを移入します。作成されるオブジェクト・ハンドルは、ESS_HSMARTLIST_T ハンドルにタイプキャストできます。

確認ルール:

- pIDs および ppszText への各エントリは一意である必要があります
- ppszText の文字列は、テキスト・リスト名に指定された名前の検証ルールに従う必要があります。
- ppszText テキスト文字列は、空、#OUTOFRANGE、または pszMissingName あるいは pszOutofRangeName と同じであってはなりません
- len のエントリ数は 1024 以下である必要があります。

構文

```
     ESS_FUNC_M EssOtlPutSmartList(hOutline, hSmartList, len, *pIDs, *ppszText,
pszMissingName, pszOutofRangeName);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	テキスト・リストのソース Essbase アウトライン。
hSmartList	ESS_HSMARTLIST_T	テキスト・リスト・ハンドル
len	ESS_UINT16	アイテム数
*pIDs	ESS_UINT32_T	整数 ID
*ppszText	ESS_STR_T	一覧表示されるテキスト

パラメータ	データ型	説明
pszMissingName	ESS_STR_T	不明なスマート・テキスト名
pszOutofRangeName	ESS_STR_T	範囲外のスマート・テキストの名前。

戻り値

戻り値:

- 0 - 正常終了の場合

```
pIDs
 and
ppszText
 contain values.
```

- エラー番号 - 失敗した場合

```
pIDs
 and
ppszText
 are NULL.
```

例

```
    void TestPutSmartList()
{
    ESS_STS_T                       sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T                hOutline = ESS_NULL;
    ESS_OBJECT_TYPES             objType;
    ESS_HOBJECT_T                 hObjHandle;
    ESS_PHOBJECT_T                hObjHandles;
    ESS_PSMARTLISTINFO_T    SmartListInfo = ESS_NULL;
    ESS_OBJDEF_T                    Object;
    ESS_HSMARTLIST_T             hSmartList;
    ESS_USHORT_T                    len;
    ESS_SMARTLISTID_T             pIds[4];
    ESS_STR_T                       ppszText[4];
    ESS_STR_T                       pszMissingName;
    ESS_STR_T                       pszOutOfRangeName;
    ESS_ULONG_T                   Count, i;
    ESS_STR_T                       smartListNames[3] =
            { "MainColors", "TempColors1", "TempColors2" };

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =                  hCtx;
    Object.ObjType =              ESS_OBJTYPE_OUTLINE;
    Object.AppName =              szAppName;
    Object.DbName =             szDbName;
    Object.FileName =         szFileName;

    /* Open outline */
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
                ESS_TRUE, &hOutline);
```

```
    /* Create a SmartList */
    objType = OBJECT_SMARTLIST;
    sts = EssOtlCreateObject(hOutline, objType,
                smartListNames[0], &hObjHandle);

    /* Set up and put SmartList */
    hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
    len = 4;
    pIds[0] = 1;
    pIds[1] = 2;
    pIds[2] = 3;
    pIds[3] = -1;
    ppszText[0] = "Red";
    ppszText[1] = "Green";
    ppszText[2] = "Blue";
    ppszText[3] = "Yellow";
    pszMissingName = "Missing";
    pszOutOfRangeName = "OutOfRange";
    sts =
    EssOtlPutSmartList(hOutline, hSmartList,
                    len, pIds, ppszText, pszMissingName,
                    pszOutOfRangeName);


    SaveOutline(hOutline);

    /* Clean up */
    for(i = 0; i <= 12; i++)
    {
        sts = EssOtlFindObject(hOutline, objType,
                    smartListNames[i], &hObjHandle);
        hSmartList = (ESS_HSMARTLIST_T)hObjHandle;
        sts = EssOtlDeleteObject(hOutline, hSmartList);
    }

    SaveOutline(hOutline);

    objType = OBJECT_SMARTLIST;
    sts = EssOtlListObjects(hOutline, objType,
                &Count, &hObjHandles);
    for (i = 0; i < Count; i++)
        DisplaySmartListInfo(hOutline, hObjHandles[i]);
    if(hObjHandles)
        EssFree (hInst, hObjHandles);

    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
}
```

EssOtlQueryAttributes

指定した属性メンバーまたは属性次元についてのメンバー情報にクエリーを行います。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
pAttributeQuery;	765 ページの「ESS_ATTRIBUTEQUERY_T」	クエリーを定義する構造体
pCount;	ESS_PULONG_T	戻されたメンバーのハンドルの数
pphMemberArray;	ESS_PPHMEMBER_T	戻されたメンバーのハンドルの配列へのポインタ

備考

この関数を呼び出す前に、EssOtlOpenOutlineQuery を呼び出し、クエリー・モードでアウトラインを開いてください。

例

```
    void ESS_OtlQueryAttributes()
{
 ESS_STS_T       sts = ESS_STS_NOERR;
 ESS_HMEMBER_T     hMember;
 ESS_OBJDEF_T     Object;
 ESS_APPNAME_T     szAppName;
 ESS_DBNAME_T     szDbName;
 ESS_OBJNAME_T     szFileName;
 ESS_SHORT_T      hOutlineQuery;
 ESS_ATTRIBUTEQUERY_T  pAttributeQuery;
 ESS_ULONG_T      Count = 0;
 ESS_PHMEMBER_T    phMemberArray = ESS_NULL;
 ESS_PMBRINFO_T    pMbrInfo = ESS_NULL;
 int          index;


 memset(&pAttributeQuery, 0x00, sizeof(ESS_ATTRIBUTEQUERY_T));
 memset(&Object, '\0', sizeof(Object));
 Object.hCtx = hCtx;
 Object.ObjType = ESS_OBJTYPE_OUTLINE;
 strcpy(szAppName, "Sample");
 strcpy(szDbName, "Basic");
 strcpy(szFileName, "Basic");
 Object.AppName = szAppName;
 Object.DbName = szDbName;
 Object.FileName = szFileName;

 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutlineQuery);
 printf("EssOtlOpenOutlineQuery() sts: %ld\n",sts);
 pAttributeQuery.bInputMemberIsHandle == ESS_FALSE;
 pAttributeQuery.uInputMember.szMember = "100-10";
 pAttributeQuery.usInputMemberType = ESS_BASE_MEMBER;
```

```
 pAttributeQuery.usOutputMemberType = ESS_ATTRIBUTE_MEMBER;
 pAttributeQuery.usOperation = ESS_ALL;
 pAttributeQuery.Attribute.usDataType = ESS_ATTRMBRDT_NONE;

 sts = EssOtlQueryAttributes(hOutlineQuery, &pAttributeQuery, &Count,
&phMemberArray);
 printf("EssOtlQueryAttributes() sts: %ld\n",sts);

 if (!sts && phMemberArray)
 {
  printf("\n------- Query Results -------\n");
  for (index = 0; index < Count; index++)
  {
   sts = EssOtlGetMemberInfo(hOutlineQuery,phMemberArray[index],&pMbrInfo);
   printf("\t%s\n",pMbrInfo->szMember);
  }

  if (Count && phMemberArray)
  {
   sts = EssOtlFreeMembers(hOutlineQuery,Count, phMemberArray);
  }
 }
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlSetAttributeSpecifications

EssOtlQueryAttributesEx

指定した属性メンバーまたは属性次元についてのメンバー情報にクエリーを行います。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル

パラメータ	データ型	説明
pAttributeQuery;	765 ページの「ESS_ATTRIBUTEQUERY_T」	クエリーを定義する構造体
pCount;	ESS_PMBRCOUNTS_T	戻されたメンバーのハンドルの数
pphMemberArray;	ESS_PPHMEMBER_T	戻されたメンバーのハンドルの配列へのポインタ

備考

この関数を呼び出す前に、EssOtlOpenOutlineQuery を呼び出し、クエリー・モードでアウトラインを開いてください。

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlQueryAttributes
- EssOtlOpenOutlineQuery
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlSetAttributeSpecifications

EssOtlQueryGenerationInfo

EssOtlQueryGenerationInfo()は、次元の最上位メンバーのコメント・フィールドに含まれている時間次元生成情報のクエリーを行います。この情報を一度取得すると、**EssOtlGetLinkedAttributeAttachLevel()** とともに使用して前期比の分析を実行できます。

構文

```
ESS_FUNC_M
EssOtlQueryGenerationInfo
 (
hOutline, szName, queryID, ppReturns
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル。これは EssOtlOpenOutlineQuery()から戻されている必要があります。
szName	ESS_STS_T	日時次元の最上位メンバーの名前
queryID	ESS_ULONG_T	クエリー識別子定数 ESS_OTLQRYTIDIM_TIMEPERIODS を使用します
ppReturns	ESS_PPVOID_T	この次元のクエリー情報構造体へのポインタ。

備考

EssOtlQueryGenerationInfo()の呼出し元は、構造体 ID が ESS_DT_STRUCT_TIGENINFO の **EssOtlFreeStructure()**を呼び出し、戻される構造体ポインタ用に用意されているメモリーを解放します。

戻り値

正常に終了した場合は、ESS_PTIMEDIM_GENINFO_T 構造体へのポインタが戻されます。

例

```
SS_STR_T strBuf1 = "Year";
ESS_ULONG_T  queryId = ESS_OTLQRYTIDIM_TIMEPERIODS;
ESS_PVOID_T pReturns;
ESS_PTIMEDIM_GENINFO_T tpStruc = NULL;


sts = EssOtlQueryGenerationInfo (hOutline, /*query outline handle*/
          strBuf1, /* IN - date-time dimension member name*/
          queryId, /* IN - query ID */
&pReturns);

if (sts)
goto exit;


switch (queryId)
  {
case ESS_OTLQRYTIDIM_TIMEPERIODS:
tpStruc = (ESS_PTIMEDIM_GENINFO_T)pReturns;


for (ii = 0; ii < tpStruc->usCount; ii++)
fprintf(cmdctxp->output, "Time period for Gen %d = %s\n", ii+1, TimePeriodNames[tpStruc->ptps[ii]]);


sts = EssOtlFreeStructure (cmdctxp->hOutline[hOutlineChoice], ESS_DT_STRUCT_TIGENINFO,
1, pReturns);
if (sts)
```

```
goto exit;
break;


default:
break;
  }
```

関連トピック

- EssOtlGetLinkedAttributeAttachLevel
- EssOtlFreeStructure

EssOtlQueryGetFirstDimension

アウトラインの最初の次元の次元ハンドルを戻します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル(入力)。
phDim;	ESS_PHMEMBER_T	次元ハンドル(出力)。

備考

- この関数を呼び出す前に、EssOtlOpenOutlineQuery を呼び出し、クエリー・モードでアウトラインを開いてください。
- この関数は、アウトラインの最初の次元の次元ハンドルを戻します。この関数によって戻されるハンドルは次に、EssOtlGetDimensionNameUniqueness、EssOtlGetCountOfDupMemberNameInDim、または EssOtlIsMemberNameUniqueWithinDim の呼出しに使用されます。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_ISUniq()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hDim = ESS_NULL;
ESS_BOOL_T   pbNameUnique = 0;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
```

```
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);



if (!sts)
{
 sts =
    EssOtlQueryGetFirstDimension
    (hOutline, &hDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
}



if (!sts)
{
 sts =  EssOtlIsMemberNameUniqueWithinDim (hOutline, hDim, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDim failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}


return sts;
}
```

関連トピック

- EssOtlQueryGetNextDimension
- EssOtlIsMemberNameUniqueWithinDim

EssOtlQueryGetNextDimension

クエリー・モードで開かれたアウトラインの次元の次の次元ハンドルを戻します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインへのハンドル(入力)
hDim;	ESS_HMEMBER_T	次元ハンドル(入力)

パラメータ	データ型	説明
phNextDim	ESS_PHMEMBER_T	次の次元のハンドル(出力)

備考

- この関数を呼び出す前に、EssOtlOpenOutlineQuery を呼び出し、クエリー・モードでアウトラインを開いてください。
- 例に示すように、この関数を呼び出す前に EssOtlQueryGetFirstDimension を呼び出す必要があります。そうしないと、エラーが戻されます。
- 次元の最後にある次元ハンドルを渡すと、この関数は NULL を戻します。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_ISUniqMemberNameWithinDimatGenLev()
{
ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hDim,hNextDim;
ESS_BOOL_T   pbNameUnique, bGen = ESS_TRUE;
ESS_USHORT_T  usGenLevel = 3;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutlineQuery (hCtx, &Object, &hOutline);


if (!sts)
{
 sts = EssOtlQueryGetFirstDimension(hOutline, &hDim);

 if (sts)
  printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
}


if (!sts)
{
```

```
 sts =  EssOtlIsMemberNameUniqueWithinDimAtGenLevel (hOutline, hDim, bGen,
usGenLevel, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDimAtGenLevel failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}

  if (!sts)
{
sts =
   EssOtlQueryGetNextDimension
    (hOutline, hDim, &hNextDim);

if (sts)
 printf("EssOtlQueryGetFirstDimension failed sts %ld\n",sts);
 }


if (!sts)
{
 sts =  EssOtlIsMemberNameUniqueWithinDimAtGenLevel (hOutline, hNextDim, bGen,
usGenLevel, &pbNameUnique);
 if (sts)
  printf("EssOtlIsMemberNameUniqueWithinDimAtGenLevel failed sts %ld\n",sts);
 else
  printf("pbNameUnique is %d\n", pbNameUnique);
}



return sts;
}
```

関連トピック

- EssOtlQueryGetFirstDimension

EssOtlQueryMembers

アウトラインをクエリーします。

構文

```
ESS_FUNC_M EssOtlQueryMembers (
hOutline, hMember, pPredicate, pMbrCounts, phMemberArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。これは EssOtlOpenOutlineQuery から戻されている必要があります。

パラメータ	データ型	説明
hMember	ESS_HMEMBER_T	操作が実行されるメンバーのハンドル。この値が NULL である場合、その次元の論理親を表し、アウトラインの最上部であるとみなされます。 ハンドルが共有メンバーである場合、この関数は、基となる保管済メンバーで実行します。 次のオプションでは、この値は無視されます: ● ESS_NAMEDGENERATION ● ESS_NAMEDLEVEL ● ESS_USERATTRIBUTE ● ESS_SEARCH ● ESS_WILDSEARCH
pPredicate	778 ページの「ESS_PREDICATE_T」	クエリーを定義している構造体。この構造体のフィールドは、「注意」を参照してください。
pMbrCounts	768 ページの「ESS_MBRCOUNTS_T」	カウントに関する情報を定義している構造体。次のフィールドが含まれます: ● ulStart- 戻す最初の番号 ● ulMaxCount- 戻されるメンバーのハンドルの最大数 ● ulTotalCount- クエリーの実行結果において定義されるメンバーの合計数 ● pulReturnCount- このクエリーにおいて戻されるメンバーのハンドルの数
phMemberArray	ESS_PPHMEMBER_T	クエリーから戻されたメンバーのハンドルの配列。

備考

- この呼出しは、操作対象のメンバーのハンドルを使用して、オプション値で指定された基準に適合するメンバーのハンドル配列を戻します。
- 戻された phMembers のメンバー配列が必要ではなくなった場合、呼出し元は、EssOtlFreeMembers を呼び出す必要があります。
- 配列の各 hMember 要素は、EssOtlOpenOutlineQuery にリストされた呼出しでのみ使用できます。たとえば、戻されたメンバー・ハンドルは、EssOtlGetSibling を呼び出すためには使用できません。
- pPredicate 構造体のフィールドは、次のように使用されます:
  - ulQuery- 実行するべき操作を定義する値。次のいずれかになります:
    - ESS_CHILDREN
    - ESS_DESCENDANTS
    - ESS_BOTTOMLEVEL
    - ESS_SIBLINGS
    - ESS_SAMELEVEL
    - ESS_SAMEGENERATION
    - ESS_PARENT
    - ESS_DIMENSION

- ESS_NAMEDGENERATION
- ESS_NAMEDLEVEL
- ESS_SEARCH
- ESS_WILDSEARCH
- ESS_USERATTRIBUTE
- ESS_ANCESTORS
- ESS_DTSMEMBERS

- ulOptions - 検索オプションを定義する値。有効な値:
  - ESS_COUNTONLY - メンバー・ハンドルは戻さずに pCounts 構造体の pTotalCount フィールドに値を入れます
  - ESS_NOTOTALCOUNTS
  - ESS_INCLUDEHYBRIDANALYSIS
  - ESS_EXCLUDEHYBRIDANALYSIS
  - ESS_FORCECASESENSITIVE
  - ESS_FORCEIGNORECASE

  Query タイプが ESS_SEARCH または ESS_WILDSEARCH に設定されると、Option ではさらに 3 つの値が有効になります:
  - ESS_MEMBERSONLY
  - ESS_ALIASESONLY
  - ESS_MEMBERSANDALIASES

  複数の値を指定するためには、ビット OR (|)を使用します。例:

  ```
  ESS_FORCECASESENSITIVE | ESS_MEMBERSONLY
  ```

- szDimension- クエリーの範囲を制限する次元。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESS_NAMEDGENERATION
  - ESS_NAMEDLEVEL
  - ESS_USERATTRIBUTE
  - ESS_SEARCH - 全次元を検索するには、NULL に設定します
  - ESS_WILDSEARCH - 全次元を検索するには、NULL に設定します
- pszString1-

  オプションによって決まる入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESS_NAMEDGENERATION - 世代の名前
  - ESS_NAMEDLEVEL - レベルの名前

- ESS_SEARCH - 検索する文字列。この文字列は完全一致として定義されています。
- ESS_WILDSEARCH - 検索する文字列。この文字列は、末尾にオプションの'*'が付いた完全一致検索文字列として定義されます。'*'は 1 文字以上の任意の文字を意味します。
- ESS_USERATTRIBUTE - ユーザー定義属性。

- pszString2- オプションによって特定される入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESS_USERATTRIBUTE - ユーザー定義属性。
  - ESS_SEARCH、ESS_WILDSEARCH - オプションで別名テーブルを検索するように設定されている場合、この文字列で検索対象の別名テーブルを指定します。NULL の場合、すべての別名テーブルが検索されます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_HMEMBER_T   hMember = 0;
ESS_PREDICATE_T  Predicate;
ESS_MBRCOUNTS_T  Counts;
ESS_PHMEMBER_T   phMemberArray = ESS_NULL;
ESS_ULONG_T    i;
ESS_ACCESS_T    Access;
ESS_STR_T     AppName;
ESS_STR_T     DbName;

AppName = "Sample";
DbName = "Basic";

sts = EssSetActive(hCtx, AppName, DbName, &Access);

if ( sts == 0)
{
 memset(&Object, '\0', sizeof(Object));

 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);

 memset(&Predicate, '\0', sizeof(Predicate));
 Predicate.ulQuery      = ESS_CHILDREN;
 Predicate.pszDimension  = "Year";

 memset(&Counts, '\0', sizeof(Counts));
 Counts.ulStart     = 0;
 Counts.ulMaxCount = 10;
```

```
 if(!sts)
 {
  sts = EssOtlQueryMembers(hOutline, hMember,
     &Predicate, &Counts, &phMemberArray);

  if (!sts && Counts.ulReturnCount)
  {
   sts = EssOtlFreeMembers(hOutline,
      Counts.ulReturnCount, phMemberArray);
  }
 }
}
```

関連トピック

- EssOtlFreeMembers
- EssOtlGetDimensionUserAttributes
- EssOtlOpenOutlineQuery
- EssOtlQueryMembersByName

EssOtlQueryMembersByName

アウトラインをクエリーします。

構文

```
ESS_FUNC_M
EssOtlQueryMembersByName
(
hOutline
,
pszMember
,
pPredicate
,

pMbrCounts
,
phMemberArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。これは EssOtlOpenOutlineQuery()から戻されている必要があります。

パラメータ	データ型	説明
pszMember	ESS_STR_T	操作を行うメンバーのメンバー名文字列。この値が NULL である場合、その次元の論理親を表し、アウトラインの最上部であるとみなされます。次のオプションでは、この値は無視されます: ● ESS_NAMEDGENERATION ● ESS_NAMEDLEVEL ● ESS_USERATTRIBUTE ● ESS_SEARCH ● ESS_WILDSEARCH
pPredicate	778 ページの「ESS_PREDICATE_T」	クエリーを定義している構造体。この構造体のフィールドは、「注意」を参照してください。
pMbrCounts	768 ページの「ESS_MBRCOUNTS_T」	メンバー・カウントに関する情報を定義している構造体。次のフィールドが含まれます: ● **ulStart**- 戻す最初の番号 ● **ulMaxCount**- 戻されるメンバーのハンドルの最大数。 ● **ulTotalCount**- クエリーの実行結果において定義されるメンバーの合計数。 ● **pulReturnCount**- このクエリーにおいて戻されるメンバーのハンドルの数。
phMemberArray	ESS_PPHMEMBER_T	クエリーから戻されたメンバーのハンドルの配列。

備考

- この呼出しは、操作対象のメンバー名文字列を使用して、オプション値で指定された基準に適合するメンバーのハンドル配列を戻します。
- 戻された phMembers のメンバー配列が不要になった場合、呼出し元は、**EssOtlFreeMembers()**を呼び出す必要があります。
- 配列の各 hMember 要素は、**EssOtlOpenOutlineQuery()**にリストされた呼出しのみで使用できます。たとえば、戻されたメンバー・ハンドルは、**EssOtlGetSibling()**を呼び出すためには使用できません。
- pPredicate 構造体のフィールドは、次のように使用されます:
  - **ulQuery** - 実行する操作を定義する値。次のいずれかになります:
    - ESS_CHILDREN
    - ESS_DESCENDANTS
    - ESS_BOTTOMLEVEL
    - ESS_SIBLINGS
    - ESS_SAMELEVEL
    - ESS_SAMEGENERATION
    - ESS_PARENT
    - ESS_DIMENSION
    - ESS_NAMEDGENERATION
    - ESS_NAMEDLEVEL

- ESS_SEARCH
- ESS_WILDSEARCH
- ESS_USERATTRIBUTE
- ESS_ANCESTORS
- ESS_DTSMEMBERS

- **ulOptions** - 検索オプションを定義する値。有効な値:
  - ESS_COUNTONLY - メンバー・ハンドルは戻さずに、pCounts 構造体の pTotalCount フィールドにのみ値を入れます
  - ESS_NOTOTALCOUNTS
  - ESS_INCLUDEHYBRIDANALYSIS
  - ESS_EXCLUDEHYBRIDANALYSIS
  - ESS_FORCECASESENSITIVE
  - ESS_FORCEIGNORECASE

  Query タイプが ESS_SEARCH または ESS_WILDSEARCH に設定されると、Option ではさらに 3 つの値が有効になります:
  - ESS_MEMBERSONLY
  - ESS_ALIASESONLY
  - ESS_MEMBERSANDALIASES

  複数の値を指定するためには、ビット OR (|)を使用します。例:

```
ESS_FORCECASESENSITIVE | ESS_MEMBERSONLY
```

- **pszString1**- オプションによって決まる入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESS_NAMEDGENERATION - 世代の名前
  - ESS_NAMEDLEVEL - レベルの名前
  - ESS_SEARCH - 検索する文字列。この文字列は完全一致として定義されています
  - ESS_WILDSEARCH - 検索する文字列。この文字列は、末尾にオプションの'*'が付いた完全一致検索文字列として定義されます。'*'は 1 文字以上の任意の文字を意味します。
  - ESS_USERATTRIBUTE - ユーザー定義属性
- **pszString2**- オプションによって特定される入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESS_USERATTRIBUTE - ユーザー定義属性。
  - ESS_SEARCH、ESS_WILDSEARCH - オプションで別名テーブルを検索するように設定されている場合、この文字列で検索対象の別名テーブ

ルを指定します。NULL の場合、すべての別名テーブルが検索されます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_STR_T     pszMember;
ESS_PREDICATE_T   Predicate;
ESS_MBRCOUNTS_T   Counts;
ESS_PHMEMBER_T   phMemberArray = ESS_NULL;
ESS_ULONG_T    i;
ESS_ACCESS_T    Access;
ESS_STR_T     AppName;
ESS_STR_T     DbName;

pszMember = "Qtr1";
AppName = "Sample";
DbName = "Basic";

sts = EssSetActive(hCtx, AppName, DbName, &Access);

if ( sts == 0)
{
 memset(&Object, '\0', sizeof(Object));

 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);

 memset(&Predicate, '\0', sizeof(Predicate));
 Predicate.ulQuery      = ESS_CHILDREN;
 Predicate.pszDimension  = "Year";

 memset(&Counts, '\0', sizeof(Counts));
 Counts.ulStart     = 0;
 Counts.ulMaxCount = 10;

 if(!sts)
 {
  sts = EssOtlQueryMembersByName(hOutline, pszMember,
     &Predicate, &Counts, &phMemberArray);

  if (!sts && Counts.ulReturnCount)
  {
    sts = EssOtlFreeMembers(hOutline,
       Counts.ulReturnCount, phMemberArray);
  }
 }
}
```

関連トピック

- EssOtlFreeMembers
- EssOtlGetDimensionUserAttributes
- EssOtlOpenOutlineQuery
- EssOtlQueryMembers

EssOtlQueryMembersEx

特定のメンバーおよびメンバー・フィールドのアウトラインをクエリーし、メンバーのハンドルの配列を戻します。戻されたメンバーのハンドルは、EssOtlGetMemberInfo()など、他のアウトライン API 関数で使用できます。(EssOtlGetMemberInfo()は、157 ページの「ESS_MEMBERINFO_T」および 768 ページの「ESS_MBRINFO_T」に含まれる個別のフィールドを取得できます。)

構文

```
ESS_FUNC_M
EssOtlQueryMembersEx
(
hOutline
,
pszFieldSelection
,
pszMemberSelection
,

pMaxCount
,
ppMemberArray
,
ppqryErrorList
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	Essbase アウトライン・ハンドル。これは EssOtlOpenOutlineQuery()から戻されている必要があります。
pszFieldSelection	ESS_STR_T	各メンバーについて戻されるフィールドの集合を定義するクエリー文字列。pszFieldSelection の構文を「注意」に示します。
pszMemberSelection	ESS_STR_T	戻されるメンバーのセットを定義するクエリー文字列。このクエリー文字列の構文は、メンバー選択の構文です。つまり、クエリー文字列は、FIX()ステートメントで使用できるものであれば、何でもかまいません。
pMaxCount	ESS_PULONG_T	入力: 戻されるメンバーのハンドルの最大数へのポインタ。 出力: 戻されるメンバーのハンドルの数へのポインタ。

パラメータ	データ型	説明
ppMemberArray	ESS_PPHMEMBER_T	戻されるメンバーのハンドルの配列の先頭を指すポインタへの参照。
ppqryErrorList	772 ページの「ESS_OTLQUERYERRORLIST_T」	クエリー内のエラー・リストに含まれる構造体を指すポインタへの参照。

備考

- 重複メンバー名を認めるアウトラインでは、この関数は、共有メンバーの完全修飾名を戻します。たとえば、Sample Basic では、共有メンバー 100-20 を含んでいるクエリーは、その完全修飾名[Diet].[100-20]を戻します。
- メンバー・フィールド選択の一部として UniqueName を使用すると、そのフィールド選択の一部として ShareOption が自動的に含まれます。
- EssOtlQueryMemberEx()は、アウトライン・ハンドルを受け取り、pszMemberSelection で指定したメンバー・ハンドルの配列を戻します。
- 戻された pphMembers のメンバー配列が不要になった場合、呼出し元は EssOtlFreeMembers()を呼び出す必要があります。
- 配列の各メンバー・ハンドル要素は、EssOtlOpenOutlineQuery()にリストされた呼出しでのみ使用できます。たとえば、戻されたメンバー・ハンドルは、EssOtlGetSibling()を呼び出すために使用できません。
- pszFieldSelection の構文は次のとおりです:

```
        QueryString ==: <SelectMbrInfo (
        FieldName
        {,
        FieldName
        }, ... )

where
        FieldName
        is one of the following:

MemberName          /* Member name */
MemberLevel          /* Member level number */
MemberGeneration       /* Member generation number */
Cosolidation         /* Whether this member is consolidated */
TwoPass            /* Whether this member undergoes a two pass operation */
Expense            /* Whether this is an expense member */
CurrencyConvType       /* Currency conversion type */
CurrencyMember        /* Whether this is a currency member */
TimeBalance          /* Time balance measure */
SkipOption          /* Whether this member skips the time balance operation */
ShareOption          /* Whether this is a shared member*/
StorageType          /* Dimension's storage type */
DimensionCategory       /* Dimension category: accounts, time, currency, etc. */
DimensionStorageCategory    /* Dimension storage category: time, units, scenario,
etc. */
Comment            /* Member comment */
ChildrenCount         /* Number of children */
MemberNumber         /* Member number */
```

```
DimensionName         /* Dimension name */
DimensionNumber        /* Dimension number */
MemberAliasName        /* Alias for this member */
ParentMemberName       /* Parent's name */
ChildMemberName        /* Child's name */
PreviousMemberName      /* Left sibling's name */
NextMemberName        /* Right sibling's name */
CurrencyConversionDatabase   /* Whether this database has currency conversion */
MemberStatus         /* Member status */
UDAList           /* List of UDAs attached to this member */
MemberFormula         /* Formula for this member */
MemberValidity        /* Whether this member is valid */
Attributes          /* All attribute fields. If the member is not attributed, then
attribute name is set to NULL */
UniqueName          /* If the member is duplicate, its fully qualified, unique
name.  */
```

注： 個別のフィールド名には、先頭の"<"文字はありません。

- この関数を EssOtlGetMemberField()とともに使用する場合は、この関数の pszFieldSelection 文字列に、EssOtlGetMemberField()の MbrFieldID 定数を使用する場合に指定するのと同じフィールドを含めます。このようにしないと、EssOtlGetMemberField()はエラー OTLAPI_ERR_MBRINVALID を戻します。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

次のコード・スニペットは、Market の子または Product の子である各メンバーの名前、集計および式を戻します。戻されたとき、MaxCount は、戻されたメンバーの数を含んでいます。また、phMemberArray は、戻されたメンバーのセットのためのハンドルの配列を含んでいます。さらなるアウトライン API の呼出しは、phMemberArray に戻されたメンバー・ハンドルの配列を使用した、メンバーの問合せを可能にします。

```
    member_fields   = "<SelectMbrInfo ( MemberName, Consolidation, MemberFormula ) ";
member_selection  = "@ichild(Product), @ichild(Market)";
MaxCount     = -1;
phMemberArray   = ESS_NULL;
pqryErrorList   = ESS_NULL;

sts = EssOtlQueryMembersEx(hOutline,
            member_fields,
            member_selection,
            &MaxCount,
            &phMemberArray,
            &pqryErrorList);

if (sts != 0) goto error_exit;
```

EssOtlQueryMembersEx()、**EssOtlGetMemberField()**および**ESS_OTLQUERYERRORLIST_T** を使用し、**EssOtlFreeMembers()**および **EssFree()**の呼出しを含んでいる例については、1070 ページの「拡張メンバーのクエリー・コードの例」を参照してください。

関連トピック

- EssOtlFreeMembers
- EssOtlGetDimensionUserAttributes
- EssOtlGetMemberField
- EssOtlOpenOutlineQuery
- EssOtlQueryMembers
- EssOtlQueryMembersByName

EssOtlQueryMembersExArray

特定のメンバーおよびメンバー・フィールドのアウトラインをクエリーし、メンバーのハンドルの配列を戻します。戻されたメンバーのハンドルは、**EssOtlGetMemberInfo()**など、他のアウトライン API 関数で使用できます。(**EssOtlGetMemberInfo()**は、157 ページの「ESS_MEMBERINFO_T」および768 ページの「ESS_MBRINFO_T」に含まれる個別のフィールドを取得できます。)

構文

```
ESS_FUNC_M
EssOtlQueryMembersExArray
(
hOutline, pszFieldSelection, queryCount, pszMemberSelectionArr, pMaxCountArr,
pphMemberArr, ppqryErrorList
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	**EssOtlOpenOutlineQuery()**から戻されるアウトライン・ハンドル。
pszFieldSelection	ESS_STR_T	クエリーで戻されるメンバー・フィールドを選択します。配列のすべてのクエリーに同じ選択内容が使用されます。
queryCount	ESS_SHORT_T	入力配列のメンバー数。
pszMemberSelectionArr	ESS_STR_T	メンバー選択の queryCount クエリー文字列の配列。このクエリー文字列の構文は、メンバー選択の構文です。つまり、クエリー文字列は、FIX()ステートメントで使用できるものであれば、何でもかまいません。
pMaxCountArr	ESS_PULONG_T	配列内の各クエリーが、最大いくつのメンバーを戻すかを決定する queryCount 値の配列。各値は、実際に戻された数に置き換えられます。
pphMemberArr	ESS_PPHMEMBER_T	戻されたメンバー・ハンドル配列(それぞれ pMaxCountArr[i]値を含む)の queryCount 配列。

パラメータ	データ型	説明
ppqryErrorList	772 ページの「ESS_OTLQUERYERRORLIST_T」	エラーのあるメンバーのリスト。

備考

- 重複メンバー名を認めるアウトラインでは、この関数は、共有メンバーの完全修飾名を戻します。たとえば、Sample Basic では、共有メンバー 100-20 を含んでいるクエリーは、その完全修飾名[Diet].[100-20]を戻します。
- メンバー・フィールド選択の一部として UniqueName を使用すると、そのフィールド選択の一部として ShareOption が自動的に含まれます。
- **EssOtlQueryMemberExArray()**は、アウトライン・ハンドルを受け取り、pszMemberSelection で指定したメンバー・ハンドルの配列を戻します。
- 戻された pphMembers のメンバー配列が不要になった場合、呼出し元は**EssOtlFreeMembers()**を呼び出す必要があります。
- 配列の各メンバー・ハンドル要素は、**EssOtlOpenOutlineQuery()**にリストされた呼出しのみで使用できます。たとえば、戻されたメンバー・ハンドルは、**EssOtlGetSibling()**を呼び出すためには使用できません。
- pszFieldSelection の構文は次のとおりです:

```
        QueryString ==: <SelectMbrInfo (
        FieldName
         {,
        FieldName
        }, ... )

where
        FieldName
         is one of the following:

MemberName          /* Member name */
MemberLevel          /* Member level number */
MemberGeneration       /* Member generation number */
Cosolidation         /* Whether this member is consolidated */
TwoPass           /* Whether this member undergoes a two pass operation */
Expense           /* Whether this is an expense member */
CurrencyConvType       /* Currency conversion type */
CurrencyMember        /* Whether this is a currency member */
TimeBalance          /* Time balance measure */
SkipOption          /* Whether this member skips the time balance operation */
ShareOption          /* Whether this is a shared member*/
StorageType          /* Dimension's storage type */
DimensionCategory       /* Dimension category: accounts, time, currency, etc. */
DimensionStorageCategory    /* Dimension storage category: time, units, scenario,
etc. */
Comment           /* Member comment */
ChildrenCount         /* Number of children */
MemberNumber          /* Member number */
DimensionName         /* Dimension name */
DimensionNumber        /* Dimension number */
MemberAliasName        /* Alias for this member */
```

```
ParentMemberName       /* Parent's name */
ChildMemberName        /* Child's name */
PreviousMemberName       /* Left sibling's name */
NextMemberName        /* Right sibling's name */
CurrencyConversionDatabase   /* Whether this database has currency conversion */
MemberStatus         /* Member status */
UDAList           /* List of UDAs attached to this member */
MemberFormula         /* Formula for this member */
MemberValidity        /* Whether this member is valid */
Attributes         /* All attribute fields. If the member is not attributed, then
attribute name is set to NULL */
UniqueName          /* If the member is duplicate, its fully qualified, unique
name.  */
```

注： 個別のフィールド名には、先頭の"<"文字はありません。

戻り値

正常終了の場合は 0 が戻されます。

例

次のコード・スニペットは、Market の子である各メンバーおよび Product の子である各メンバーの名前、集計および式を 2 つの個別のメンバー配列に戻します。このコード・スニペットは、EssOtlQueryMembersEx で 2 つのクエリーとなる呼出しを 1 回の呼出し EssOtlQueryMembersExArray にまとめます。戻されるメンバー・フィールドは、配列内のすべてのクエリーで同じになり、すべての配列のサイズは queryCount に一致する必要があります。

戻されたとき、MaxCountArray[i]には各クエリーで戻されたメンバーの数が含まれており、phMemberArrayArray[i]には各クエリーで戻されたメンバーのセットに対するハンドル配列が含まれています。さらなるアウトライン API の呼出しによって、phMemberArrayArray[i]に戻されたメンバーのハンドルの配列を使用して、メンバーの問合せが可能になります。

```
    member_fields = "<SelectMbrInfo ( MemberName, Consolidation, MemberFormula ) ";
queryCount = 2;
member_selectionArray[0] = "@ichild(Product)";
member_selectionArray[1] = "@ichild(Market)";
MaxCountArray[0] = -1;
MaxCountArray[1] = -1;
phMemberArrayArray[0] = ESS_NULL;
phMemberArrayArray[1] = ESS_NULL;
pqryErrorListArray[0] = ESS_NULL;
pqryErrorListArray[1] = ESS_NULL;

sts = EssOtlQueryMembersExArray(hOutline, member_fields, queryCount,
member_selectionArray, MaxCountArray, &phMemberArrayArray, pqryErrorListArray);

if (sts != 0) goto error_exit;
```

EssOtlQueryMembersEx()、EssOtlGetMemberField()および ESS_OTLQUERYERRORLIST_T を使用し、EssOtlFreeMembers()および EssFree()の呼出しを含んでいる例については、1070 ページの「拡張メンバーのクエリー・コードの例」を参照してください。

関連トピック

- EssOtlFreeMembers
- EssOtlGetDimensionUserAttributes
- EssOtlGetMemberField
- EssOtlOpenOutlineQuery
- EssOtlQueryMembers
- EssOtlQueryMembersByName

EssOtlQueryObjects

指定されたタイプの入力オブジェクト名のオブジェクト・ハンドルの配列を戻します。または、*pcount がゼロの場合は、すべてのオブジェクト・ハンドルを戻します。

構文

```
ESS_FUNC_M EssOtlQueryObjects(
hOutline, objType, objNames, pcount, ppObjHandles
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(クエリー・モードのみ)
objType	ESS_OBJECT_TYPES	オブジェクトのタイプ: ● OBJECT_SMARTLIST オブジェクト・タイプはテキスト・リスト(スマートリスト)
objNames	ESS_PSTR_T	クエリーを実行するオブジェクト名の配列
pcount	ESS_PULONG_T	オブジェクト名のカウント。pcount がゼロの場合、実行時のテキスト・リスト(SmartList)ハンドルの数が含まれます。
ppObjHandles	ESS_PPHOBJECT_T	オブジェクト・ハンドルの配列 EssOtlFreeObjectArray を使用して割り当てる必要があります

戻り値

戻り値:

- 0 - 正常終了の場合

  pcount には値が含まれます。
- エラー番号 - 失敗した場合

  pcount は NULL です。

例

```
    void TestFreeObjectArray()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T             hOutline = ESS_NULL;
    ESS_OBJDEF_T                 Object;
    ESS_STR_T                    objNames[1];
    ESS_OBJECT_TYPES         objType;
    ESS_ULONG_T                 count;
    ESS_PHOBJECT_T             hObjHandles = ESS_NULL;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =              hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =      szFileName;

    /* Set up */
    sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
     count = 2;
    objType = OBJECT_SMARTLIST;
    objNames[0] = "Smartlist1";
    objNames[1] = "Smartlist2";

    /* Query objects */
    sts =
    EssOtlQueryObjects(hOutline, objType,
                    objNames, &Count, &hObjHandles);


    /* Free object array */
    if(hObjHandles)
    {
        sts = EssOtlFreeObjectArray(hOutline, count, hObjHandles);
    }

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount

- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlQueryVaryingAttributes

特定の属性メンバーまたは関数のメンバー情報のクエリーを行い、各種属性のパースペクティブの指定を可能にします。

構文

```
      ESS_FUNC_M EssOtlQueryVaryingAttributes (
      hOutline, pAttrQuery, pPerspective,
pCount, pphMembers
      );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pAttrQuery	ESS_PATTRIBUTEQUERY_T	クエリーを定義する構造体へのポインタ。 • pAttrQuery.bInputMemberIsHandle = ESS_TRUE の場合、pAttrQuery.uInputMember.hMember にメンバーへのハンドルが割り当てられていることを確認してください。 • pAttrQuery.bInputMemberIsHandle = ESS_FALSE の場合、pAttrQuery.uInputMember.szMember にメンバー名が割り当てられていることを確認してください。
pPerspective	ESS_PPERSPECTIVE_T	クライアントまたはサーバーで関連付けをクエリーする際に使用される、独立したメンバーの集合へのポインタ。
pCount	ESS_PMBRCOUNTS_T	戻される基本メンバーの数へのポインタ。
pphMembers	ESS_PPHMEMBER_T	属性メンバーのハンドルの配列へのポインタ。

備考

EssOtlQueryAttributesEx に類似のこの関数は、属性のクエリーを行います。クエリーに入力基本メンバーおよび出力属性メンバー、または入力属性メンバーおよび出力基本メンバーが含まれる場合、特定のパースペクティブを使用して、そのパースペクティブで有効な関連付けに基づき、結果を限定します。

構造体 ESS_VARYING_ATTRIBUTEQUERY_T は ESS_ATTRIBUTEQUERY_T とバージョンごとに属性次元のフィールドが含まれている点を除き同一です。

パースペクティブは、独立したメンバーを個別に指定する必要がある点に注意してください。

パースペクティブが指定されていない場合、またはパースペクティブで独立したメンバーの NULL セットが指定されている場合、独立したメンバーの任意の組合せに対して存在するすべての関連付けがルーチンによって検討されます。この場合、戻された妥当性セットには、個別の独立したメンバーの範囲が含まれることがあり、クライアントはこれらを分割する必要があります。

戻り値

成功の場合、0 が戻されます。

例

```
    void TestEssOtlQueryVaryingAttributes()
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T    hOutline = ESS_NULL;
    ESS_OBJDEF_T    Object;
    ESS_USHORT_T    i = 0;
    ESS_HMEMBER_T    hBaseMbr = ESS_NULL;
    ESS_PMBRINFO_T    pMbrInfo = ESS_NULL;
    ESS_VARYING_ATTRIBUTEQUERY_T    pAttrQuery;
    ESS_MBRCOUNTS_T    Counts;
    ESS_HMEMBER_T    hIndepMbrHandlesArray[4];
    ESS_PERSPECTIVE_T Perspective;
    ESS_PHMEMBER_T    phMbrHandles;
    ESS_HMEMBER_T    hAttrMbr;
    ESS_HMEMBER_T    hAttrDim;
    ESS_PREDICATE_T Predicate;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;

    printf("\n");
    sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
    printf("EssOtlOpenOutlineQuery sts: %ld\n",sts);

    memset(&Counts, '\0', sizeof(Counts));
    Counts.ulStart = 0;
    Counts.ulMaxCount = 0;

    /* Get handles for independent members */
    memset(&Predicate, '\0', sizeof(Predicate));
    Predicate.ulQuery = ESS_SEARCH;
    Predicate.ulOptions = ESS_MEMBERSONLY;
    Predicate.pszDimension  = "";
    Predicate.pszString1 = "Jan";
    Predicate.pszString2 = "";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[0] = phMbrHandles[0];
    hIndepMbrHandlesArray[2] = phMbrHandles[0];
```

```
    Predicate.pszString1 = "FY03";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[1] = phMbrHandles[0];
    Predicate.pszString1 = "FY04";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hIndepMbrHandlesArray[3] = phMbrHandles[0];

    /* Get handles for attribute member and dimension */
    Predicate.pszString1 = "Type";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hAttrDim = phMbrHandles[0];
    Predicate.pszString1 = "Contractor";
    sts = EssOtlQueryMembersByName(hOutline, ESS_NULL, &Predicate, &Counts,
&phMbrHandles);
    hAttrMbr = phMbrHandles[0];


    memset(&Perspective, '\0', sizeof(ESS_PERSPECTIVE_T));
    Perspective.usValiditySetType = ESS_VALIDITYSET_TYPE_MBRHDLS;
    Perspective.countOfIndepDims = 2;
    Perspective.countOfIndepRanges = 1;
    Perspective.pIndepMbrs = hIndepMbrHandlesArray;

    /* Query by handle with InputMemberType of ESS_ATTRIBUTE_MEMBER and
OutputMemberType of ESS_BASE_MEMBER*/
    printf("\n*** Query by handle with InputMemberType of ESS_ATTRIBUTE_MEMBER and
OutputMemberType of ESS_BASE_MEMBER:\n");
    memset(&pAttrQuery, '\0', sizeof(ESS_ATTRIBUTEQUERY_T));
    pAttrQuery.bInputMemberIsHandle = ESS_TRUE;
    pAttrQuery.uInputMember.hMember = hAttrMbr;
    pAttrQuery.uAttributeDimension.hMember = hAttrDim;
    pAttrQuery.usInputMemberType = ESS_ATTRIBUTE_MEMBER;
    pAttrQuery.usOutputMemberType = ESS_BASE_MEMBER;
    pAttrQuery.Attribute.usDataType = ESS_ATTRMBRDT_NONE;
    pAttrQuery.usOperation = ESS_ALL;

    sts = EssOtlQueryVaryingAttributes(hOutline, &pAttrQuery, &Perspective, &Counts,
&phMbrHandles);
    printf("EssOtlQueryVaryingAttributes sts: %d\n", sts);
    if (!sts)
    {
        if(phMbrHandles)
        {
            GetMemberInfo(hOutline, Counts, phMbrHandles);
            if(Counts.ulReturnCount && phMbrHandles)
                sts = EssOtlFreeMembers(hOutline, Counts.ulReturnCount,
phMbrHandles);
        }
        else
            printf("\tNo member returned.\n");
    }


    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
```

```
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);

    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlVaryingAssociateAttribute
- EssOtlVaryingAssociateAttributeDimension
- EssOtlVaryingDisassociateAttribute
- EssOtlVaryingGetAssociatedAttributes
- EssOtlVaryingGetAttributeIndepDims

EssOtlRenameAliasTable

既存の別名テーブル名を変更します。

構文

```
ESS_FUNC_M
EssOtlRenameAliasTable
 (
hOutline, pszAliasTable, pszNewAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable	ESS_STR_T	変更する別名テーブル名。
pszNewAliasTable	ESS_STR_T	別名テーブルの新しい名前。

備考

- デフォルトの別名テーブルの名前を、"Default"から変更しないでください。
- 別名テーブルの名前を変更した場合、別名テーブルに関連付けられた言語コードは名前が変更された別名テーブルに保持されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_RENAMEDEFALIAS
- OTLAPI_ERR_ALIASTABLENAME
- OTLAPI_ERR_ALIASTABLEEXISTS

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlRenameAliasTable(hOutline,
    "Alias Table 2","2nd alias table");
}
```

関連トピック

- EssOtlCreateAliasTable
- EssOtlCopyAliasTable
- EssOtlClearAliasTable
- EssOtlDeleteAliasTable
- EssOtlSetAliasTableLanguage

EssOtlRenameMember

メンバー名を変更します。

構文

```
ESS_FUNC_M
EssOtlRenameMember
 (
hOutline, hMember, pszNewMember
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	名前を変更するメンバーのハンドル
pszNewMember	ESS_STR_T	新しいメンバー名

備考

- すべての共有メンバー名も変更されます。
- この呼出しは、hMember パラメータが共有メンバーを指す場合には失敗します。
- ESS_ATTRMBRDT_STRING 型でない、ゼロレベル(リーフ・ノード)属性のメンバーの名前を変更すると、次の値がリセットされます:
  - 属性値
  - メンバーのロング名、これは 125 ページの「ESS_ATTRSPECS_T」構造体のアウトラインの指定を使用しています
- 祖先の名前を変更すると、ゼロレベルの属性メンバーのロング名に影響する場合があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_MBRNAME
- OTLAPI_BAD_RENAMESHARE
- OTLAPI_ERR_RENAMENAMEUSED

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMemberJan;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;
```

```
sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}

if (!sts && hMemberJan)
{
 sts = EssOtlRenameMember(hOutline, hMemberJan,
    "January prelim");
}
```

関連トピック

- EssOtlFindMember
- EssOtlMoveMember
- EssOtlAddMember
- EssOtlDeleteMember

EssOtlRestructure

サーバー上のアウトラインを再構築します。これは非同期の呼出しです。

構文

```
ESS_FUNC_M
EssOtlRestructure
 (
hCtx, usRestructType
);
```

パラメータ	データ型	説明
hCtx	ESS_HCTX_T	サーバー・ログイン・コンテキスト・ハンドル。これは、**EssOtlWriteOutline()**によってアウトラインが保存されたサーバーである必要があります。
usRestructType	ESS_USHORT_T	実行する再構築のタイプ。これは、次のいずれかの値にできます: • ESS_DOR_ALLDATA • ESS_DOR_INDATA • ESS_DOR_LOWDATA • ESS_DOR_NODATA • ESS_DOR_FORCE_ALLDATA

備考

- この関数を呼び出す前に、**EssOtlWriteOutline()**を使用してアウトラインを保存する必要があります。
- この呼出しは、サーバーに保存されたアウトラインに対してのみ有効です。

- これは非同期の呼出しです。この呼出しを行った後、**EssGetProcessState()**で再構築操作の完了を示すステータスが戻されるまで、**EssGetProcessState()**を呼び出す必要があります。
- データが正しく再構築されるためには、fKeepTrans フラグを ESS_TRUE に設定した **EssOtlOpenOutline()**を使用して、アウトラインを開いておく必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は OTLAPI_BAD_RESTRUCTTYPE 構造体が戻されます。

アクセス

この関数を使用するには、呼出し元は指定したアプリケーション、アウトライン・オブジェクトの保存先であるデータベースのいずれか、または両方に対するアクセス権が必要です。アウトライン・オブジェクトを再構築するには、指定したアプリケーション、またはアウトラインが含まれるデータベースに対して、アプリケーション・デザイナまたはデータベース・デザイナの権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)が必要です。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HCTX_T    hCtx;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/* body of code */
/* write outline to server using */
/* EssOtlWriteOutline()     */

if (!sts)
{
 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
}
```

```
/* need to call EssGetProcessState() */
/* to check for completion before proceeding */
```

関連トピック

- EssOtlOpenOutline
- EssOtlNewOutline
- EssOtlWriteOutline
- EssOtlVerifyOutline
- EssOtlCloseOutline

EssOtlSetAggLevelUsage

保管されている階層に対してビュー選択プロパティが適用されます。

構文

```
ESS_FUNC_M EssOtlSetAggLevelUsage (
hOutline, hMember, sAgglevelUsage
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)値
hMember	ESS_HMEMBER_T	階層メンバー(入力)。0
sAgglevelUsage	ESS_SHORT_T	レベルの使用定数のいずれか(入力)。

定数	値	説明
ESS_AGGLEVELUSAGE_DEFAULT	11	プライマリ階層では、すべてのレベルを検討します。別のロールアップが使用可能になっていないかぎり、セカンダリ階層の集約は行いません。
ESS_AGGLEVELUSAGE_ALL	12	集約にすべてのレベルを検討します。これはセカンダリ階層ではなく、プライマリ階層のデフォルトと同じです。
ESS_AGGLEVELUSAGE_NOAGGREGATION	13	この階層に沿って集約を行いません。選択したすべてのビューは入力レベルです。
ESS_AGGLEVELUSAGE_BOTTOMONLY	14	セカンダリ階層にのみ適用されます。この階層の最下位レベルのみを集約に検討します。
ESS_AGGLEVELUSAGE_TOPONLY	15	プライマリ階層にのみ適用されます。この階層の最上位レベルのみを集約に検討します。

パラメータ　　データ型　　説明

定数	値	説明
ESS_AGGLEVELUSAGE_BOTTOMTOP	16	プライマリ階層に適用されます。最上位と最下位のレベルのみを選択します。

備考

- この関数は、リリース 9.3 以上の集約ストレージ・データベースにのみ適用可能です。
- この関数を使用して、保管されている階層にビュー選択プロパティを適用し、Essbase が集約に特定のレベルを選択するのを防止します。

戻り値

正常終了の場合は、0 が戻されます。

例

```
     ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T   hOutline = ESS_NULL;
 ESS_HMEMBER_T    hMember = ESS_NULL;
 ESS_SHORT_T    sAggLevelUsage = 0;

 /* code to assign hOutline variable omitted */
 /* code to assign hMember variable omitted */
 /* code to assign sAggLevelUsage variable omitted */

 if (hOutline && hMember)
 {
   sts = EssOtlSetAggLevelUsage (hOutline, hMember, sAggLevelUsage);
if (sts)
 printf("Error (%ld) setting AggLevelUsage\n", sts);
 }
 else
 {
   if (!hOutline)
    printf("Outline not provided\n");
   if (!hMember)
    printf("Member not provided\n");
 }
```

関連トピック

- EssOtlGetAggLevelUsage

EssOtlSetAliasTableLanguage

指定した別名テーブルの言語コードを設定します。

別名テーブルの言語コードを設定すると、ApplCore セッションで実行されているアプリケーションが Essbase データベースにアクセスしたときに、アプリケーション選択で正しい別名テーブルが自動的に選択されます。

構文

```
ESS_FUNC_M
EssOtlSetAliasTableLanguage
(
hOutline
,
pszAliasTable
,
pszLanguageCode
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszAliasTable	ESS_STR_T	言語コードを設定する別名テーブル名。
pszLanguageCode	ESS_STR_T	pszAliasTable で指定された別名テーブルに割り当てる言語コード。 言語コードは、ApplCore セッションからの中間層言語タグである必要があります。言語コードの大文字と小文字は区別されません。

備考

- デフォルトの別名テーブルで言語コードを設定することはできません。
- 別名テーブルにはいくつでも言語コードを割り当てることができます。複数の言語コードを設定するには、言語コードごとにこの関数を呼び出します。
- 新しい言語コードを設定しても、別名テーブルに現在割り当てられている言語コードは上書きされません。
- 同じ言語コードを同じデータベース内の別の別名テーブルに割り当てないでください。

戻り値

- 成功の場合、0 が戻されます。
- 失敗した場合は、次のいずれかのエラーが戻されます:
  - OTLAPI_BAD_ALIASTABLE (無効な別名テーブル)
  - OTLAPI_ERR_DUP_LANGCODE (言語コードが同じデータベース内の別の別名テーブルに割り当てられている)

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
#include <essapi.h>
```

```
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OUTLINEINFO_T  NewInfo;
ESS_HOUTLINE_T   hOutline;
ESS_PALIASLANG_T  pLangs=ESS_NULL;
ESS_ULONG_T    nLangs = 0, i=0;

memset(&NewInfo, '\0', sizeof(NewInfo));
sts = EssOtlNewOutline(hCtx, &NewInfo, &hOutline);

if (!sts)
{
 sts = EssOtlCreateAliasTable(hOutline,
 "French Alias Table");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr");
}

if (!sts)
{
 sts = EssOtlSetAliasTableLanguage (hOutline,
 "French Alias Table", "fr-CA");
}

if (!sts)
{
    sts = EssOtlGetAliasTableLanguages(hOutline, "French Alias Table", &nLangs,
&pLangs);

 if ( !sts == ESS_STS_NOERR && ( pLangs) )
 {
   for (i=0;i<nLangs ;++i)
   {
     if (pLangs[i])
     {
       printf("Language Code:  %s\n", pLangs[i]);
     }
   }
   EssFree(hInst, pLangs);
 }
}
if (!sts)
{
 sts = EssOtlClearAliasTableLanguages (hOutline,
 "French Alias Table");
}
```

関連トピック

- EssOtlGetAliasTableLanguages
- EssOtlClearAliasTableLanguages

EssOtlSetAltHierarchyEnabled

次元を複合階層使用可能に設定します。

構文

```
ESS_FUNC_M EssOtlSetAltHierarchyEnabled(
hOutline
,
hDimMember
,
cEnabled
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDimMember	ESS_HMEMBER_T	次元メンバー(入力)。
cEnabled	ESS_BOOL_T	TRUE の場合、次元は複合階層使用可能に設定されます。FALSE の場合、次元は単一の保管階層に設定されます。

戻り値

- 0 - 正常終了の場合
- hDimMember が次元メンバーでない場合、エラー OTLAPI_ERR_BADDIM を戻します。

関連トピック

- EssOtlGetAltHierarchyEnabled
- EssOtlGetHierarchyType
- EssOtlSetHierarchyType

EssOtlSetASOCompressionDimension

集約ストレージ次元に圧縮タグを付けます。

構文

```
ESS_FUNC_M
EssOtlSetASOCompressionDimension
 (
hOutline, hDim
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hDim	ESS_HMEMBER_T	次元ハンドル(入力)。

備考

- デフォルトでは、集約ストレージ・データベース内の圧縮次元は会計次元です。現行の圧縮次元を入手するには、EssOtlGetASOCompressionDimension を使用します。圧縮次元を変更すると、データベース全体の再構築がトリガーされます。
- 同時に複数の次元を圧縮次元に使用できません。API は新しい圧縮次元を設定すると、API によって前の次元が自動的に設定解除されます。属性次元は圧縮次元に使用できません。
- アウトラインにどの次元も圧縮次元として選択していなくても問題ありません。hDim を NULL に設定してこの関数を呼び出すと、現行の圧縮次元の設定が解除されます。
- Essbase では単一の動的階層を圧縮次元として指定する必要があります。次元に複数の階層など、異なる階層設定がある場合、自動的に単一の動的階層に設定されます。そして元の階層設定は失われます(異なる次元を圧縮用に設定しても、元の階層設定には戻りません)。
- 圧縮次元の選択は、パフォーマンスに大きな影響を与えることがあります。圧縮次元に大きな次元を選択することはお薦めしません。

戻り値

正常終了の場合は、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T   hOutline = ESS_NULL;
 ESS_HMEMBER_T    hMember = ESS_NULL;

 /* code to assign hOutline variable omitted */
 /* code to assign hMember variable omitted */

 if (hOutline)
 {
   sts = EssOtlSetASOCompressionDimension(hOutline, hMember);
if (sts)
 printf("Error (%ld) setting compression dimension\n", sts);
else
 if (hMember)
  printf("Compression dimension set\n");
 else
  printf("Compression dimension cleared\n");
 }
 else
 {
   printf("Outline not provided\n");
 }
```

関連トピック

- EssOtlGetASOCompressionDimension

EssOtlSetAttributeSpecifications

アウトラインの属性指定を設定します。

構文

パラメータ	データ型	説明
hOutline;	ESS_HOUTLINE_T	アウトラインのハンドル
pAttrSpecs;	125 ページの「ESS_ATTRSPECS_T」	属性指定

備考

- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供
- 属性指定を設定しない場合、アウトラインではデフォルトの属性指定が使用されます。
- 属性指定を変更すると、再構築されることがあります。

戻り値

属性メンバー名の変更が正常に実行されなかった場合は、OTLAPI_ERR_ATTRRENAMENAMEUSED エラーが戻されます。

例

```
    void  ESS_OtlSetAttributeSpecifications()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_ATTRSPECS_T  AttrSpecs;
 ESS_CHAR_T    buffer[8][20];
 ESS_OBJDEF_T   Object;
 ESS_HOUTLINE_T  hOutline;
 ESS_APPNAME_T   szAppName;
 ESS_DBNAME_T   szDbName;
 ESS_OBJNAME_T   szFileName;
 ESS_PROCSTATE_T  pState;
 int       test;

 memset(&Object, '\0', sizeof(Object));
  Object.hCtx = hCtx;
  Object.ObjType = ESS_OBJTYPE_OUTLINE;
  strcpy(szAppName, "Sample");
  strcpy(szDbName, "Basic");
  strcpy(szFileName, "Basic");
  Object.AppName = szAppName;
  Object.DbName = szDbName;
  Object.FileName = szFileName;
```

```
  sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);

 printf("\n\nEnter the NUMBERS for the appropriate choices that follow.");
 printf("\n\nEnter GenNameBy:\n\t\t0. ESS_GENNAMEBY_PREFIX\n\t\t1.
ESS_GENNAMEBY_SUFFIX\n\nChoice: ");
 test = atoi(gets(buffer[0]));
 switch (test)
  {
   case 0:
    AttrSpecs.usGenNameBy=ESS_GENNAMEBY_PREFIX;
    break;
   case 1:
    AttrSpecs.usGenNameBy=ESS_GENNAMEBY_SUFFIX;
    break;
   default:
    printf("\n\nInvalid choice.\n\n");
  }

 printf("\n\nEnter UseNameOf:\n\t\t0.  ESS_USENAMEOF_NONE\n\t\t1.
ESS_USENAMEOF_PARENT");
 printf("\n\t\t2.  ESS_USENAMEOF_GRANDPARENTANDPARENT\n\t\t3.
ESS_USENAMEOF_ALLANCESTORS");
 printf("\n\t\t4.  ESS_USENAMEOF_DIMENSION\n\nChoice: ");
 test = atoi(gets(buffer[0]));
 switch (test)
  {
   case 0:
    AttrSpecs.usUseNameOf=ESS_USENAMEOF_NONE;
    break;
   case 1:
    AttrSpecs.usUseNameOf=ESS_USENAMEOF_PARENT;
    break;
   case 2:
    AttrSpecs.usUseNameOf=ESS_USENAMEOF_GRANDPARENTANDPARENT;
    break;
   case 3:
    AttrSpecs.usUseNameOf=ESS_USENAMEOF_ALLANCESTORS;
    break;
   case 4:
    AttrSpecs.usUseNameOf=ESS_USENAMEOF_DIMENSION;
    break;
   default:
    printf("\n\nInvalid choice.\n\n");
  }

  printf("Enter Delimiter:\n\t\t0.  ESS_DELIMITER_UNDERSCORE\n\t\t1.
ESS_DELIMITER_PIPE");
 printf("\n\t\t2.  ESS_DELIMITER_CARET\n\nChoice: ");
 test = atoi(gets(buffer[0]));
 switch (test)
  {
   case 0:
    AttrSpecs.cDelimiter=ESS_DELIMITER_UNDERSCORE;
    break;
   case 1:
    AttrSpecs.cDelimiter=ESS_DELIMITER_PIPE;
```

```
    break;
   case 2:
    AttrSpecs.cDelimiter=ESS_DELIMITER_CARET;
    break;
   default:
    printf("\n\nInvalid choice.\n\n");
  }

 printf("Enter DateFormat:\n\t\t0.  ESS_DATEFORMAT_MMDDYYYY\n\t\t1.
ESS_DATEFORMAT_DDMMYYYY\n\nChoice: ");
 test = atoi(gets(buffer[0]));
 switch (test)
  {
   case 0:
    AttrSpecs.usDateFormat=ESS_DATEFORMAT_MMDDYYYY;
    break;
   case 1:
    AttrSpecs.usDateFormat=ESS_DATEFORMAT_DDMMYYYY;
    break;
   default:
    printf("\n\nInvalid choice.\n\n");
  }

  printf("Enter BucketingType:\n\t\t0.  ESS_UPPERBOUNDINCLUSIVE\n\t\t1.
ESS_LOWERBOUNDINCLUSIVE");
 printf("\n\t\t2.  ESS_UPPERBOUNDNONINCLUSIVE\n\t\t3.  ESS_LOWERBOUNDNONINCLUSIVE\n
\nChoice:  ");
 test = atoi(gets(buffer[0]));
 switch (test)
  {
   case 0:
    AttrSpecs.usBucketingType=ESS_UPPERBOUNDINCLUSIVE;
    break;
   case 1:
    AttrSpecs.usBucketingType=ESS_LOWERBOUNDINCLUSIVE;
    break;
   default:
    printf("\n\nInvalid choice.\n\n");
  }

 printf("\nEnter a word for your default true string (or 'ESS_DEFAULT_TRUESTRING'):
\n");

 gets(buffer[0]);
 if (buffer[0] == "ESS_DEFAULT_TRUESTRING")
  AttrSpecs.pszDefaultTrueString = "";
 else
  AttrSpecs.pszDefaultTrueString=buffer[0];

  printf("\nEnter your default false string (or 'ESS_DEFAULT_FALSESTRING'):\n");
 gets(buffer[1]);
 if (buffer[1] == "ESS_DEFAULT_FALSESTRING")
  AttrSpecs.pszDefaultFalseString = "";
 else
  AttrSpecs.pszDefaultFalseString=buffer[1];

  printf("\nEnter your default attribute calculation dimension name (or
```

```
'ESS_DEFAULT_ATTRIBUTECALCULATIONS'):\n");
 gets(buffer[2]);
 if (buffer[2] == "ESS_DEFAULT_ATTRIBUTECALULATIONS")
  AttrSpecs.pszDefaultAttrCalcDimName="";
 else
  AttrSpecs.pszDefaultAttrCalcDimName=buffer[2];

 printf("\nEnter your default sum member name (or 'ESS_DEFAULT_SUM'):\n");
 gets(buffer[3]);
 if (buffer[3] == "ESS_DEFAULT_SUM")
  AttrSpecs.pszDefaultSumMbrName = "";
 else
  AttrSpecs.pszDefaultSumMbrName=buffer[3];

 printf("\nEnter your default count member name (or 'ESS_DEFAULT_COUNT'):\n");
 gets(buffer[4]);
 if (buffer[4] == "ESS_DEFAULT_COUNT")
  AttrSpecs.pszDefaultCountMbrName = "";
 else
  AttrSpecs.pszDefaultCountMbrName=buffer[4];

 printf("\nEnter your default average member name (or 'ESS_DEFAULT_AVERAGE'):\n");
 gets(buffer[5]);
 if (buffer[5] == "ESS_DEFAULT_AVERAGE")
  AttrSpecs.pszDefaultAverageMbrName = "";
 else
  AttrSpecs.pszDefaultAverageMbrName=buffer[5];

 printf("\nEnter your default minimum member name (or 'ESS_DEFAULT_MIN'):\n");
 gets(buffer[6]);
 if (buffer[6] == "ESS_DEFAULT_MIN")
  AttrSpecs.pszDefaultMinMbrName = "";
 else
  AttrSpecs.pszDefaultMinMbrName=buffer[6];

 printf("\nEnter your default maximum member name (or 'ESS_DEFAULT_MAX'):\n");
 gets(buffer[7]);
 if (buffer[7] == "ESS_DEFAULT_MAX")
  AttrSpecs.pszDefaultMaxMbrName = "";
 else
  AttrSpecs.pszDefaultMaxMbrName=buffer[7];

 sts = EssOtlSetAttributeSpecifications(hOutline, &AttrSpecs);
 printf("EssOtlSetAttributeSpecifications() sts: %ld\n",sts);

 sts = EssOtlWriteOutline(hOutline, &Object);
 printf("EssOtlWriteOutline() sts: %ld\n",sts);

 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
 printf("EssOtlRestructure() sts: %ld\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
```

```
 sts = EssOtlCloseOutline(hOutline);
 printf("EssOtlCloseOutline() sts: %ld\n",sts);
}
```

関連トピック

- EssCheckAttributes
- EssFreeStructure
- EssGetAssociatedAttributesInfo
- EssGetAttributeInfo
- EssGetAttributeSpecifications
- EssOtlAssociateAttributeDimension
- EssOtlAssociateAttributeMember
- EssOtlDisassociateAttributeDimension
- EssOtlDisassociateAttributeMember
- EssOtlFindAttributeMembers
- EssOtlFreeStructure
- EssOtlGetAssociatedAttributes
- EssOtlGetAttributeInfo
- EssOtlGetAttributeSpecifications
- EssOtlQueryAttributes

EssOtlSetDateFormatString

この関数は、アウトライン・プロパティ日付フォーマット文字列を設定します。

構文

```
  ESS_FUNC_M EssOtlSetDateFormatString(
ESS_HOUTLINE_T hOutline,
ESS_STR_T  formatString)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	スマートリストのアウトライン。
formatString	ESS_STR_T	この引数にアウトライン日付フォーマット文字列を戻します。

戻り値

戻り値:

- 0 - 正常終了の場合

  formatString は日付フォーマット文字列を含みます。
- エラー番号 - 失敗した場合

例

```
 void TestGetSetDateFormatString()
```

```
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T             hOutline = ESS_NULL;
    ESS_OBJDEF_T                 Object;
    ESS_SHORT_T                 length = 80;
    ESS_STR_T                    dateFormatString = "";
    ESS_STR_T                    localeStr;
    ESS_USHORT_T                count, i;
    ESS_STR_T*                  pdateStrings;
    ESS_STR_T*                  pformatStrings;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =          hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =      szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object,
                ESS_TRUE, ESS_TRUE, &hOutline);

    /* Get current value */
    sts = EssOtlGetDateFormatString(hOutline, &dateFormatString);
    printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
    printf("\tDate format string: %s\n", dateFormatString);

    printf("\n");
    localeStr = "English_UnitedStates.Latin1@Binary";
    sts = EssOtlGetServerDateFormats(hCtx, localeStr,
                &Count, &pdateStrings, &pformatStrings);
    printf("EssOtlGetServerDateFormats sts: %d \n", sts);

    for (i = 0; i < count; i++)
    {
        printf("\nCase with %s:\n", pformatStrings[i]);
        sts =
    EssOtlSetDateFormatString(hOutline,
                    pformatStrings[i]);

        printf("EssOtlSetSMDateFormatString sts: %d \n", sts);
        SaveOutline(hOutline);

        sts = EssOtlGetDateFormatString(hOutline,
                    &dateFormatString);
        printf("EssOtlGetSMDateFormatString sts: %d \n", sts);
        printf("\tDate format string: %s\n", dateFormatString);

    }
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);
    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline  sts: %d\n",sts);
}
```

関連トピック

- EssOtlGetServerDateFormats
- EssOtlGetDateFormatString

EssOtlSetDimensionNameUniqueness

重複する一意でないメンバー名を禁止するように設定します。

構文

```
ESS_FUNC_M EssOtlSetDimensionNameUniqueness (
hOutline, hMember, bNameUnique
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	次元ルート・メンバーのメンバー・ハンドル(入力)。
bNameUnique	ESS_BOOL_T	次元メンバー名の一意の設定(入力)。TRUE の場合、次元に重複するメンバー名を持たせることはできません。

備考

EssOtlFindMember を呼び出して、ESS_HMEMBER_T (hDim)変数を設定してください。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_GetSetDimNameUniq()
{
ESS_STS_T   sts = 0;
ESS_POUTLINEINFO_T  pInfo = ESS_NULL;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_BOOL_T  pbNameUnique;
ESS_HMEMBER_T  hDim = ESS_NULL;


memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName =  szAppName;
```

```
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Year",&hDim);

 if (sts)
  printf("EssOtlFindMember failed sts %ld\n",sts);
}

/*Get the dimension's, Year, member-name uniqueness setting */
if (!sts)
{
 sts = EssOtlGetDimensionNameUniqueness (hOutline, hDim, &pbNameUnique);

 if (sts)
  printf("EssOtlGetDimensionNameUniqueness failed sts %ld\n",sts);
 else
  printf("Dimension Year has Member Name Uniqueness value: %ld\n", pbNameUnique);
}

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Product",&hDim);

 if (sts)
  printf("EssOtlFindMember failed sts %ld\n",sts);
}

if (!sts)
{
 /*set Product to prohibit duplicate (non-unique) member names*/
 pbNameUnique = ESS_TRUE;
 sts = EssOtlSetDimensionNameUniqueness (hOutline, hDim, pbNameUnique);

 if (sts)
  printf("EssOtlSetDimensionNameUniqueness failed sts %ld\n",sts);
 else
  printf("Dimension Product has Member Name Uniqueness value: %ld\n", pbNameUnique);
}

return sts;

}
```

関連トピック

- EssOtlGetDimensionNameUniqueness

EssOtlSetDimensionSolveOrder

次元の解決順を設定します。

構文

```
ESS_FUNC_M EssOtlSetDimensionSolveOrder (
hOutline, hMember, cOrder
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	次元ハンドル(入力)。
cOrder	ESS_UCHAR_T	解決順(入力)。0 - 127

備考

- メンバーまたは次元の解決順プロパティは、計算順序を指定します。
- メンバーの解決順は次元の解決順よりも優先されます。解決順は 0 から 127 までにできます。デフォルトは 0 です。
- 解決順が指定されていない式を持たないメンバーは、その次元の解決順を継承します。解決順が指定されていない式を持つメンバーは、ゼロの解決順を持ちます。

戻り値

正常終了の場合は、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_HMEMBER_T   hMember = ESS_NULL;
ESS_UCHAR_T    ucOrder = 0;

/* code to assign hOutline variable omitted */
/* code to assign hMember variable omitted */
/* code to assign ucOrder variable omitted */


if (hOutline && hMember)
{
 if (ucOrder > 127)
 {
  printf("Solveorder must be less than 128\n");
 {
 else
 {
  sts = EssOtlSetDimensionSolveOrder(hOutline, hMember, ucOrder);

  if (sts)
   printf("Error [%ld] returned\n", sts);
```

```
    else
     printf("Solve Order: %d\n", ucOrder);
  }
 }
 else
  printf("Both hOutline and hMember must have values\n");
```

関連トピック

- EssOtlGetDimensionSolveOrder
- EssOtlSetMemberSolveOrder
- EssOtlGetMemberSolveOrder

EssOtlSetDTSMemberAlias

動的時系列(Dynamic Time Series: DTS)メンバーの別名を設定します。

構文

```
ESS_STS_T
EssOtlSetDTSMemberAlias
 (
hOutline, pszDTSMember, pszAlias, pszAliasTable
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	EssOtlOpenOutline 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember	ESS_STR_T	別名を提供する DTS メンバー名。
pszAlias	ESS_STR_T	DTS メンバーの別名を含む C 文字列を指すポインタ。
pzsAliasTable	ESS_STR_T	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルが使用されます。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_ILLEGALALIASSTRING
- OTLAPI_ERR_DUPLICATEALIAS

例

```
    #include "essapi.h"
#include "essotl.h"
#include "esserror.h"
```

```
ESS_STS_T ESS_OtlSetDTSMemberAlias(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
    ESS_STS_T    sts =ESS_STS_NOERR;
    ESS_OBJDEF_T  Object;
    ESS_HOUTLINE_T hOutline;
    ESS_APPNAME_T  szAppName;
    ESS_DBNAME_T  szDbName;
    ESS_OBJNAME_T  szFileName;
    ESS_CHAR_T   pszAliasTable[ESS_ALIASNAMELEN];
    ESS_CHAR_T   pszAlias[ESS_ALIASNAMELEN];
    ESS_CHAR_T   pszDTSMember[ESS_MBRNAMELEN];
    ESS_PROCSTATE_T pState;
    ESS_ULONG_T   ulErrors;
    ESS_ULONG_T   ulCount;
    ESS_POUTERROR_T pMbrErrors = NULL;

    strcpy(szAppName, "sample");
    strcpy(szDbName, "Basic");
    strcpy(szFileName, "Basic");
    strcpy(pszDTSMember, "Q-T-D");
    strcpy(pszAliasTable, "Default");
    strcpy(pszAlias, "QuarterToDate");

    memset(&Object, '\0', sizeof(ESS_OBJDEF_T));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szFileName;

    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);

    if(sts)
    {
        printf("Could not open outline\n");
        return sts;
    }
    sts = EssOtlSetDTSMemberAlias(hOutline, pszDTSMember, pszAlias , pszAliasTable);
    if(sts)
    {
        printf("Could not set DTS member alias. Error is %d\n", sts);
    }

    sts = EssOtlWriteOutline(hOutline, &Object);
    if(sts)
    {
        printf("Could not write outline\n");
        return sts;
    }

    sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
    if(sts)
    {
        printf("Could not restructure outline\n");
        return sts;
    }
```

```
    memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
    sts = EssGetProcessState(hCtx, &pState);
    {
        while ((sts == ESS_STS_NOERR ) && (pState.State != ESS_STATE_DONE))
        {
            memset (&pState, 0, sizeof(ESS_PROCSTATE_T));
            sts = EssGetProcessState(hCtx, &pState);
        }
    }

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, szAppName, szDbName,
szFileName);
    if (sts)
    {
        printf("Could not unlock outline\n");
        return sts;
    }

    EssOtlCloseOutline(hOutline);
    return sts;
}
```

関連トピック

- EssOtlDeleteDTSMemberAlias
- EssOtlEnableDTSMember
- EssOtlGetEnabledDTSMembers
- EssOtlGetDTSMemberAlias

EssOtlSetGenName

次元内の特定世代に対して名前を設定します。

構文

```
ESS_FUNC_M
EssOtlSetGenName
 (
hOutline, pszDimension, usGen, pszName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象の世代を含む次元の名前。
usGen	ESS_USHORT_T	名前を設定する世代の番号。次元自体は世代 1 です。
pszName	ESS_STR_T	世代に与える名前。

備考

- 世代名はメンバーの名前スペース全体で一意の必要があります。他の世代、レベル、メンバー名、または別名と重複できません。重複した世代名を追加しようとすると、エラーが発生します。
- 各特定次元および世代で持つことができる名前は 1 つです。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_GENLEVELNAME
- OTLAPI_ERR_GENLEVELNAMEEXISTS
- OTLAPI_ERR_GENLEVELEXISTS
- OTLAPI_ERR_GENLEVELVALUE
- OTLAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    GenNum;
ESS_STR_T     GenName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName   = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/************  Set Generation Name  ************/
Dimension = "Year";
GenNum = 2;
GenName = "Qtr123";

if (!sts)
```

```
{
 sts = EssOtlSetGenName(hOutline, Dimension,
    GenNum, GenName);
}
```

関連トピック

- EssOtlDeleteGenName

EssOtlSetGenNameEx

指定した世代番号に関する世代名とメンバーの一意性を設定します。

構文

```
ESS_FUNC_M EssOtlSetGenNameEx (
hOutline, pszDimension, usGen, pszName, bUniqueName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	次元名。
usGen	ESS_USHORT_T	名前を設定する世代の番号。
pszName	ESS_STR_T	世代に与える名前。
bUniqueName	ESS_BOOL_T	TRUE の場合は、pszDimension の次元の usGen 世代のメンバーに重複した名前は指定できません。

備考

- この関数は世代名と世代の一意性を設定します。世代名のみを設定する場合は、EssOtlSetGenName を使用します。
- 一意性のみを設定し、名前は変更しない場合でも、名前を渡す必要があります。その場合は、EssOtlGetGenName を呼び出して、この関数への値として usGen を渡します。
- usGen パラメータに NULL を渡さないでください。

戻り値

正常終了の場合は 0 が戻されます。それ以外はエラー・コードが戻されます。

例

```
    void ESS_GetGenNameEx()

{

ESS_STS_T      sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
```

```
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    GenNum;
ESS_STR_T     GenName;
ESS_BOOL_T     bUnique= ESS_FALSE;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);
 printf("EssOtlOpenOutline sts: %ld\n",sts);

/***************  Set and Get GenName **************/
 Dimension = "Year";
 GenNum = 1;
 GenName = "Gen 1 Year";

 //SetGenNameEx() so that Gen 1 members of Year cannot be non-unique
 if (!sts)
 {
  sts = EssOtlSetGenNameEx(hOutline, Dimension,
   GenNum, GenName, ESS_TRUE);
 }



 // GetGenNameEx() to see if the gen is able to be non-unique
 if (!sts)
 {
  sts = EssOtlGetGenNameEx(hOutline, Dimension,
    GenNum, &GenName, &bUnique);
  printf("Generation 1 members of Year have bUnique value of %ld\n", bUnique);
  printf("EssOtlGetGenNameEx sts: %ld\n",sts);
 }

 if (!sts && GenName)
 {
  printf("Gen Name: %s\n",GenName);
    EssFree(hInst, GenName);
 }

}
```

関連トピック

- EssOtlSetGenName
- EssFree
- EssOtlDeleteGenName
- EssOtlGetGenNameEx

EssOtlSetHierarchyType

次元の階層タイプの指定(複合階層使用可能、動的階層、または保管階層)を設定します。

構文

```
ESS_FUNC_M EssOtlSetHierarchyType(
hOutline
,
hMember
,
cType
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	次元メンバー(入力)。
cType	ESS_UCHAR_T	hMember が次元メンバーである場合は次のいずれかの値(入力): ● ESS_STORED_HIERARCHY - 次元は単一の保管階層です。 ● ESS_DYNAMIC_HIERARCHY - 次元は単一の動的階層です。 ● ESS_MULTIPLE_HIERARCHY_IS_ENABLED - 次元は複合階層使用可能に設定されます(EssOtlSetAltHierarchyEnabled を使用する場合と同じ)。 「注意」を参照してください。

備考

次元が複合階層使用可能になると、階層タイプは世代 2 のメンバーによって決定されます。hMember が世代 2 のメンバーである場合、cType は次の値を持つ可能性があります:

- ESS_STORED_HIERARCHY - hMember が最上位である階層は単一の保管階層です。
- ESS_DYNAMIC_HIERARCHY - hMember が最上位である階層は単一の動的階層です。

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

関連トピック

- EssOtlGetHierarchyType
- EssOtlSetAltHierarchyEnabled
- EssOtlGetAltHierarchyEnabled

EssOtlSetImpliedShare

アウトラインの暗黙の共有設定を変更します。

構文

```
ESS_FUNC_M EssOtlSetImpliedShare(
hOutline
,
impliedShareSetting
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
impliedShareSetting	ESS_USHORT	暗黙の共有設定値。109 ページの「暗黙の共有設定(C)」を参照してください。

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

関連トピック

- EssOtlGetImpliedShare

EssOtlSetLevelName

次元内の特定レベルに対して名前を設定します。

構文

```
ESS_FUNC_M
EssOtlSetLevelName
 (
hOutline, pszDimension, usLevel, pszName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象のレベルを含む次元の名前。
usLevel	ESS_USHORT_T	名前を設定するレベルの番号。リーフ・メンバーはレベル 0 です。

パラメータ	データ型	説明
pszName	ESS_STR_T	レベルに与える名前。

備考

- レベル名はメンバー名と同じルールに従い、メンバーの名前スペース全体で一意の必要があります。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- 個々の特定次元およびレベルが持てる名前は 1 つです。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_GENLEVELNAME
- OTLAPI_ERR_GENLEVELNAMEEXISTS
- OTLAPI_ERR_GENLEVELEXISTS
- OTLAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0 ;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    LevelNum;
ESS_STR_T     LevelName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName  = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/************* Set Level Name **************/
Dimension = "Year";
LevelNum = 1;
LevelName = "Qtr 1 2 3";
```

```
if (!sts)
{
 sts = EssOtlSetLevelName(hOutline, Dimension,
  LevelNum, LevelName);
}
```

関連トピック

- EssOtlDeleteLevelName
- EssOtlGetLevelName

EssOtlSetLevelNameEx

あるレベルの次元にあるメンバーに対して重複する名前を持つことを禁止するかどうかを設定します。

構文

```
ESS_FUNC_M
EssOtlSetLevelNameEx
 (
hOutline, pszDimension, usLevel, pszName, bUniqueName
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pszDimension	ESS_STR_T	対象のレベルを含む次元の名前。
usLevel	ESS_USHORT_T	名前を設定するレベルの番号。リーフ・メンバーはレベル 0 です。
pszName	ESS_STR_T	レベルに与える名前。
bUniqueName	ESS_BOOL_T	TRUE の場合は、次元 pszDimension のレベル uslevel にあるメンバーに、重複した名前を付けることができません。FALSE の場合は、重複した名前を付けることができます。

備考

- レベル名はすべてのメンバー名で一意である必要があります。他の世代、レベル、メンバー名、または別名と重複できません。重複したレベル名を追加しようするとエラーが発生します。
- この関数によって、レベルの名前と一意性を設定します。名前のみを設定する場合は、EssOtlSetLevelName を使用してください。
- 一意性のみを設定し、名前は変更しない場合でも、名前を渡す必要があります。その場合は、EssOtlGetLevelName を呼び出して、この関数への値として usLevel を渡します。
- usLevel パラメータに NULL を渡さないでください。

戻り値

正常終了の場合は 0 が戻されます。それ以外はエラー・コードが戻されます。

例

```
    ESS_FUNC_M
ESS_GetLevelNameEx()
{
ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_STR_T     Dimension;
ESS_USHORT_T    LevelNum;
ESS_STR_T     LevelName;
ESS_BOOL_T     bUnique= ESS_FALSE;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Demo");
strcpy(szDbName, "Test");
strcpy(szFileName, "Test");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   ESS_TRUE, &hOutline);

/***************  Set and Get Level Name **************/
 Dimension = "Year";
 LevelNum = 0;
 LevelName = "Level 0 Year";

 //SetLevelNameEx() so that level 0 member of Year cannot be non-unique
 if (!sts)
 {
  sts =
     EssOtlSetLevelNameEx
     (hOutline, Dimension,
   LevelNum, LevelName, ESS_TRUE);
 }

 // GetLevelNameEx() to see if the level is able to be non-unique
 if (!sts)
 {
  sts = EssOtlGetLevelNameEx(hOutline, Dimension,
     LevelNum, &LevelName, &bUnique);
  printf("Level 0 members of Year have bUnique value of %ld\n", bUnique);

 }

 if (!sts && LevelName)
```

```
  {
  printf("Level Name: %s\n",LevelName);
     EssFree(hInst, LevelName);
  }

 return (sts);
}
```

関連トピック

- EssOtlGetLevelNameEx

EssOtlSetMemberAlias

特定の別名テーブルにおける特定のメンバーに対する、デフォルトのメンバーの別名を設定します。

構文

```
ESS_FUNC_M
EssOtlSetMemberAlias
 (
hOutline, hMember, pszAliasTable, pszAlias
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	別名を設定するメンバーのハンドル。
pszAliasTable	ESS_STR_T	別名を設定する別名テーブル。このパラメータが ESS_NULL の場合、デフォルトの別名テーブルが使用されます。
pszAlias	ESS_STR_T	別名。

備考

EssOtlDeleteMemberAlias()を使用して別名を削除します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_ILLEGALDEFALIAS
- OTLAPI_ERR_ILLEGALCOMBOALIAS
- OTLAPI_ERR_ILLEGALALIASSTRING
- OTLAPI_ERR_DUPLICATEALIAS

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T    hMember;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Year",
    &hMember);
}

if (!sts  && hMember)
{
 sts = EssOtlSetMemberAlias(hOutline,
    hMember, ESS_NULL, "Time Dimension");
}
```

関連トピック

- EssOtlGetMemberAlias
- EssOtlDeleteMemberAlias

EssOtlSetMemberCommentEx

指定したメンバーに対して、拡張コメントを設定します。

構文

```
ESS_FUNC_M
EssOtlSetMemberCommentEx
 (
hOutline, hMember, pszCommentEx
```

```
    );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	メンバーのハンドル。
pszCommentEx	ESS_STR_T	拡張コメントを含むバッファ。

備考

- 拡張コメントを削除するには、空の文字列または NULL ポインタを使用してこの関数を呼び出します。

戻り値

正常終了の場合は 0 が戻され、コメントが長すぎる場合は、OTLAPI_ERR_MBRCOMMENTEXLEN が戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T   Object;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T   szDbName;
ESS_OBJNAME_T   szFileName;
ESS_HMEMBER_T   hMember;
ESS_STR_T     pszCommentEx;

memset(&Object, '\0', sizeof(Object));
Object.hCtx   = hCtx;
Object.ObjType  = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName  = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);

/************  Set Extended Member Comment *************/
pszCommentEx = "EXTENDED MEMBER COMMENT";

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Variance",&hMember);
}

if (!sts && hMember)
{
```

```
 sts = EssOtlSetMemberCommentEx(hOutline, hMember, pszCommentEx);
}
```

関連トピック

- EssFree
- EssOtlGetMemberCommentEx
- EssOtlOpenOutline

EssOtlSetMemberFormula

指定されたメンバーに対して式を設定します。

構文

```
ESS_FUNC_M
EssOtlSetMemberFormula
 (
hOutline, hMember, pszFormula
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	メンバーのハンドル。
pszFormula	ESS_STR_T	メンバー式を含むバッファ。

備考

- **EssOtlDeleteMemberFormula()**を使用して、メンバー式を削除します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_SHAREDMEMBERFORMULA
- OTLAPI_ERR_MEMBERCALC

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T      sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_HMEMBER_T    hMember;
ESS_STR_T      pszFormula;
```

```
memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName  = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/************  Set Member Formula *************/
pszFormula = "@VAR(Budget, Actual);";
if (!sts)
{
 sts = EssOtlFindMember(hOutline,
    "Variance",&hMember);
}

if (!sts && hMember)
{
 sts = EssOtlSetMemberFormula(hOutline, hMember,
    pszFormula);
}
```

関連トピック

- EssOtlGetMemberFormula
- EssOtlDeleteMemberFormula

EssOtlSetMemberInfo

この関数は、メンバー属性情報を設定します。

構文

```
ESS_FUNC_M
EssOtlSetMemberInfo
 (
hOutline, hMember, pInfo
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hMember	ESS_HMEMBER_T	属性を設定するメンバーのハンドル
pInfo	768 ページの「ESS_MBRINFO_T」	メンバー情報の構造体

備考

属性

- ESS_MBRINFO_T 構造体の 3 つのフィールドは、属性専用です:

データ型	フィールド	説明
ESS_BOOL_T	fAttributed	基本メンバーに関連する属性があるかどうかを示します: ESS_TRUE または ESS_FALSE のいずれか。
ESS_USHORT_T	Attribute. usDataType	属性次元またはゼロレベル(リーフ・ノード)の属性メンバーは、次のいずれかのデータ型になります: ❍ ESS_ATTRMBRDT_BOOL ❍ ESS_ATTRMBRDT_DATETIME ❍ ESS_ATTRMBRDT_DOUBLE ❍ ESS_ATTRMBRDT_STRING 属性次元ではなく、属性メンバーの場合: ❍ ESS_ATTRMBRDT_NONE ❍ ESS_ATTRMBRDT_AUTO **注：** ESS_ATTRMBRDT_AUTO は、メンバーを追加する場合にのみ使用してください。「属性メンバーの追加に関する注意」を参照してください。
ユニオン ESS_BOOL_T ESS_STR_T ESS_DATETIME_T ESS_DOUBLE_T	Attribute.value bData strData dtData dblData	次の属性メンバー値のいずれかになります: ❍ ブール値 ❍ 文字列値 ❍ 日付と時刻の値 ❍ DOUBLE 値

- ESS_MBRINFO_T 構造体の 2 つのフィールドの値は、属性専用です:

データ型	フィールド	説明
ESS_USHORT_T	usCategory	次の次元カテゴリのいずれかになります: ❍ ESS_CAT_ATTRIBUTE ❍ ESS_CAT_ATTRCALC (システム内部でのみ使用)
ESS_USHORT_T	usStorageCategory	次の次元ストレージ・カテゴリのいずれかになります: ❍ ESS_STORECAT_ATTRIBUTE ❍ ESS_STORECAT_ATTRCALC (システム内部でのみ使用)

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME

- OTLAPI_BAD_MEMBER
- OTLAPI_ERR_ADDNAMEUSED
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE
- OTLAPI_ERR_ILLEGALNAME
- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM
- OTLAPI_ERR_SHARENOTLEVEL0

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T   hMemberJan;
ESS_MBRINFO_T   MbrInfo;
ESS_PMBRINFO_T   pMbrInfo = ESS_NULL;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);
```

```
if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMemberJan);
}

if (!sts && hMemberJan)
{
 sts = EssOtlGetMemberInfo(hOutline,
    hMemberJan, &pMbrInfo);
}

if (!sts && pMbrInfo)
{
 pMbrInfo->usConsolidation = ESS_UCALC_SUB;
 pMbrInfo->fTwoPass = ESS_TRUE;
 pMbrInfo->fExpense = ESS_TRUE;
 sts = EssOtlSetMemberInfo(hOutline,
    hMemberJan, pMbrInfo);
}

if (pMbrInfo)
{
 EssOtlFreeStructure(hOutline, count, structId, structPtr);
}
```

関連トピック

- EssOtlGetMemberInfo
- EssOtlFindMember

EssOtlSetMemberSolveOrder

メンバーの解決順を設定します。

構文

```
ESS_FUNC_M EssOtlSetMemberSolveOrder (
hOutline, hMember, cOrder
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	次元ハンドル(入力)。
cOrder	ESS_UCHAR_T	解決順(入力)。0 - 127

備考

- 解決順は集約ストレージ・データベースにのみ適用できます。
- メンバーまたは次元の解決順プロパティは、計算順序を指定します。

- メンバーの解決順は次元の解決順よりも優先されます。解決順は 0 から 127 までにできます。デフォルトは 0 です。
- 解決順が指定されていない式を持たないメンバーは、その次元の解決順を継承します。解決順が指定されていない式を持つメンバーは、ゼロの解決順を持ちます。

戻り値

成功の場合、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T   hOutline = ESS_NULL;
ESS_HMEMBER_T   hMember = ESS_NULL;
ESS_UCHAR_T    ucOrder = 0;

/* code to assign hOutline variable omitted */
/* code to assign hMember variable omitted */
/* code to assign ucOrder variable omitted */


if (hOutline && hMember)
{
 if (ucOrder > 127)
 {
  printf("Solveorder must be less than 128\n");
 {
 else
 {
  sts = EssOtlSetMemberSolveOrder(hOutline, hMember, ucOrder);

  if (sts)
   printf("Error [%ld] returned\n", sts);
  else
   printf("Solve Order: %d\n", ucOrder);
 }
}
else
 printf("Both hOutline and hMember must have values\n");
```

関連トピック

- EssOtlGetMemberSolveOrder
- EssOtlSetDimensionSolveOrder
- EssOtlGetDimensionSolveOrder

EssOtlSetMemberType

入力アウトライン・メンバーのメンバー・タイプを設定します。

構文

```
    ESS_FUNC_M EssOtlSetMemberType(
```

```
hOutline
,
hMember
,
usType
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
hMember	ESS_HMEMBER_T	アウトライン・メンバー・ハンドル
usType	ESS_USHORT_T	アウトライン・メンバーのタイプ: ● ESS_MEMBERTYPE_NUMERIC メンバー・タイプは数値です。 ● ESS_MEMBERTYPE_DATE メンバー・タイプは日付型です。

備考

型を ESS_MEMBERTYPE_SMARTLIST に設定できません。
EssOtlSetMemberTypeToSmartList を使用してください。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestGetSetMemberType()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T             hOutline = ESS_NULL;
    ESS_OBJDEF_T                 Object;
    ESS_HMEMBER_T              hMember;
    ESS_USHORT_T                 usMemberType;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =               hCtx;
    Object.ObjType =           ESS_OBJTYPE_OUTLINE;
    Object.AppName =           szAppName;
    Object.DbName =           szDbName;
    Object.FileName =      szFileName;

    /* Open outline */
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
                ESS_TRUE, &hOutline);

    /* Find a member */
    sts = EssOtlFindMember(hOutline, "Original Price", &hMember);
```

```
/* Get Member Type of an outline that is not member
    type enabled */
/* Get original type */
sts = EssOtlGetMemberType(hOutline, hMember, &usMemberType);
DisplayMemberType(usMemberType); /* a display function */

/* Set type to NUMERIC */
usMemberType = ESS_MEMBERTYPE_NUMERIC;
sts =
EssOtlSetMemberType(hOutline, hMember, usMemberType);

printf("EssOtlSetMemberType sts: %d\n",sts);

/* Set type to SmartList */
usMemberType = ESS_MEMBERTYPE_SMARTLIST;
sts =
EssOtlSetMemberType(hOutline, hMember, usMemberType);

    printf("EssOtlSetMemberType sts: %d\n",sts);

/* Set type to DATE */
usMemberType = ESS_MEMBERTYPE_DATE;
sts =
EssOtlSetMemberType(hOutline, hMember, usMemberType);

    printf("EssOtlSetMemberType sts: %d\n",sts);

/* Get Member Type of an outline that is member
    type enabled */
EnableSmartList(hOutline);

/* Get original type */
sts = EssOtlGetMemberType(hOutline, hMember, &usMemberType);
printf("EssOtlGetMemberType sts: %d\n", sts);
DisplayMemberType(usMemberType);

/* Set type to DATE */
usMemberType = ESS_MEMBERTYPE_DATE;
sts =
EssOtlSetMemberType(hOutline, hMember, usMemberType);

printf("EssOtlSetMemberType sts: %d\n",sts);

sts = EssOtlGetMemberType(hOutline, hMember, &usMemberType);
printf("EssOtlGetMemberType sts: %d\n", sts);
DisplayMemberType(usMemberType);

/* Set type to NUMERIC */
usMemberType = ESS_MEMBERTYPE_NUMERIC;
sts =
EssOtlSetMemberType(hOutline, hMember, usMemberType);

    printf("EssOtlSetMemberType sts: %d\n",sts);

sts = EssOtlGetMemberType(hOutline, hMember, &usMemberType);
printf("EssOtlGetMemberType sts: %d\n", sts);
```

```
    DisplayMemberType(usMemberType);


    /* Clean up */
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlSetMemberTypeToSmartList

入力アウトライン・メンバーを ESS_MEMBERTYPE_SMARTLIST として設定し、入力テキスト・リスト(スマートリスト・オブジェクト)と関連付けます。

構文

```
ESS_FUNC_M EssOtlSetMemberTypeToSmartList(
hOutline, hMember, hSmartList
)
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトライン・ハンドル(編集モードのみ)
hMember	ESS_HMEMBER_T	アウトライン・メンバー・ハンドル
hSmartList	ESS_HSMARTLIST_T	関連付けられるスマートリスト・ハンドル。

戻り値

戻り値:

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

例

```
    void TestSetMemberTypeToSmartList()
{
    ESS_STS_T                    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T             hOutline = ESS_NULL;
    ESS_OBJDEF_T                 Object;
    ESS_OBJECT_TYPES         objType;
    ESS_STR_T                    objName;
    ESS_HOBJECT_T             hObjHandle1, hObjHandle2;
    ESS_HMEMBER_T             hMember;
    ESS_HSMARTLIST_T         hSmartList;
    //ESS_USHORT_T              usVerifyType;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx =              hCtx;
    Object.ObjType =          ESS_OBJTYPE_OUTLINE;
    Object.AppName =          szAppName;
    Object.DbName =          szDbName;
    Object.FileName =     szFileName;

    /* Open outline */
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
                ESS_TRUE, &hOutline);

    /* Find a member */
    sts = EssOtlFindMember(hOutline, "Original Price",
                &hMember);

    /* Get original SmartList association */
    sts = EssOtlGetMemberSmartList(hOutline, hMember,
                &hSmartList);

    /* Set member type to SmartList */
    hSmartList = (ESS_HSMARTLIST_T)hObjHandle1;
    sts =
    EssOtlSetMemberTypeToSmartList(hOutline,
                hMember, hSmartList);


    /* Unlock */
    sts = EssUnlockObject(hCtx, Object.ObjType,
                Object.AppName, Object.DbName, Object.FileName);

    /* Close outline */
    sts = EssOtlCloseOutline(hOutline);
}
```

関連トピック

- EssOtlGetMemberSmartList
- EssOtlCreateObject
- EssOtlDeleteObject
- EssOtlGetSmartListInfo
- EssOtlFindObject
- EssOtlFreeObjectArray
- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetMemberType
- EssOtlGetObjectReferenceCount
- EssOtlGetObjectReferences
- EssOtlImportExportObject
- EssOtlListObjects
- EssOtlQueryObjects
- EssOtlSetMemberType
- EssOtlSetMemberTypeToSmartList

EssOtlSetOriginalMember

拡張された共有メンバーとしてメンバーを設定します。

構文

```
ESS_FUNC_M EssOtlSetOriginalMember (
hOutline, hMember, pszOriginalMbr
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hMember	ESS_HMEMBER_T	メンバー名(入力)。このメンバーは拡張された共有メンバーとして設定されます。
pszOriginalMbr	ESS_STR_T	共有を行う元のメンバー名(入力)。

備考

- hMember がまだ共有されていない場合には、拡張された共有メンバーとしてマークされます。
- すべてのメンバー名が一意であるアウトラインでこの関数を使用した場合は、無効です。
- この関数を呼び出す前に、EssOtlOpenOutline を呼び出して編集モードでアウトラインを開いてください。
- 次の階層の場合、[Diet].[100-10]に対応するメンバー・ハンドル(hMember)をこの関数に渡し、元のメンバー(pszOriginalMbr)を[200].[100-10]とすると、[Diet].[100-10]は、[200].[100-10]の拡張された共有メンバーとなります。

```
          100
    100-10
  200
    100-10 (duplicate)
  Diet
    100-10 (shared with [200.100-10])
```

戻り値

正常終了の場合は 0 が戻され、それ以外はエラーが戻されます。

例

```
    ESS_FUNC_M ESS_SetOrigMember()
{

ESS_STS_T   sts = 0;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T  szAppName;
ESS_DBNAME_T  szDbName;
ESS_OBJNAME_T  szFileName;
ESS_HMEMBER_T  hMember = ESS_NULL;



memset(&Object, '\0', sizeof(Object));
Object.hCtx =  hCtx;
Object.ObjType =  ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName =  szAppName;
Object.DbName =  szDbName;
Object.FileName =  szFileName;


sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);


if (!sts)
{
 sts = EssOtlFindMember(hOutline, "[Diet].[100-10]", &hMember);
}

if (!sts && hMember)
{
 sts = EssOtlSetOriginalMember (hOutline, hMember, "[100].[100-10]");
}

return sts;
}
```

関連トピック

- EssOtlGetOriginalMember

EssOtlSetOutlineInfo

アウトライン情報を設定します。

構文

```
ESS_FUNC_M
EssOtlSetOutlineInfo
 (
hOutline, pInfo
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pInfo	775 ページの「ESS_OUTLINEINFO_T」	アウトライン情報を保管するために、呼出し元で割り当てる構造体へのポインタ。

備考

- 775 ページの「ESS_OUTLINEINFO_T」構造体の一部のフィールドのみを使用して情報を設定します。詳細は構造体の説明を参照してください。
- ESS_OUTLINEINFO_T 構造体の fCaseSensitive フラグが、ESS_TRUE から ESS_FALSE に変更され、そのためにメンバー名が重複してしまった場合、この呼出しは失敗します。アウトラインが重複メンバー名アウトラインの場合、この関数のかわりに EssOtlSetOutlineInfoEx を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外はエラーが戻されます。

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T  sts = 0;
ESS_FALSE = 0;
ESS_TRUE  = 1;
ESS_POUTLINEINFO_T pInfo = ESS_NULL;
ESS_HOUTLINE_T  hOutline;
ESS_OBJDEF_T  Object;
ESS_APPNAME_T     szAppName;
ESS_DBNAME_T     szDbName;
ESS_OBJNAME_T     szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
```

```
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
   &hOutline);

if (!sts)
{
 sts = EssOtlGetOutlineInfo(hOutline, &pInfo);
}

if (!sts && Info)
{
 pInfo->fCaseSensitive = ESS_FALSE;
 sts = EssOtlSetOutlineInfo(hOutline, pInfo);
}

if (pInfo)
{
 EssFree(hInst, pInfo);
}
```

関連トピック

- EssOtlGetOutlineInfo
- EssOtlSetOutlineInfoEx

EssOtlSetOutlineInfoEx

一意のメンバー名のアウトラインを、名前の重複が許可されるアウトラインに変換します。

pInfo ->fNonUniqueName が TRUE に設定されている場合、この関数は一意のメンバー名のアウトラインを重複メンバー名を許可するアウトラインに変換します。重複メンバー名を許可するアウトラインを、一意のメンバー名のアウトラインに変換して戻すことはできません。

構文

```
ESS_FUNC_M
EssOtlSetOutlineInfoEx
 (
hOutline, pInfo
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。

パラメータ	データ型	説明
pInfo	775 ページの「ESS_OUTLINEINFO_T」	アウトライン情報を保管するために、呼出し元で割り当てる構造体へのポインタ(入力)。

備考

775 ページの「ESS_OUTLINEINFO_T」構造体の一部のフィールドのみを使用して情報を設定します。詳細は構造体の説明を参照してください。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合については、753 ページの「C のアウトライン API のエラー戻り値」を参照してください。

例

```
    void SetOutlineInfoEx()
{
ESS_STS_T           sts = 0;
ESS_POUTLINEINFO_T   pInfo = ESS_NULL;
ESS_HOUTLINE_T       hOutline;
ESS_OBJDEF_T        Object;
ESS_APPNAME_T       szAppName;
ESS_DBNAME_T        szDbName;
ESS_OBJNAME_T        szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
 ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlGetOutlineInfo(hOutline, &pInfo);
}


if (!sts && pInfo)
{
 pInfo->fNonUniqueName = ESS_TRUE;
 sts =
    EssOtlSetOutlineInfoEx
    (hOutline, pInfo);
}

if (!sts)
{
 sts = EssOtlWriteOutline(hOutline, &Object);
```

```
}


if (!sts)
{
 sts = EssOtlRestructure(hCtx, ESS_DOR_ALLDATA);
}



if (pInfo)
{
 EssFree(hInst, pInfo);
}

}
```

関連トピック

- EssOtlGetOutlineInfo

EssOtlSetQueryHint

既存のクエリー・ヒントのコンテンツ(pMemberArray)を変更します。リリース 9.3 以上の集約ストレージ・データベースにのみ適用されます。

構文

```
ESS_FUNC_M EssOtlSetQueryHint (
hOutline, hintNum, numMembers, pMemberArray
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル(入力)。
hintNum	ESS_SHORT_T	クエリー・ヒント番号(入力)。
numMembers	ESS_SHORT_T	提供された配列のメンバーの数 – 通常は、アウトライン(入力)の実次元の数。
pMemberArray	ESS_PHMEMBER_T	ヒント用のメンバーの配列。通常、配列には、実次元当たり 1 つのメンバーを持ち、NULL がヒントの一部でない次元に使用されます。この配列は割り当てる必要があります。(入力)

備考

- 競合が発生したときに、レベル使用制約がクエリー・ヒントより優先されます(SetAggLevelUsage を参照)。
- ヒントには、動的メンバー、ラベルのみメンバー、または共有メンバーは含まれない場合があります。
- アウトラインが変化すると、ヒントは無効になる場合があります。無効なヒントにより警告メッセージが発生します。

- 共通クエリーのプロファイルについて Essbase に通知することにより、ヒントは標準ビュー選択に影響を及ぼすことができます。
- ヒントは、MDX タプルとして書かれており、指定された各次元からのメンバーを 1 つのみ含みます。
- クエリー・ヒントで使用される各メンバーは、代表的なメンバーと考えられます。Essbase サーバーは、「このメンバーまたは類似した集約レベルのメンバー」として、代表的なメンバーを解釈します。たとえば、Sample Basic で `(Qtr1, Sales, 100, East, Actual)`のクエリー・ヒントを使用すれば、四半期ごとに、レベル 1 マーケットでのレベル 1 製品の実績利益率のメジャーが、クエリーの一般タイプになります。
- ある指定した次元について、Essbase は、代表的なメンバーの省略を、次元からのメンバーをクエリーで使用できると解釈します。たとえば、Sample Basic で`(Sales, 100, East)`のクエリー・ヒントを使用すると、省略されている Year および Scenario 次元に関係なく、レベル 1 マーケットでのレベル 1 製品の利益率のメジャーが、クエリーの一般タイプになります。ヒント`(Sales, 100, East)`は、`(NULL, Sales, 100, East, NULL)`と同一のものとして扱われます。

戻り値

成功の場合、0 が戻されます。

例

```
    ESS_STS_T     sts = ESS_STS_NOERR;
ESS_HOUTLINE_T  hOutline = ESS_NULL;
ESS_SHORT_T    nmHints = 0;
ESS_SHORT_T    i, j, hintNum;
ESS_HMEMBER_T   hMember[10]; /* (nm real dimensions) < 10 */

/* clear array just to be safe */
memset(hMember, 0x00, 10*sizeof(ESS_HMEMBER_T));

/* Code to assign hOutline variable omitted */
/* Code to assign hintNum variable omitted */

sts = EssOtlGetNumQueryHints(hOutline, &nmHints);
if (sts) return sts;  /* error out */

if (hintNum <= nmHints)
{
 sts = EssOtlGetQueryHint(hOutline, hintNum, 10, hMember);
 if (sts) return sts;  /* error out */

 for (j = 0; j < 10; j++)
 {
  /* Code to inspect and change hMember[j] omitted */
 }

 sts = EssOtlSetQueryHint(hOutline, hintNum, 10, hMember);
 if (sts) return sts;  /* error out */
 printf("Query-Hint number: (%d) updated\n", hintNum);
}
```

```
else
{
 printf("Query-Hint number: (%d) does not exist\n", hintNum);
}
```

関連トピック

- EssOtlAddQueryHint
- EssOtlGetQueryHint
- EssOtlGetNumQueryHints
- EssOtlGetQueryHintSize
- EssOtlDeleteQueryHint

EssOtlSetUserAttribute

メンバーに対するユーザー定義属性を設定します。

構文

```
ESS_FUNC_M
EssOtlSetUserAttribute
 (
hOutline, hMember, pszString
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hMember	ESS_HMEMBER_T	ユーザー定義属性を設定するメンバーのハンドル。
pszString	ESS_STR_T	設定するユーザー定義属性。

備考

- 呼出し元はメンバーに任意の数のユーザー定義属性を設定できます。一意に渡された文字列が各属性を定義し、ユーザー名と同様の表記規則に従います。**EssOtlGetUserAttributes** を参照してください。
- 共有メンバーに対してユーザー属性を設定しようとすると、エラーが発生します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_USERATTR
- OTLAPI_ERR_SHAREUDA

例

```
    #include <essapi.h>
#include <essotl.h>
```

```
ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;
ESS_HMEMBER_T    hMember;
ESS_STR_T     AttributeList;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName  = szAppName;
Object.DbName   = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/***********  Set User Attributes ************/
AttributeList = "Read Write";

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Jan",
    &hMember);
}

if (!sts && hMember)
{
 sts = EssOtlSetUserAttribute(hOutline, hMember,
    AttributeList);
}
```

関連トピック

- EssOtlDeleteUserAttribute
- EssOtlGetUserAttributes

EssOtlSortChildren

アウトライン・メンバーの子をソートします。

構文

```
ESS_FUNC_M
EssOtlSortChildren
 (
hOutline, hParent, usType, fpCompare, pUserData
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hParent	ESS_HMEMBER_T	ソートする子の親のハンドル。ESS_NULL が指定されている場合は、次元がソートされます。
usType	ESS_USHORT_T	ソート・タイプ。次のいずれかを指定できます: ● ESS_SORT_ASCENDING ● ESS_SORT_DESCENDING ● ESS_SORT_USERDEFINED
fpCompare	ESS_POTLSORTFUNC_T	ユーザーが定義した比較関数へのポインタ。usType パラメータが ESS_SORT_USERDEFINED の場合にのみ使用されます。次のように定義された関数へのポインタです: (ESS_INTFUNC_M Compare ESS_HMEMBER_T mbr1, ESS_HMEMBER_T mbr2, ESS_PVOID_T pUserData); この関数は 2 つのメンバーのハンドルを受け入れ、次の値を戻します: ● mbr1 が mbr2 より前にある場合は< 0。 ● mbr1 が mbr2 と等しい場合は= 0。 ● mbr1 が mbr2 より後ろにある場合は> 0。
pUserData	ESS_PVOID_T	ユーザーが指定した任意のデータへのポインタ。usType パラメータが ESS_SORT_USERDEFINED の場合にのみ使用されます。比較関数を呼び出すたびに、このパラメータの値が比較関数に渡されます。

備考

コールバック関数中は、アウトラインを変更する可能性のあるアウトライン関数は呼び出さないでください。**EssOtlGetMemberInfo()**、**EssOtlGetMemberFormula()**、および **EssOtlGetMemberAlias()**のみ呼び出すことができます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_SORTTYPE
- OTLAPI_BAD_SORTCOMPAREFUNC
- OTLAPI_SORT_TOOMANY

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_HMEMBER_T   hMeasures;
```

```
FARPROC      pfnSort;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

if (!sts)
{
 sts = EssOtlFindMember(hOutline, "Measures",
    &hMeasures);
}

if (!sts)
{
 sts = EssOtlSortChildren(hOutline, hMeasures,
    ESS_SORT_USERDEFINED,
    (ESS_POTLSORTFUNC_T)pfnSort,
    (ESS_PVOID_T)hOutline);
}

/*********************************************/
int  ESS_INTFUNCT_M SortCompare (
  ESS_HMEMBER_T hMember1,
  ESS_HMEMBER_T hMember2,
  ESS_PVOID_T pData)
{
  int nRet = 0;
  int nLen1;
  int nLen2;
  ESS_STS_T sts = 0;
  ESS_PMBRINFO_T pMbrInfo1 = ESS_NULL;
  ESS_PMBRINFO_T pMbrInfo2 = ESS_NULL;
  ESS_HOUTLINE_T hOutline =
(ESS_HOUTLINE_T)pData;

 sts = EssOtlGetMemberInfo(hOutline, hMember1,
    &pMbrInfo1);

 if (!sts && pMbrInfo1)
 sts = EssOtlGetMemberInfo(hOutline,
    hMember2, &pMbrInfo2);

 if (!sts && pMbrInfo2)
 {
```

```
  nLen1 = strlen(pMbrInfo1->szMember);
  nLen2 = strlen(pMbrInfo2->szMember);
  if (nLen1 < nLen2)
    nRet = -1;
  else if (nLen1 > nLen2)
    nRet = 1;
 }

 if (pMbrInfo1)
 {
 EssFree(hInst, pMbrInfo1);
 }

 if (pMbrInfo2)
 {
 EssFree(hInst, pMbrInfo2);
 }
 return (nRet);
}
```

関連トピック

- EssOtlFindMember

EssOtlVaryingAssociateAttribute

属性メンバーを基本メンバーに関連付けます。その際、指定した妥当性セットで指定された関連付けの妥当性が適用されます。

構文

```
ESS_FUNC_M EssOtlVaryingAssociateAttribute (
hOutline, hBaseMember, hAttrMember, mode, pValiditySet
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hBaseMember	ESS_HMEMBER_T	基本メンバーのハンドル
hAttrMember	ESS_HMEMBER_T	属性メンバーのハンドル
mode	ESS_INT_T	関連付けモード。可能な値: ● MODE_OVERWRITE ● MODE_NOOVERWRITE
pValiditySet	779 ページの「ESS_VALIDITYSET_T」	関連付けが有効な個別の次元を定義する妥当性セットへのポインタ

備考

- 完全な範囲が指定された場合、関連付けは指定された内容で行われます。

- 関連付けモードにより、完全な範囲ではなく、開始タプルのみ指定されている制限のない状況の処理方法が決まります。(説明用の例では、関連付けられた Ounces 属性メンバーが、関連付け前に製品 100-10 に関連付けられたと想定しています):

```
        Jan  Feb  Mar  Apr  May
12   12   16   16   20
```

  - MODE_OVERWRITE - 関連付けは指定したメンバー以降の全メンバーに対して行われます。例: 3 月から 12 を関連付け。

    結果: 3 月以降のすべてのメンバーが 12 に関連付けられます。矛盾している関連付けは上書きされます:

```
           Jan  Feb  Mar  Apr  May
12   12
           12   12   12
```

  - MODE_NOOVERWRITE - 関連付けは指定したタプルから開始し、既存の関連付け属性メンバーと異なるまで続行します。例: 3 月から 12 を関連付け。

    結果: 3 月と 4 月は同じ属性だったため、関連付けは属性が変わる 5 月の前まで続行します:

```
           Jan  Feb  Mar  Apr  May
12   12
           12   12
             20
```

戻り値

成功の場合、0 が戻されます。

例

```
    void TestEssOtlVaryingAssociateAttribute()
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T            hOutline = ESS_NULL;
    ESS_OBJDEF_T            Object;
    ESS_HMEMBER_T            hBaseMbr, hAttrMbr, hBaseDim, hAttrDim, hIndDim =
ESS_NULL;
    ESS_INT_T            mode;
    ESS_VALIDITYSET_T        ValiditySet;
    ESS_HMEMBER_T            IndDimsArray[2];
    ESS_STR_T            IndepMbrsArray[4];
    ESS_INT32_T            countOfIndDims;
    ESS_USHORT_T            usValiditySetType;
    ESS_UCHAR_T            pucIndependentTypes[2];
```

```
    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;

    printf("\n");
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
    printf("EssOtlOpenOutline sts: %ld\n",sts);

    /* Disassociate base dimension from attribute dimension before test.*/
    printf("\nDisassociate base dimension from attribute dimension before test:");
    sts = EssOtlFindMember(hOutline, "Entities", &hBaseDim);
    printf("\nEssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "Type", &hAttrDim);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlDisassociateAttributeDimension(hOutline, hBaseDim, hAttrDim);
    printf("EssOtlDisassociateAttributeDimension sts: %d\n", sts);

    /* Get handle for base member*/
    printf("\nGet handle for base member:\n");
    sts = EssOtlFindMember(hOutline, "Doe,Jane", &hBaseMbr);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Get handle for indep dimensions*/
    printf("\nGet handle for indep dimensions:\n");
    sts = EssOtlFindMember(hOutline, "Time Periods", &IndDimsArray[0]);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "Years", &IndDimsArray[1]);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Associate the dimension Entities and Type*/
    printf("\nAssociate the dimensions:\n");
    countOfIndDims = 2;
    pucIndependentTypes[0] = ESS_ASSOCIATE_TYPE_DISCRETE;
    pucIndependentTypes[1] = ESS_ASSOCIATE_TYPE_CONTINUOUS;
    sts = EssOtlVaryingAssociateAttributeDimension(hOutline, hBaseDim, hAttrDim,
countOfIndDims, IndDimsArray, pucIndependentTypes);
    printf("EssOtlVaryingAssociateAttributeDimension sts: %d\n", sts);

    /* Initial valid case with ValiditySetType of member handles*/
    printf("\n*** Initial valid case with ValiditySetType of member handles ***\n");
    printf("\nGet handle for attribute member:\n");
    sts = EssOtlFindMember(hOutline, "Regular", &hAttrMbr);
    printf("EssOtlFindMember sts: %d\n", sts);

    sts = EssOtlFindMember(hOutline, "Jan", &hIndepMbrsArray[0]);
    sts = EssOtlFindMember(hOutline, "FY03", &hIndepMbrsArray[1]);
    sts = EssOtlFindMember(hOutline, "Jan", &hIndepMbrsArray[2]);
    sts = EssOtlFindMember(hOutline, "FY06", &hIndepMbrsArray[3]);

    memset(&ValiditySet, '\0', sizeof(ValiditySet));
    ValiditySet.countOfIndepDims = 2;
    ValiditySet.usValiditySetType = ESS_VALIDITYSET_TYPE_MBRHDLS;
    ValiditySet.countOfIndepRanges = 1;
    ValiditySet.pIndepMbrs = hIndepMbrsArray;
```

```
    printf("\nBefore association:");
    usValiditySetType =  ESS_VALIDITYSET_TYPE_MBRHDLS;
    DisplayVaryingAttributes(hOutline, hBaseMbr, hAttrDim, ESS_NULL,
usValiditySetType);

    mode = ESS_ASSOCIATE_MODE_NOOVERWRITE;
    sts = EssOtlVaryingAssociateAttribute(hOutline, hBaseMbr, hAttrMbr, mode,
&ValiditySet);
    printf("EssOtlVaryingAssociateAttribute sts: %d\n", sts);

    /* Restructure and save outline */
    SaveOutline(hOutline);

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);

    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlQueryVaryingAttributes
- EssOtlVaryingAssociateAttributeDimension
- EssOtlVaryingDisassociateAttribute
- EssOtlVaryingGetAssociatedAttributes
- EssOtlVaryingGetAttributeIndepDims

EssOtlVaryingAssociateAttributeDimension

基本次元を属性次元と関連付け、可変属性を持つ基本次元として定義します。メンバー属性の関連付けは、pucIndependentTypes で指定されたタイプの、指定した独立次元のレベル-0 メンバーに応じて異なります。連続独立次元は最後に指定する必要があります。

構文

```
     ESS_FUNC_M EssOtlVaryingAssociateAttributeDimension (
     hOutline, hBaseDim, hAttrDim, countOfIndepDims, *pIndepDims,
*pucIndependentTypes
     );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hBaseDim	ESS_HMEMBER_T	基本次元ハンドル
hAttrDim	ESS_HMEMBER_T	属性次元ハンドル

パラメータ	データ型	説明
countOfIndepDims	ESS_INT32_T	可変属性を制御する独立次元の数
*pIndepDims	ESS_HMEMBER_T	独立次元についてのメンバー・ハンドルの配列へのポインタ
*pucIndependentTypes	ESS_UCHAR_T	*pIndepDims に含まれる独立タイプの配列へのポインタ。 次の独立したタイプをサポートします: ● ESS_ASSOCIATE_TYPE_DISCRETE ● ESS_ASSOCIATE_TYPE_CONTINUOUS

戻り値

成功の場合、0 が戻されます。

例

```
    void TestEssOtlVaryingAssociateAttributeDimenion()
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T    hOutline = ESS_NULL;
    ESS_OBJDEF_T    Object;
    ESS_HMEMBER_T    hBaseDim = ESS_NULL;
    ESS_HMEMBER_T    IndDimsArray[2];
    ESS_HMEMBER_T    hAttrDimArray[9];
    ESS_INT32_T    countOfIndDims;
    ESS_UCHAR_T    pucIndependentTypes[2];
    int        i;

    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;

    printf("\n");
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
    printf("EssOtlOpenOutline sts: %ld\n",sts);

    SetupTest(hOutline);

    /* Assign independent dimension array*/
    sts = EssOtlFindMember(hOutline, "Time Periods", &IndDimsArray[0]);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "Years", &IndDimsArray[1]);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Get handles to base and attribute dimensions for test.*/
    sts = EssOtlFindMember(hOutline, "Entities", &hBaseDim);
    printf("\nEssOtlFindMember sts: %d\n", sts);

    sts = EssOtlFindMember(hOutline, "Type", &hAttrDimArray[0]);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "FT/PT", &hAttrDimArray[1]);
    printf("EssOtlFindMember sts: %d\n", sts);
```

```
    /* Disassociate current association before tests.*/
    for (i = 0; i < 2; i++)
    {
        sts = EssOtlDisassociateAttributeDimension(hOutline, hBaseDim,
hAttrDimArray[i]);
        printf("EssOtlDisassociateAttributeDimension sts: %d\n", sts);
    }

    /* Associate the dimension Entities to Type*/
    printf("\nValid case: Associate the dimension Entities and Type:\n");
    countOfIndDims = 2;
    pucIndependentTypes[0] = ESS_ASSOCIATE_TYPE_DISCRETE;
    pucIndependentTypes[1] = ESS_ASSOCIATE_TYPE_CONTINUOUS;
    sts = EssOtlVaryingAssociateAttributeDimension(hOutline, hBaseDim,
hAttrDimArray[0], countOfIndDims, IndDimsArray, pucIndependentTypes);
    printf("EssOtlVaryingAssociateAttributeDimension sts: %d\n", sts);

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);

    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlQueryVaryingAttributes
- EssOtlVaryingAssociateAttribute
- EssOtlVaryingDisassociateAttribute
- EssOtlVaryingGetAssociatedAttributes
- EssOtlVaryingGetAttributeIndepDims

EssOtlVaryingDisassociateAttribute

指定した基本メンバーから特定の属性次元の属性メンバーの関連付けを解除します。指定した妥当セットは、関連付けの解除場所を指定します。

構文

```
ESS_FUNC_M EssOtlVaryingDisassociateAttribute (
hOutline, hBaseMember, hAttrDim, mode, pValiditySet
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hBaseMember	ESS_HMEMBER_T	基本メンバーのハンドル
hAttrDim	ESS_HMEMBER_T	属性メンバーのハンドル

パラメータ	データ型	説明
mode	ESS_INT_T	関連付けモード。可能な値: ● MODE_OVERWRITE ● MODE_NOOVERWRITE ● MODE_EXTEND
pValiditySet	779 ページの「ESS_VALIDITYSET_T」	関連付けの解除が実行される独立したメンバーのセットへのポインタ

備考

- この関数は、指定された基本メンバーから属性の関連付けを削除します。
- 完全な範囲が指定された場合、関連付けの解除は指定された内容で行われます。
- 関連付けモードにより、完全な範囲ではなく、開始タプルのみ指定された場合の処理方法が決まります。(説明用の例では、関連付けられた Ounces 属性メンバーが、関連付け解除前に製品 100-10 に関連付けられたと想定しています):

```
          Jan  Feb  Mar  Apr  May
12   12   16   16   12
```

  - MODE_OVERWRITE: 関連付けの解除は指定したメンバー以降の全メンバーに対して行われます。例: 2 月から関連付け解除。

    結果: 2 月以降のすべてのメンバーの関連付けが解除されます:

```
              Jan  Feb  Mar  Apr  May
12
```

  - MODE_NOOVERWRITE: 関連付けの解除は指定したタプルから開始し、既存の関連付け属性メンバーと異なるまで続行します。例:3 月から関連付け解除を開始。

    結果: 3 月にも属性 16 があるため、4 月と 3 月の関連付けが解除されます:

```
              Jan  Feb  Mar  Apr  May
12   12           12
```

  - MODE_EXTEND: MODE_NOOVERWRITE に似ていますが、開始タプルの直前の関連付けが関連付け解除メンバー上に拡張される点が異なります。例: 3 月から関連付け解除が開始

    結果: 3 月と 4 月の関連付けが解除され、前の月である 2 月の関連付けが適用されます。

```
              Jan  Feb  Mar  Apr  May
12   12
              12   12
                12
```

戻り値

成功の場合、0 が戻されます。

例

```
    void TestEssOtlVaryingDisassociateAttribute()
{
    ESS_STS_T    sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T    hOutline = ESS_NULL;
    ESS_OBJDEF_T    Object;
    ESS_HMEMBER_T    hBaseMbr, hAttrMbr, hBaseDim, hAttrDim, hIndDim = ESS_NULL;
    ESS_INT_T    mode;
    ESS_VALIDITYSET_T    ValiditySet;
    ESS_HMEMBER_T    IndDimsArray[2];
    ESS_HMEMBER_T    hIndepMbrsArray[4];
    ESS_STR_T    IndepMbrsArray[4];
    ESS_INT32_T    countOfIndDims;
    ESS_USHORT_T    usValiditySetType;
    ESS_UCHAR_T    pucIndependentTypes[2];


    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;

    printf("\n");
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
    printf("EssOtlOpenOutline sts: %ld\n",sts);

    /* Disassociate base dimension from attribute dimension before test.*/
    printf("\nDisassociate base dimension from attribute dimension before test:");
    sts = EssOtlFindMember(hOutline, "Entities", &hBaseDim);
    printf("\nEssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "Type", &hAttrDim);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlDisassociateAttributeDimension(hOutline, hBaseDim, hAttrDim);
    printf("EssOtlDisassociateAttributeDimension sts: %d\n", sts);

    /* Get handle for base member*/
    printf("\nGet handle for base member:\n");
    sts = EssOtlFindMember(hOutline, "Doe,Jane", &hBaseMbr);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Get handle for indep dimensions*/
    printf("\nGet handle for indep dimensions:\n");
    sts = EssOtlFindMember(hOutline, "Time Periods", &IndDimsArray[0]);
    printf("EssOtlFindMember sts: %d\n", sts);
    sts = EssOtlFindMember(hOutline, "Years", &IndDimsArray[1]);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Associate the dimension Entities and Type*/
    printf("\nAssociate the dimensions:\n");
    countOfIndDims = 2;
```

```
    pucIndependentTypes[0] = ESS_ASSOCIATE_TYPE_DISCRETE;
    pucIndependentTypes[1] = ESS_ASSOCIATE_TYPE_CONTINUOUS;
    sts = EssOtlVaryingAssociateAttributeDimension(hOutline, hBaseDim, hAttrDim,
countOfIndDims, IndDimsArray, pucIndependentTypes);
    printf("EssOtlVaryingAssociateAttributeDimension sts: %d\n", sts);



    /* Initial valid case with ValiditySetType of member handles*/
    printf("\n*** Initial valid case with ValiditySetType of member handles ***\n");
    printf("\nGet handle for attribute member:\n");
    sts = EssOtlFindMember(hOutline, "Regular", &hAttrMbr);
    printf("EssOtlFindMember sts: %d\n", sts);

    sts = EssOtlFindMember(hOutline, "Jan", &hIndepMbrsArray[0]);
    sts = EssOtlFindMember(hOutline, "FY03", &hIndepMbrsArray[1]);
    sts = EssOtlFindMember(hOutline, "Jan", &hIndepMbrsArray[2]);
    sts = EssOtlFindMember(hOutline, "FY06", &hIndepMbrsArray[3]);

    memset(&ValiditySet, '\0', sizeof(ValiditySet));
    ValiditySet.countOfIndepDims = 2;
    ValiditySet.usValiditySetType = ESS_VALIDITYSET_TYPE_MBRHDLS;
    ValiditySet.countOfIndepRanges = 1;
    ValiditySet.pIndepMbrs = hIndepMbrsArray;

    printf("\nBefore association:");
    usValiditySetType =  ESS_VALIDITYSET_TYPE_MBRHDLS;
    DisplayVaryingAttributes(hOutline, hBaseMbr, hAttrDim, ESS_NULL,
usValiditySetType);

    mode = ESS_ASSOCIATE_MODE_NOOVERWRITE;
    sts = EssOtlVaryingAssociateAttribute(hOutline, hBaseMbr, hAttrMbr, mode,
&ValiditySet);
    printf("EssOtlVaryingAssociateAttribute sts: %d\n", sts);

    /* Disassociation */
    IndepMbrsArray[0]= "";
    IndepMbrsArray[1]= "";
    IndepMbrsArray[2]= "";
    IndepMbrsArray[3]= "";
    memset(&ValiditySet, '\0', sizeof(ValiditySet));
    ValiditySet.countOfIndepDims = 2;
    ValiditySet.usValiditySetType = ESS_VALIDITYSET_TYPE_MBRNAMS;
    ValiditySet.countOfIndepRanges = 1;
    ValiditySet.pIndepMbrs = IndepMbrsArray;
    mode = ESS_DISASSOCIATE_MODE_NOOVERWRITE;
    sts = EssOtlVaryingDisassociateAttribute(hOutline, hBaseMbr, hAttrDim, mode,
&ValiditySet);
    printf("EssOtlVaryingDisassociateAttribute sts: %d\n", sts);

    /* Restructure and save outline */
    SaveOutline(hOutline);

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);
```

```
    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlQueryVaryingAttributes
- EssOtlVaryingAssociateAttribute
- EssOtlVaryingAssociateAttributeDimension
- EssOtlVaryingGetAssociatedAttributes
- EssOtlVaryingGetAttributeIndepDims

EssOtlVaryingGetAssociatedAttributes

指定した基本メンバーと関連付けられた指定の属性次元に属性メンバーを戻します。この場合、関連付けの妥当性には指定したパースペクティブのタプルが 1 つ以上含まれています。

該当している各属性メンバーに対して、基本メンバーへの関連付けの完全な妥当性セットをオプションで戻すことができます(pppValiditySets に NULL 以外の値を指定することによって可能です)。

pphMembers にはメンバー・ハンドルの配列が含まれ、pppValiditySets には妥当性セット・ポインタの配列が含まれます。

usValiditySetType を使用して、必要な妥当性セットのタイプを指定します。

パースペクティブは、独立したメンバーを個別に指定する必要がある点に注意してください。

パースペクティブが指定されていない場合、またはパースペクティブで独立したメンバーの NULL セットが指定されている場合、独立したメンバーの任意の組合せに対して存在するすべての関連付けがルーチンによって検討されます。この場合、戻された妥当性セットには、個別の独立したメンバーの範囲が含まれることがあり、クライアントはこれらを分割する必要があります。

構文

```
     ESS_FUNC_M EssOtlVaryingGetAssociatedAttributes (
     hOutline, hBaseMember, hAttrDim, pPerspective, pusCount,
pphMembers, usValiditySetType, **pppValiditySets
     );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hBaseMember	ESS_HMEMBER_T	基本次元メンバーのハンドル
hAttrDim	ESS_HMEMBER_T	属性メンバーのハンドル
pPerspective	777 ページの「ESS_PERSPECTIVE_T」	クライアントまたはサーバーで関連付けをクエリーする際に使用される、独立したメンバーの集合へのポインタ

パラメータ	データ型	説明
pusCount	ESS_PUSHORT_T	戻される可変属性メンバーの数へのポインタ
pphMembers	ESS_PPHMEMBER_T	属性メンバーのハンドルの配列へのポインタ
usValiditySetType	ESS_USHORT_T	独立したメンバーに割り当てられた妥当性セットのタイプ: ● ESS_VALIDITYSET_TYPE_MBRHDLS - メンバー・ハンドルの XRange。たとえば、Mar 2003-Feb 2004 は 2003 年の 10 か月(3 月から開始)と、2004 年の最初の 2 か月(2 月に終了)から構成されています。 ● ESS_VALIDITYSET_TYPE_MBRNAMS - メンバー名の XRange
**pppValiditySets	779 ページの「ESS_VALIDITYSET_T」	関連付けが TRUE の独立したメンバーの集合

戻り値

正常終了の場合は、0 が戻されます。

例

```
    void DisplayVaryingAttributes(ESS_HOUTLINE_T hOutline, ESS_HMEMBER_T hBaseMbr,
ESS_HMEMBER_T hAttrDim,
                                                ESS_PERSPECTIVE_T *pPerspective,
ESS_USHORT_T usValiditySetType)
{
    ESS_STS_T                sts = ESS_STS_NOERR;
    ESS_USHORT_T            Count, i, j, totalIndMbrs;
    ESS_PHMEMBER_T            phAttrMbrs;
    ESS_PVALIDITYSET_T  *ppValiditySets;
    ESS_PMBRINFO_T            pMemberInfo1, pMemberInfo2;

    sts = EssOtlVaryingGetAssociatedAttributes(hOutline,
            hBaseMbr,
            hAttrDim,
            pPerspective,
            &Count,
            &phAttrMbrs,
            usValiditySetType,
            &ppValiditySets);
    printf("\nEssOtlVaryingGetAssociatedAttributes sts: %d", sts);
    if(!sts)
    {
        if(Count)
        {
            for (i = 0; i < Count ;++i)
            {
                sts = EssOtlGetMemberInfo(hOutline, phAttrMbrs[i], &pMemberInfo1);
                printf("\n\t%s", pMemberInfo1->szMember);
                EssOtlFreeStructure(hOutline, ESS_DT_STRUCT_MBRINFO, 1,
pMemberInfo1);

                if(ppValiditySets[i])
                {
```

```
                    totalIndMbrs = (ESS_SHORT_T)(ppValiditySets[i]-
>countOfIndepRanges) * 4;
                    printf("\n\t\tValidity Type: %d - ", ppValiditySets[i]-
>usValiditySetType);
                    switch(ppValiditySets[i]->usValiditySetType)
                    {
                        case ESS_VALIDITYSET_TYPE_MBRHDLS:
                            printf("Member Handles");

                            for(j = 0; j < totalIndMbrs; j++)
                            {
                                if(j >= 3)
                                    if(j%4 == 0)
                                        printf("\n");
                               sts = EssOtlGetMemberInfo(hOutline, ppValiditySets[i]-
>pIndepMbrs[j], &pMemberInfo2);
                                printf("\n\t\tValidity independent member: %s",
pMemberInfo2->szMember);
                                EssOtlFreeStructure(hOutline, ESS_DT_STRUCT_MBRINFO,
1, pMemberInfo2);
                            }
                            break;
                        case ESS_VALIDITYSET_TYPE_MBRNAMS:
                            printf("Member Names");

                            for(j = 0; j < totalIndMbrs; j++)
                            {
                                if(j >= 3)
                                    if(j%4 == 0)
                                        printf("\n");
                                printf("\n\t\tValidity independent member: %s",
ppValiditySets[i]->pIndepMbrs[j]);
                            }
                            break;
                        default:
                            printf("Unrecognized");
                    }
                    printf("\n\t\tValidity count of Indep Dims: %d",
ppValiditySets[i]->countOfIndepDims);
                    printf("\n\t\tValidity count of Indep Ranges: %d",
ppValiditySets[i]->countOfIndepRanges);
                    printf("\n");
                }
                EssFree(hInst, ppValiditySets[i]);
            }
            printf("\n");
        }
        else
            printf("\n\tNo member returned.\n");
    }
    printf("\n");
    EssFree(hInst,ppValiditySets);

}
```

関連トピック

- EssOtlQueryVaryingAttributes
- EssOtlVaryingAssociateAttribute
- EssOtlVaryingAssociateAttributeDimension
- EssOtlVaryingDisassociateAttribute
- EssOtlVaryingGetAttributeIndepDims

EssOtlVaryingGetAttributeIndepDims

指定した属性メンバーの、独立次元を戻します(ある場合)。

構文

```
     ESS_FUNC_M EssOtlVaryingGetAttributeIndepDims (
     hOutline, hAttrMember, *pCountOfIndepDims, **ppIndepDims,
**ppucIndependentTypes
     );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル
hAttrMember	ESS_HMEMBER_T	属性メンバーのハンドル
*pCountOfIndepDims	ESS_INT32_T	可変属性を制御する独立次元の数へのポインタ
**ppIndepDims	ESS_HMEMBER_T	独立次元についてのメンバー・ハンドルの配列へのポインタ
**ppucIndependentTypes	ESS_UCHAR_T	*pIndepDims に含まれる独立次元の配列へのポインタ

備考

独立次元は、属性の関連付けが変化する可能性がある状況を識別する値を持つ次元です。独立次元は属性次元が基本次元と関連付けられる場合に選択されます。VaryingGetAttributeIndepdims は、指定した属性の次元と関連付けられた独立次元のリストを戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    void TestEssOtlVaryingGetAttributeIndepDims()
{
    ESS_STS_T     sts = ESS_STS_NOERR;
    ESS_HOUTLINE_T     hOutline = ESS_NULL;
    ESS_OBJDEF_T     Object;
    ESS_STR_T     attrMbr, attrDim, baseMbr, baseDim;
    ESS_USHORT_T     i;
    ESS_HMEMBER_T     hAttrMbr;
    ESS_HMEMBER_T     hAttrDim;
    ESS_HMEMBER_T     hBaseMbr;
```

```
    ESS_HMEMBER_T    hBaseDim;
    ESS_INT32_T    pCountOfIndepDims;
    ESS_HMEMBER_T    *ppIndepDims;
    ESS_PMBRINFO_T    pMemberInfo;
    ESS_UCHAR_T    *pucIndependentTypes;


    memset(&Object, '\0', sizeof(Object));
    Object.hCtx = hCtx;
    Object.ObjType = ESS_OBJTYPE_OUTLINE;
    Object.AppName = szAppName;
    Object.DbName = szDbName;
    Object.FileName = szDbName;

    printf("\n");
    sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE, ESS_TRUE, &hOutline);
    printf("EssOtlOpenOutline sts: %ld\n",sts);

    printf("\nGet handles of members for tests:\n");
    attrMbr = "Contractor";
    sts = EssOtlFindMember(hOutline, attrMbr, &hAttrMbr);
    printf("EssOtlFindMember sts: %d\n", sts);
    attrDim = "Type";
    sts = EssOtlFindMember(hOutline, attrDim, &hAttrDim);
    printf("EssOtlFindMember sts: %d\n", sts);
    baseMbr = "Doe,Jane";
    sts = EssOtlFindMember(hOutline, baseMbr, &hBaseMbr);
    printf("EssOtlFindMember sts: %d\n", sts);
    baseDim = "Entities";
    sts = EssOtlFindMember(hOutline, baseDim, &hBaseDim);
    printf("EssOtlFindMember sts: %d\n", sts);

    /* Valid case with a valid attribute member handle. */
    printf("\nValid case with a valid attribute member handle:\n");
    sts = EssOtlVaryingGetAttributeIndepDims(hOutline, hAttrMbr, &pCountOfIndepDims,
&ppIndepDims, &pucIndependentTypes);
    printf("EssOtlVaryingGetAttributeIndepDims sts: %d\n", sts);
    if(pCountOfIndepDims)
    {
        printf("Independent dimension(s) for attribute member %s:", attrMbr);
        for (i = 0; i < pCountOfIndepDims; i++)
        {
            sts = EssOtlGetMemberInfo(hOutline, ppIndepDims[i], &pMemberInfo);
            printf("\n\t%s", pMemberInfo->szMember);
            switch(pucIndependentTypes[i])
            {
                case ESS_ASSOCIATE_TYPE_CONTINUOUS:
                    printf(" - (Continuous)");
                    break;
                case ESS_ASSOCIATE_TYPE_DISCRETE:
                    printf(" - (Discrete)");
                    break;
            }
        }
        printf("\n");
    }
    else
```

```
        printf("\tAttribute member %s has no independent dimension.\n", attrMbr);

    sts = EssUnlockObject(hCtx, ESS_OBJTYPE_OUTLINE, Object.AppName, Object.DbName,
Object.FileName);
    printf("\nEssUnlockObject sts: %d\n", sts);

    sts = EssOtlCloseOutline(hOutline);
    printf("EssOtlCloseOutline sts: %d\n",sts);
}
```

関連トピック

- EssOtlQueryVaryingAttributes
- EssOtlVaryingAssociateAttribute
- EssOtlVaryingAssociateAttributeDimension
- EssOtlVaryingDisassociateAttribute
- EssOtlVaryingGetAssociatedAttributes

EssOtlVerifyFormula

アウトラインが正しいことを確認します。この関数は、グローバル・アウトライン・エラーおよび不正なメンバーそれぞれのエラーの両方を戻します。この関数は EssOtlVerifyOutlineEx によって呼び出されますが、クライアント・プログラムから直接呼び出すこともできます。

構文

```
      ESS_FUNC_M
      EssOtlVerifyFormula
       (
      hOutline, hCtx, FormulaString, pErrorNumber, pErrorLine, MemberName,
ErrorBufferLength, ErrorMessage
      );
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
hCtx	ESS_HCTX_T	API コンテキスト・ハンドル。アウトラインがローカルのアウトラインである場合、実行中のサーバーにこの個別の hCtx を提供することも必要です。式の確認は、サーバー上でのみ行われ、ファイル・システムからのアウトラインはサーバーに接続されていないためです。このパラメータは通常 NULL である必要があります。
FormulaString	ESS_STR_T	構文上の計算式。
pErrorNumber	ESS_PULONG_T	エラー数へのポインタ。
pErrorLine	ESS_PULONG_T	エラー行番号へのポインタ。
MemberName	ESS_STR_T	式があるメンバーの名前。これはオプションのフィールドです。特に EssOtlVerifyFormula() がループ内で呼び出される場合に、この名前を提供すると、エラー・メッセージが拡張されます。

パラメータ	データ型	説明
ErrorBufferLength	ESS_ULONG_T	エラー・バッファのサイズ。
ErrorMessage	ESS_STR_T	エラー・バッファに含まれるエラー・メッセージ。これは、(エラー番号、行番号およびメンバー名を含む)エラーの記述的なメッセージを含む、前もって割り当てられた文字列です。長さは少なくとも 400 バイトに設定する必要があります。

備考

- 式にエラーがあっても、戻り値は通常 0 です。0 でない戻り値は、重大なコードレベルのエラーを意味します。
- この関数は EssOtlVerifyOutlineEx から呼び出されますが、クライアント・プログラムから直接呼び出すこともできます。

戻り値

この関数は、正常終了すると 0 を戻します。それ以外の場合は、OTLAPI_ERR_HOUTLINE または OTLAPI_NULL_ARG のいずれかのエラー・コードを戻します。戻り値は、式に軽微なエラーがある場合でも、0 になる場合があります。0 でない戻り値は、重大なコードレベルのエラーを示します。

式のエラーは、pErrorNumber 変数および pErrorLine 変数に戻されます。

0 でない戻り値は、コードレベルでの重大なエラーを示します。その場合、エラー確認は中断され、pErrorNumber および pErrorLine の両方が 0 に設定されます。

関連トピック

- EssOtlVerifyOutlineEx
- EssVerifyFilter
- EssVerifyRulesFile

EssOtlVerifyOutline

アウトラインが正しいことを確認します。この関数は、グローバル・アウトライン・エラーおよび不正なメンバーそれぞれのエラーの両方を戻します。

構文

```
ESS_FUNC_M
EssOtlVerifyOutline
 (
hOutline, pulErrors, pulCount, pMbrErrors
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pulErrors	ESS_PULONG_T	グローバル・アウトライン・エラーのビットマスク宛先を指すポインタ。現在、このフィールドには 1 つの値のみあります: ESS_OUTERROR_CURTOOMANYDIMS

パラメータ	データ型	説明
pulCount	ESS_PULONG_T	エラーのあるメンバーの数。pMbrErrors 配列の要素数を定義します。
pMbrErrors	772 ページの「ESS_OUTERROR_T」	*pulCount メンバーのある配列を指すポインタ。配列の各要素には、単一のメンバーのエラーが含まれています。

備考

- この関数では、次の点が確認されます:
  - 共有メンバーにおけるユーザー属性の重複
  - 重複するレベル名、世代名または別名
  - 属性の追加および関連付けに関する制限
- サーバーへのアウトラインの保存は、そのアウトラインにエラーがない場合のみ成功します(*pulErrors == 0 かつ*pulCount == 0)。
- pMbrErrors 配列を解放するには、**EssFree()**を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- ESS_OUTERROR_SHAREUDA
- ESS_OUTERROR_DUPGENLEVNAME

例

```
    #include <essapi.h>
#include <essotl.h>

ESS_STS_T     sts = 0;
ESS_OBJDEF_T    Object;
ESS_HOUTLINE_T   hOutline;
ESS_ULONG_T    ulErrors;
ESS_ULONG_T    ulCount;
ESS_POUTERROR_T  pMbrErrors = ESS_NULL;
ESS_APPNAME_T   szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T   szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);
```

```
if (!sts)
{
 sts = EssOtlVerifyOutline(hOutline, &ulErrors,
    &ulCount, &pMbrErrors);
}

if (pMbrErrors)
{
 EssFree(hInst, pMbrErrors);
}
```

関連トピック

- EssOtlNewOutline
- EssOtlOpenOutline
- EssOtlWriteOutline

EssOtlVerifyOutlineEx

指定したアウトラインが正しいことを確認し、そのアウトラインで検出したエラーの配列を構築します。この関数は、グローバル・アウトライン・エラーおよび不正なメンバーそれぞれのエラーの両方を戻します。

構文

```
ESS_FUNC_M
EssOtlVerifyOutlineEx
 (
hOutline, pulErrors, pulCount, pMbrErrors
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pulErrors	ESS_PULONG_T	グローバル・アウトライン・エラーのビットマスク宛先を指すポインタ。アウトラインに式のエラーがある場合、値のある唯一のフィールドは次のようになります: ESS_OUTERROREX_OUTLINEHASFORMULAERROR
pulCount	ESS_PULONG_T	エラーのあるメンバーの数。pMbrErrors 配列の要素数を定義します。アウトラインにエラーがある場合、エラーはビットマスクになります。アウトラインに式のエラーのみがある場合、pMbrError フィールドはエラー番号(ulErrors)と行番号(ulErrors2)から構成されます。この場合、pulErrors が ESS_OUTERROREX_OUTLINEHASFORMULAERROR に設定されます。
pMbrErrors	772 ページの「ESS_OUTERROR_T」	*pulCount メンバーのある配列を指すポインタ。配列の各要素には、単一のメンバーのエラーが含まれています。

備考

- この関数は EssOtlVerifyOutline を呼び出します。呼出しが正常終了すると、次に数式を含む各メンバーに対して EssOtlVerifyFormula を呼び出し、式にエラーがあれば、出力エラー配列に含みます。EssOtlVerifyOutline()の呼出

しが正常終了しなかった場合、この関数は EssOtlVerifyOutline()とまったく同じ機能になります。

- この関数では、次の点が確認されます:
  - 共有メンバーにおけるユーザー属性の重複。
  - 重複するレベル名、世代名または別名。
  - 属性の追加および関連付けに関する制限。
- pMbrErrors 配列を解放するには、**EssFree()**を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_OPENMODE
- OTLAPI_BAD_HOUTLINE OTLAPI_NULL_ARG

例

```
    ESS_STS_T TestVerifyOtlEx(ADT_CMDCTX_T *cmdctxp)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STS_T     sts2 = ESS_STS_NOERR;
 ESS_SHORT_T    hOutline;
 ESS_ULONG_T    ulErrors;
 ESS_ULONG_T    ulCount;
 ESS_POUTERROR_T  pMbrErrors;
 ESS_ULONG_T    ind;
 ESS_PMBRINFO_T  ppMbrInfo;

 if (cmdctxp->cmdbuf.argn < 2)
 {
  hOutlineChoice = ishOutlineMenu(cmdctxp);
 }
 else
 {
  hOutlineChoice = atoi(*(cmdctxp->cmdbuf.args + 1));
 }

 sts = EssOtlVerifyOutlineEx(cmdctxp->hOutline[hOutlineChoice], &ulErrors,
             &ulCount, &pMbrErrors);

 if (sts == ESS_STS_NOERR)
 {
  fprintf(cmdctxp->output, "\n------Global Errors------\n");

  if (ulErrors & ESS_OUTERROR_CURTOOMANYDIMS)
  {
   fprintf(cmdctxp->output, "Too many dimensions in currency outline\n");
  }
  else if(ulErrors & ESS_OUTERROR2_ATTRCALCABSENT)
  {
   fprintf(cmdctxp->output, "Attribute calculations dimension is absent\n");
  }
```

```
  else if(ulErrors & ESS_OUTERROREX_OUTLINEHASFORMULAERROR)
  {
   fprintf(cmdctxp->output, "Outline has formula error\n");
  }
  else if (ulErrors == 0)
  {
   fprintf(cmdctxp->output, "No errors\n");
  }
  else
  {
   fprintf(cmdctxp->output, "Unknown error\n");
  }

  fprintf(cmdctxp->output, "\n------Member Errors------\n");

      if(ulErrors != ESS_OUTERROREX_OUTLINEHASFORMULAERROR)
      {
           for (ind = 0; ind < ulCount; ind++)
           {
              sts2 = EssOtlGetMemberInfo(cmdctxp->hOutline[hOutlineChoice],
                                  pMbrErrors[ind].hMember, &ppMbrInfo);

              if (sts2 == ESS_STS_NOERR)
              {
                 fprintf(cmdctxp->output, "Member: %s\n", ppMbrInfo->szMember);
                 EssFree(cmdctxp->hInst, ppMbrInfo);
              }
              else
              {
                 fprintf(cmdctxp->output, "Member: Unknown member\n");
              }

              if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALNAME)
              {
                 fprintf(cmdctxp->output, "  ESS_OUTERROREX_OUTLINEHASFORMULAERROR
\n");
              }

              if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_DUPLICATENAME)
              {
                 fprintf(cmdctxp->output, "  ESS_OUTERROR_DUPLICATENAME\n");
              }

              if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALCURRENCY)
              {
                 fprintf(cmdctxp->output, "  ESS_OUTERROR_ILLEGALCURRENCY\n");
              }

              if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALDEFALIAS)
              {
                 fprintf(cmdctxp->output, "  ESS_OUTERROR_ILLEGALDEFALIAS\n");
              }

              if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALCOMBOALIAS)
              {
                 fprintf(cmdctxp->output, "  ESS_OUTERROR_ILLEGALCOMBOALIAS\n");
              }
```

```
if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALALIASSTRING)
{
   fprintf(cmdctxp->output, "  ESS_OUTERROR_ILLEGALALIASSTRING\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ILLEGALTAG)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_ILLEGALTAG\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_NOTIMEDIM)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_NOTIMEDIM\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_DUPLICATEALIAS)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_DUPLICATEALIAS\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_MEMBERCALC)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_MEMBERCALC\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_SHARENOTLEVEL0)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_SHARENOTLEVEL0\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_NOSHAREPROTO)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_NOSHAREPROTO\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_TIMESPARSE)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_TIMESPARSE\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_LEAFLABEL)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_LEAFLABEL\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_ALIASSHARED)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_ALIASSHARED\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADTIMEBAL)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADTIMEBAL\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADSKIP)
```

```
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADSKIP\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADSHARE)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADSHARE\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADSTORAGE)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADSTORAGE\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADCATEGORY)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADCATEGORY\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_BADSTORAGECATEGORY)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_BADSTORAGECATEGORY\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_SHAREDMEMBERFORMULA)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_SHAREDMEMBERFORMULA\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_SHAREUDA)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_SHAREUDA\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_DUPGENLEVNAME)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_DUPGENLEVNAME\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_VIRTLEV0NOFORMULA)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_VIRTLEV0NOFORMULA\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_VIRTBADPARENT)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_VIRTBADPARENT\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_VIRTBADCHILD)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_VIRTBADCHILD\n");
}

if (pMbrErrors[ind].ulErrors & ESS_OUTERROR_VIRTWHOLEDIMVIRTUAL)
{
   fprintf(cmdctxp->output,"  ESS_OUTERROR_VIRTWHOLEDIMVIRTUAL\n");
```

```
                }

                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_NOTLEVEL0)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_NOTLEVEL0\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_LEVELMISMATCH)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_LEVELMISMATCH\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALORDER)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALORDER\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALDATATYPE)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALORDER\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_DATATYPEMISMATCH)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_DATATYPEMISMATCH\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALATTRIBUTEPARENT)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALATTRIBUTEPARENT
\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ATTRDIMNOTASSOCIATED)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ATTRDIMNOTASSOCIATED\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALUDA)
                {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALUDA\n");
                }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_CHILDCOUNT)
               {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_CHILDCOUNT\n");
               }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALATTRCALC)
               {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALATTRCALC\n");
               }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_DUPLICATEATTRCALC)
               {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_DUPLICATEATTRCALC\n");
               }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALATTRSET)
               {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALATTRSET\n");
               }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALATTRCALCSET)
               {
                   fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALATTRCALCSET\n");
               }
               if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_NOTATTRIBUTE)
               {
```

```
                    fprintf(cmdctxp->output,"  ESS_OUTERROR2_NOTATTRIBUTE\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ATTRCALCABSENT)
                {
                    fprintf(cmdctxp->output,"  ESS_OUTERROR2_ATTRCALCABSENT\n");
                }
                if (pMbrErrors[ind].ulErrors2 & ESS_OUTERROR2_ILLEGALATTRVALUE)
                {
                    fprintf(cmdctxp->output,"  ESS_OUTERROR2_ILLEGALATTRVALUE\n");
                }
            }
        }

   if(ulErrors != ESS_OUTERROREX_OUTLINEHASFORMULAERROR)
   {
        for (ind = 0; ind < ulCount; ind++)
        {
           sts2 = EssOtlGetMemberInfo(cmdctxp->hOutline[hOutlineChoice],
                                pMbrErrors[ind].hMember, &ppMbrInfo);

           if (sts2 == ESS_STS_NOERR)
           {
              fprintf(cmdctxp->output, "Member: %s\n", ppMbrInfo->szMember);
              EssFree(cmdctxp->hInst, ppMbrInfo);
           }
           else
           {
              fprintf(cmdctxp->output, "Member: Unknown member\n");
           }

           fprintf(cmdctxp->output, "Error %d at line %d\n",
pMbrErrors[ind].ulErrors, pMbrErrors[ind].ulErrors2);
        }
   }

   if (ulCount == 0)
   {
    fprintf(cmdctxp->output, "No errors\n");
   }

   EssFree(cmdctxp->hInst, pMbrErrors);
  }

  fprintf(cmdctxp->output, "\nsts: %ld\n\n", sts);

  return(sts);
}
```

関連トピック

- EssOtlNewOutline
- EssOtlOpenOutline
- EssOtlWriteOutline
- EssOtlVerifyOutline
- EssOtlVerifyFormula

EssOtlWriteOutline

既存のアウトライン情報をディスクに書き込みます。

構文

```
ESS_FUNC_M
EssOtlWriteOutline
(
hOutline, pObject
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pObject	ESS_POBJDEF_T	書込み対象のアウトライン・オブジェクト。

備考

- アウトラインをサーバー・オブジェクトとして保存する場合、アウトラインは当初.OTN ファイルとして保存されます。次に **EssOtlRestructure()**を呼び出して、実際の.OTL ファイルを作成する必要があります。
- アウトラインをサーバー・オブジェクトとして保存する場合、オブジェクト名はデータベース名と同じである必要があります。
- サーバー・アウトライン・オブジェクトまたはクライアント・アウトライン・オブジェクトをローカル・データベースに保存する場合、データベースがすでに存在している必要があります。
- 指定されたユーザーによってアウトラインが現在ロックされていない場合、この呼出しは正常に行われません(ESS_OBJDEF_T 構造体の hCtx パラメータ)。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_NOTVERIFIED

アクセス

この関数を使用するには、指定したアプリケーションと、アウトライン・オブジェクトを含むデータベースのいずれか、またはその両方に対して、呼出し元が適切なレベルのアクセス権を持っている必要があります。アウトライン・オブジェクトを書き込むには、指定したアプリケーション、またはアウトラインが含まれたデータベースに対してアプリケーション・デザイナまたはデータベース・デザイナ権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)が必要です。

例

```
    #include <essapi.h>
#include <essotl.h>
```

```
ESS_STS_T     sts = 0;
ESS_HOUTLINE_T   hOutline;
ESS_OBJDEF_T    Object;
ESS_APPNAME_T    szAppName;
ESS_DBNAME_T    szDbName;
ESS_OBJNAME_T    szFileName;

memset(&Object, '\0', sizeof(Object));
Object.hCtx = hCtx;
Object.ObjType = ESS_OBJTYPE_OUTLINE;
strcpy(szAppName, "Sample");
strcpy(szDbName, "Basic");
strcpy(szFileName, "Basic");
Object.AppName = szAppName;
Object.DbName = szDbName;
Object.FileName = szFileName;

sts = EssOtlOpenOutline(hCtx, &Object, ESS_TRUE,
  ESS_TRUE, &hOutline);

/* body of code */
if (!sts)
{
 sts = EssOtlWriteOutline(hOutline, &Object);
}

/* restructure db using EssOtlRestructure() */
```

関連トピック

- EssOtlWriteOutlineEx
- EssOtlOpenOutline
- EssOtlNewOutline
- EssOtlVerifyOutline
- EssOtlRestructure
- EssOtlCloseOutline

EssOtlWriteOutlineEx

既存のアウトライン情報をディスクに書き込み、UTF-8 エンコード方式と非 Unicode エンコード方式のどちらで保存するかを指定します。

構文

```
ESS_FUNC_M
     EssOtlWriteOutlineEx
     (
     hOutline
     ,
     pObject
     ,
     iOtlType
```

```
);
```

パラメータ	データ型	説明
hOutline	ESS_HOUTLINE_T	アウトラインのコンテキスト・ハンドル。
pObject	ESS_POBJDEF_T	書込み対象のアウトライン・オブジェクト。
iOtlType		アウトラインを Unicode モードで保存するか非 Unicode モードで保存するか。 有効な値は次のとおりです: ● ESS_OUTLINE_UTF8 0x0002 - UTF-8 でエンコードします。 ● ESS_OUTLINE_NONUNICODE 0x0003 - Unicode でエンコードしません。

備考

- アウトラインをサーバー・オブジェクトとして保存する場合、アウトラインは当初.OTN ファイルとして保存されます。次に **EssOtlRestructure()**を呼び出して、実際の.OTL ファイルを作成する必要があります。
- アウトラインをサーバー・オブジェクトとして保存する場合、オブジェクト名はデータベース名と同じである必要があります。
- サーバー・アウトライン・オブジェクトまたはクライアント・アウトライン・オブジェクトをローカル・データベースに保存する場合、データベースがすでに存在している必要があります。
- 指定されたユーザーによってアウトラインが現在ロックされていない場合、この呼出しは正常に行われません(ESS_OBJDEF_T 構造体の hCtx パラメータ)。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_NOTVERIFIED

アクセス

この関数を使用するには、指定したアプリケーションと、アウトライン・オブジェクトを含むデータベースのいずれか、またはその両方に対して、呼出し元が適切なレベルのアクセス権を持っている必要があります。アウトライン・オブジェクトを書き込むには、指定したアプリケーション、またはアウトラインが含まれたデータベースに対してアプリケーション・デザイナまたはデータベース・デザイナ権限(ESS_PRIV_APPDESIGN または ESS_PRIV_DBDESIGN)が必要です。

関連トピック

- EssOtlOpenOutlineEx
- EssOtlCloseOutline
- EssOtlGetMemberCommentEx
- EssOtlNewOutline
- EssOtlRestructure
- EssOtlSetMemberCommentEx

- EssOtlVerifyOutline

12 CのアウトラインAPIの例

この章の内容

アウトラインの走査例

この例では、アウトライン・ツリーの使用方法を示します。TraverseTree は再帰的アルゴリズムで、アウトライン・ツリーを走査してすべてのアウトライン・メンバーへのアクセスを提供します。順番に各メンバーを選択して、最終メンバーに到達するまでそれぞれで処理します。コード中のコメントは、追加の処理ができることを示します。

このアルゴリズムには、複数の C のアウトライン API コマンドが組み込まれています。

- **EssOtlGetFirstMember()**は、アウトライン中の最初のメンバー(最初に定義されている次元)へのメンバーのハンドルを戻します。
- **EssOtlGetMemberInfo()**は、指定されたメンバーに関する情報を取得します。
- **EssOtlGetChild()**は、メンバーの子を戻します。
- **EssOtlGetNextSibling()**は、メンバーの次の兄弟を戻します。

このコードを実行する前に、API を初期化してアウトラインを開きます。このコードの後に、アウトラインを閉じて API を終了します。

```
   TraverseTree (ESS_HOUTLINE_T)
{
 ESS_HMEMBER_T hMember;
 ESS_STS_T  sts = 0;

 sts = EssOtlGetFirstMember(hOutline, &hMember);
 if (!sts && hMember)
  sts = TraverseTreeRecurse(hOutline, hMember);
}

TraverseTreeRecurse(ESS_HOUTLINE_T hOutline, ESS_HMEMBER_T hMember)
{
 ESS_MEMBERINFO_T MbrInfo;
 ESS_HMEMBER_T, hChild;
 ESS_STS_T  sts = 0;
```

```
 while (!sts && hMember)
 {
  sts = EssOtlGetMemberInfo (hOutline, hMember, &MbrInfo);

  /* ADD THE PROCESSING FOR EACH MEMBER HERE. */

  if (!sts)
  {
   sts = EssOtlGetChild(hOutline, hMember, &hChild);
   if (!sts && hChild)
   {
    sts = TraverseTreeRecurse(hOutline, hChild);
   }
  }
  sts = EssOtlGetNextSibling(hOutline, hMember, &hMember);
 }
 return (sts);
}
```

拡張メンバーのクエリー・コードの例

```
#include <windows.h>
#include <essapi.h>
#include <essotl.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#define AD_CHK_PRINTF_1(ARG1, ARG2)
do
{
 printf(ARG1, (ARG2) ? (ARG2) : "NullValue");
}
while (0)

void PrintResult(ESS_HCTX_T   hCtx,
        ESS_HINST_T   hInst,
        ESS_HOUTLINE_T  hOutline,
        ESS_HMEMBER_T  hMbr)
{
 ESS_PMBRINFO_T pMbrInfo = NULL;
 ESS_STS_T   sts;
 int      size;
 ESS_STR_T   pszFormula = NULL;
 ESS_STR_T   pszLastFormula = NULL;
 ESS_STR_T   pszCommentEx = NULL;
 ESS_STR_T   pszAlias = NULL;
 ESS_STR_T   pszAliasCombo = NULL;
 ESS_PMBRNAME_T pUDAList = NULL;
 ESS_USHORT_T  iCount = 0;
 ESS_STR_T   pszPrev = NULL;
 ESS_USHORT_T  iIndex;
```

```
 ESS_ULONG_T* pMemNum;
 ESS_ULONG_T* pDimNum;
 ESS_STR_T  pDimName = NULL;
 ESS_STR_T  pAliasName = NULL;
 ESS_STR_T  pNextName = NULL;
 ESS_STR_T  pPrevName = NULL;
 ESS_STR_T  pParentName = NULL;
 ESS_STR_T  pChildName = NULL;
 ESS_BOOL_T* pCurrConv = NULL;
 ESS_ULONG_T* pStatus = NULL;

 sts = EssOtlGetMemberInfo(hOutline, hMbr, &pMbrInfo);
 if (sts != 0) goto Error;

 size = sizeof(ESS_MBRINFO_T);

 printf("MbrInfo\n");
 AD_CHK_PRINTF_1("  szMember --------------->(%s)\n", pMbrInfo->szMember);
 printf("  usLevel ---------------->(%hd)\n", pMbrInfo->usLevel);
 printf("  usGen ------------------>(%hd)\n", pMbrInfo->usGen);
 printf("  usConsolidation ------->(%hd)\n", pMbrInfo->usConsolidation);
 printf("  fTwoPass --------------->(%hd)\n", pMbrInfo->fTwoPass);
 printf("  fExpense --------------->(%hd)\n", pMbrInfo->fExpense);
 printf("  usConversion ----------->(%hd)\n", pMbrInfo->usConversion);
 AD_CHK_PRINTF_1("  szCurMember ------------>(%s)\n", pMbrInfo->szCurMember);
 printf("  usTimeBalance ---------->(%hd)\n", pMbrInfo->usTimeBalance);
 printf("  usSkip ----------------->(%hd)\n", pMbrInfo->usSkip);
 printf("  usShare ---------------->(%hd)\n", pMbrInfo->usShare);
 printf("  usStorage -------------->(%hd)\n", pMbrInfo->usStorage);
 printf("  usCategory ------------->(%hd)\n", pMbrInfo->usCategory);
 printf("  usStorageCategory ------>(%hd)\n", pMbrInfo->usStorageCategory);
 AD_CHK_PRINTF_1("  szComment -------------->(%s)\n", pMbrInfo->szComment);
 printf("  ulChildCount ----------->(%ld)\n", pMbrInfo->ulChildCount);

 sts = EssOtlGetMemberFormula(hOutline, hMbr, &pszFormula);
 if (sts) printf("sts=%d ", sts);
 AD_CHK_PRINTF_1("szFormula ------------------>(%s)\n", pszFormula);

 sts = EssOtlGetMemberLastFormula(hOutline, hMbr, &pszLastFormula);
 if (sts) printf("sts=%d ", sts);
 AD_CHK_PRINTF_1("szLastFormula -------------->(%s)\n", pszLastFormula);

 sts = EssOtlGetMemberCommentEx(hOutline, hMbr, &pszCommentEx);
 if (sts) printf("sts=%d ", sts);
 AD_CHK_PRINTF_1("szCommentEx ---------------->(%s)\n", pszCommentEx);

 sts = EssOtlGetMemberAlias(hOutline, hMbr, ESS_NULL, &pszAlias);
 if (sts) printf("sts=%d ", sts);
 AD_CHK_PRINTF_1("szAlias (Default)----------->(%s)\n", pszAlias);

 sts = EssOtlGetNextAliasCombination(hOutline, hMbr, ESS_NULL, "\0",
&pszAliasCombo);
 if (sts) printf("sts=%d ", sts);

 printf("szAliasCombo ::\n" );
 pszPrev = pszAliasCombo;
```

```
 while (sts && pszAliasCombo)
 {
  AD_CHK_PRINTF_1("\t(%s)\n", pszAliasCombo);
  sts = EssOtlGetNextAliasCombination(hOutline, hMbr, ESS_NULL, pszPrev,
&pszAliasCombo);
  EssFree(hInst, pszPrev);
  pszPrev = pszAliasCombo;
 }

 sts = EssOtlGetUserAttributes(hOutline, hMbr, &iCount, &pUDAList);
 if (sts) printf("sts=%d ", sts);

 printf("User Defined Attributes ::\n");
 for(iIndex = 0; iIndex < iCount; iIndex++)
  AD_CHK_PRINTF_1("\t(%s)\n", pUDAList[iIndex]);

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_NUMBER, (ESS_PPVOID_T)
&pMemNum);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  printf("Member Number ------------------>(%ld)\n", *pMemNum);
  EssFree(hInst, pMemNum);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_DIMNUMBER, (ESS_PPVOID_T)
&pDimNum);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  printf("Dimension Number ----------------->(%ld)\n", *pDimNum);
  EssFree(hInst, pDimNum);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_DIMNAME, (ESS_PPVOID_T)
&pDimName);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  AD_CHK_PRINTF_1("Dimension Name ------------------>(%s)\n", pDimName);
  EssFree(hInst, pDimName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_ALIASNAME, (ESS_PPVOID_T)
&pAliasName);
 if (sts)
 {
  printf("sts=%d ", sts);
```

```
 }
 else
 {
  AD_CHK_PRINTF_1("Alias Name   ------------------>(%s)\n", pAliasName);
  EssFree(hInst, pAliasName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_NEXTNAME, (ESS_PPVOID_T)
&pNextName);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  AD_CHK_PRINTF_1("Next Mbr  Name ------------------>(%s)\n", pNextName);
  EssFree(hInst, pNextName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_PREVNAME, (ESS_PPVOID_T)
&pPrevName);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  AD_CHK_PRINTF_1("Prev Mbr  Name ------------------>(%s)\n", pPrevName);
  EssFree(hInst, pPrevName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_PARENTNAME, (ESS_PPVOID_T)
&pParentName);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  AD_CHK_PRINTF_1("Parent MbrName ------------------>(%s)\n", pParentName);
  EssFree(hInst, pParentName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_CHILDNAME, (ESS_PPVOID_T)
&pChildName);
 if (sts)
 {
  printf("sts=%d ", sts);
 }
 else
 {
  AD_CHK_PRINTF_1("Child Mbr Name ------------------>(%s)\n", pChildName);
  EssFree(hInst, pChildName);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_CURRENCYCONVDB,
(ESS_PPVOID_T) &pCurrConv);
```

```
 if (sts)
 {
  printf("sts=%d ", sts); printf("Curr Conv Type ------------------>\n");
 }
 else
 {
  AD_CHK_PRINTF_1("Curr Conv Type ------------------>(%ld)\n", *pCurrConv);
  EssFree(hInst, pCurrConv);
 }

 sts = EssOtlGetMemberField(hOutline, hMbr, ESS_OTLQRYMBR_STATUS, (ESS_PPVOID_T)
&pStatus);
 if (sts)
  { printf("sts=%d ", sts); printf("Status  ------------------>\n"); }
 else
 {
  printf("Status    ------------------>(%hd)\n", *pStatus);
  EssFree(hInst, pStatus);
 }

 EssFree(hInst, pMbrInfo);
 EssFree(hInst, pszFormula);
 EssFree(hInst, pszLastFormula);
 EssFree(hInst, pszCommentEx);
 EssFree(hInst, pszAlias);
 EssFree(hInst, pszAliasCombo);
 return;

Error:
 printf("****************** Error ************************");
}


int TestCode_EssOtlQueryMembersEx(ESS_HCTX_T hCtx,
               ESS_HINST_T  hInst)
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_HOUTLINE_T hOutline;
 ESS_OBJDEF_T Object;
 ESS_HMEMBER_T hMember = 0;
 ESS_PHMEMBER_T phMemberArray = ESS_NULL;
 ESS_ULONG_T i;
 unsigned long MaxCount = -1;
 ESS_STR_T  member_fields;
 ESS_STR_T  member_selection;

 /* query string to get level numbers of all markets members */
 member_fields  = "<SelectMbrInfo ( MemberName, MemberLevel,Consolidation,
MemberFormula ) ";
 member_selection = "@ichild(Product), @ichild(Market)";
 memset(&Object, '\0', sizeof(Object));
 Object.hCtx   = hCtx;
 Object.ObjType  = ESS_OBJTYPE_OUTLINE;
 Object.AppName  = "Basic";
 Object.DbName   = "Demo";
 Object.FileName  = "Demo";
```

```
 sts = EssOtlOpenOutlineQuery(hCtx, &Object, &hOutline);
 if (sts) goto exit;

 if(!sts)
 {
  ESS_POTLQUERYERRORLIST_T pqryErrorList;

  sts = EssOtlQueryMembersEx(hOutline, member_fields, member_selection, &MaxCount,
&phMemberArray, &pqryErrorList);
  if (sts) goto exit;

  if (phMemberArray)
   for (i = 0; i< MaxCount; i++)
    PrintResult(hOutline, phMemberArray[i]);
 }

 if(MaxCount && phMemberArray)
 {
  sts = EssOtlFreeMembers(hOutline, MaxCount, phMemberArray);
  if (sts)
   printf("EssOtlFreeMembers  sts = %d\n",sts);
 }

 sts = EssOtlCloseOutline(hOutline);

exit:
 return sts;
}
```

EssOtlQueryMembersEx または EssOtlGetMemberField に戻ります。

第 I V 部

C グリッド API

C グリッド API の内容：

13 CグリッドAPIの使用

この章の内容

C グリッド API に関する一般情報

Essbase アプリケーション・プログラミング・インタフェース(API)を使用して、Essbase サーバーへのカスタム・インタフェースを作成します。

グリッド API 関数は、グリッド・パラダイムの Essbase サーバーと対話します。グリッド API 関数を使用して、Essbase データベースからデータを抽出し、グリッドベースのレポート・インタフェースまたはチャートにデータを表示します。

グリッド API 関数は、Smart View および他の自動グリッド・ツールの機能をすべて含んでいます。これらの関数は、クエリー、ドリルダウン、選択項目のみ保持、およびピボットを含みます。グリッド API は、レポート指定を起動し、グリッドの形式で結果データを表示できます。

Essbase グリッド API は、EssbaseAPI およびレポート・スクリプト・コマンドを使用して、レポート作成アプリケーションを構築している開発者に大変有用です。利点のいくつかは次のとおりです:

- グリッド API はグリッドベースのデータ集約に最適化されています。レポート・スクリプトによる取得より著しく高速で、向上したクエリー性能をアプリケーションに提供します。
- グリッド API では、既存のアプリケーションに堅牢な対話型更新機能を簡単に追加できます。Essbase の高性能計算機とグリッド API を組み合せて使用すると、大幅な追加機能を比較的小さな負担で利用できます。

- グリッド API では、プログラムは、Essbase から戻されたデータを解析する必要はありません。グリッド API は、個別の各セル・メンバーのタイプおよび値を通知する、2 次元バイナリ形式でデータを自動的に配置します。
- ズーム・イン、ズーム・アウト、ピボットなど、ほとんどの Smart View コマンドはグリッド API から利用可能です。サードパーティ製グリッド制御を使用して、カスタム・ユーザー・インタフェースを構築できます。
- また、グリッド API は、共有(暗黙または明示)、親、子などのメンバー属性へのアクセスを提供します。これらの属性を使用して、Essbase データのルック・アンド・フィールをカスタマイズできます。

C グリッド API アーキテクチャの概要

Essbase グリッド API 関数は共通グリッド・レイヤーを使用して、Essbase サーバーと通信します。

グリッド API 関数はグリッド上で動作を実行します。アクションの結果もグリッドです。呼出し元は、各呼出しに対する 2 次元データをグリッド API 関数に提供し、戻されたデータを表示する必要があります。また呼出し元は、エラー・メッセージ・ダイアログ・ボックスなどの通知を、API アプリケーションの接続がタイム・アウトしないように処理する必要があります。

C グリッド API をサポートしているプラットフォームとコンパイラ

Essbase API がサポートされているプラットフォームのリストは、Oracle Hyperion Enterprise Performance Management System 動作保証マトリックス(http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html)を参照してください。Essbase API がサポートしているコンパイラ・リリースのリストは、29 ページの「サポートされているコンパイラ」を参照してください。

関数名とパラメータ順序はすべてのプラットフォームで共通です。ただし、各プラットフォームごとに異なるファイルをリンクする必要があります。45 ページの「API ライブラリ」を参照してください。

- Microsoft Visual C++など、統合化された C 環境を使用している場合は、コンパイラとリンカー・オプションを慎重に確認して、Essbase API が正しく機能することを確認する必要があります。特に、構造体フィールドが 1 バイトで揃っていること、正しいライブラリが使用されていること(Intel X86 プラットフォーム上の大容量メモリー・モデルの使用)を確認する必要があります。さらに、リンク・プロセスには必ず適切な Essbase API ライブラリを含むようにします。45 ページの「API ライブラリ」を参照してください。
- シングルバイトの構造体配置を使って、Essbase API 関数をコンパイルする必要があります。Microsoft コンパイラを使用している場合は、pragma を使用できます:

```
#pragma pack(push, id, 1)
```

```
#include "essgapi.h"
#pragma pack(pop, id)
```

C グリッド API プログラムに含めるファイル

Essbase API 関数をプログラムで使用するには、Essbase API 定義を含むファイルを含める必要があります。

グリッド API 関数を C プログラムで使用するには、グリッド API ヘッダー定義ファイル(ESSGAPI.H)を適切なソース・モジュールに含める必要があります。グリッド API 関数の他、通常の API 関数を使用している場合は、メイン API 定義ファイル(ESSAPI.H)を適切なソース・モジュールに含める必要もあります。

これらの定義ファイルは、常に C ランタム・ライブラリ・ヘッダー・ファイルの後に含めてください。Windows 環境でプログラミングする場合は、ESSGAPI.H を、オプションで ESSAPI.H を、Windows 組込みファイル WINDOWS.H の後に配置します。

C グリッド API の初期化と設定

グリッド API 関数を使用する場合、初期化関数 EssGInit を呼び出す必要があります。この関数は、次のタスクを実行します:

- 環境に関する情報をグリッド API 関数に渡します。
- 後でグリッド API 関数と通信するために使用する、インスタンス・ハンドルを提供します。

注意:

- EssGInit を呼び出すと、グリッド・インスタンス・ハンドルが取得されます。通常の API 関数を使用する場合は、EssGGetAPIInstance を使用して API インスタンス・ハンドルを取得し、EssGGetAPIContext を使用してログイン・コンテキスト・ハンドルを取得できます。1082 ページの「C メイン API 関数の使用」を参照してください。
- メイン API インスタンス・ハンドルとログイン・コンテキストをグリッド API 呼出しで使用できません。1082 ページの「C メイン API 関数の使用」を参照してください。

C グリッド API のメモリー管理

ユーザーが割り当てたメモリー、および使用したグリッド API 関数が割り当てたメモリーを解放する必要があります。必要に応じてメモリーを解放するグリッド API 関数があります。

C グリッド API のバージョン管理

EssGInit を使用して API を初期化する場合、アプリケーションのコンパイルに使用する API ライブラリのバージョン番号を渡す必要があります。これによって、アプリケーションを再配布しなくても、以前のバージョンのアプリケーションが新しいグリッド API DLL および CSL DLL を使用できるようになります。

グリッド API 関数はグリッド API の現在のリリースをレポートします。この呼出しを行う前に初期化を行う必要はありません。

C グリッド API 関数の使用

多くの操作では、操作の開始に呼出しが要求されます。その他の呼出しは操作を完了し、データを取得するために実行する必要があります。次のリストは、これらの操作呼出しを実行する順序を示します:

1. **EssGBeginXxx** を呼び出して操作を開始します。
2. **EssGSendRows** を呼び出して行を送信します。これを複数回呼び出して、複数のデータを送信できます。
3. **EssGPerformOperation** を呼び出して、すべての情報が渡されたことを API へ伝達します。
4. **EssGGetResults** を呼び出して、受け取る行数と列数に関する情報を戻します。
5. すべてのデータを取得するまで **EssGetRows** を呼び出します。
6. **EssGEndOperation** を呼び出して、内部リソースをクリーン・アップします。
7. このプロセスの必要な段階で **EssGCancelOperation** を呼び出して、操作を取り消します。

EssGEndOperation、**EssGCancelOperation**、またはもう一度 **EssGBeginXxxx** を呼び出すと、以前の操作に関する情報はすべて消去されます。

C メイン API 関数の使用

グリッド API 関数はグリッドに固有のもので、メインの Essbase API 関数にかわるものではありません。このように操作が分離しているため、グリッド API プログラムからメイン API 関数を呼び出す必要が生じることもあります。

メイン API を呼び出すには、次の 2 つの情報が必要です:

- Essbase インスタンス・ハンドル
- 有効なログイン・コンテキスト

EssGConnect および **EssGNewGrid** は呼び出しておらず、**EssGInit** は呼び出した場合、**EssGGetAPIInstance** を呼び出して Essbase インスタンスを入手できます。これによって、メモリー呼出しである **EssAlloc**、**EssFree**、**EssRealloc**、およびログイン呼出しである **EssLogin** および **EssAutoLogin** にアクセスできます。

有効なグリッド・ハンドルを入手し、接続したら、**EssGGetAPIContext** を呼び出して有効なログイン・コンテキストを入手できます。次にこのログイン・コンテキスト・ハンドルを、ログイン・コンテキスト・ハンドルを受け付ける任意の Essbase 関数と一緒に使用できます。ログイン・コンテキストを変更する Essbase API 関数では、グリッド API からログイン・コンテキストを使用しないでください。ログイン・コンテキストを変更する関数は、**EssLogin**、**EssAutoLogin**、**EssSetActive** および **EssClearActive** です。

メイン Essbase API で入手したハンドルおよびログイン・コンテキストは、グリッド API 呼出しでは使用できません。メイン Essbase API およびグリッド API の両方を使用する場合は、グリッド API で初期化、接続して、グリッド API から他の Essbase 関数用にハンドルとログイン・コンテキストを使用する必要があります。

C グリッド API の座標系

データの 2 次元配列を受け取る関数に渡される範囲構造体で、ゼロベースの列と行の番号付けスキームを想定します。入力と出力のデータ範囲は同じ相対座標系にあり、データ配列は常にゼロベースになります。

たとえば、最初のデータ・セルが 3 番目の行と 4 番目の列にあり、それぞれ 5 つの列に 3 つの行があるとします。構造体 `ESSG_RANGE` を渡すと、ulStartRow = 2、ulStartColumn = 3、ulNumRows = 3、ulNumColumn = 5 が含まれます。

`ESSG_DATA_T` アイテムの 2 次元配列は、インデックス `[0][0]` で始まり、インデックス `[2][4]` で終わります。

14 CグリッドAPIの宣言

この章の内容

C グリッド API の定数

次の定数は、Essbase グリッド API で定義されています。

API 呼出しが正常終了すると戻される定数。

定数	定義
ESSG_STS_NOERR	0

EGAPI API のバージョンを定義する定数。API が変更されるたびに変わります。

定数	定義
ESSG_VERSION	0x00040000

サポートする行と列の最大数を定義する定数

定数	定義
WINDOWS	
ESSG_MAXROWS	0xFFFFFFFF / sizeof(ESSG_PDATA_T)
ESSG_MAXCOLUMNS	0xFFFFFFFF / sizeof(ESSG_DATA_T)
その他	
ESSG_MAXROWS	0xFFFF / sizeof(ESSG_PDATA_T)
ESSG_MAXCOLUMNS	0xFFFF / sizeof(ESSG_DATA_T)

1094 ページの「ESSG_DATA_T」構造体の pAttributes メンバーで使用する定数。

定数	定義
ESSG_CA_READONLY	0x00000001

定数	定義
ESSG_CA_READWRITE	0x00000002
ESSG_CA_LINKEDOBJ	0x00000004
ESSG_CA_LINKPARTITION	0x00000008
ESSG_CA_LINKCELLNOTE	0x00000010
ESSG_CA_LINKWINAPP	0x00000020
ESSG_CA_LINKURL	0x00000040
ESSG_CA_AISDT	0x00000080
ESSG_CA_GLDT	0X00000400

1094 ページの「ESSG_DATA_T」構造体の pAttributes メンバーで使用する定数。

定数	定義
ESSG_MA_DIMTOP	0x00000001
ESSG_MA_ZOOMINABLE	0x00000002
ESSG_MA_NEVERSHARE	0x00000004
ESSG_MA_LABELONLY	0x00000008
ESSG_MA_STOREDATA	0x00000010
ESSG_MA_EXPSHARE	0x00000020
ESSG_MA_IMPSHARE	0x00000040
ESSG_MA_DYNCALC	0x00000080
ESSG_MA_FORMULA	0x00000100
ESSG_MA_ATTRIBUTE	0x00000200
ESSG_MA_DIMNUMBITS	0xF8000000(右端からの 5 ビットは、次元数)

1094 ページの「ESSG_DATA_T」構造体の usType メンバーで使用する定数。

定数	定義
ESSG_DT_UNUSED	0
ESSG_DT_STRING	1
ESSG_DT_LONG	2
ESSG_DT_DOUBLE	3
ESSG_DT_BLANK	4

定数	定義
ESSG_DT_RESERVED	5
ESSG_DT_ERROR	6
ESSG_DT_MISSING	7
ESSG_DT_ZERO	8
ESSG_DT_NOACCESS	9
ESSG_DT_MEMBER	10
ESSG_DT_FORMULA	11
ESSG_DT_ZEROwFORMULA	12
ESSG_DT_DOUBLEwFORMULA	13
ESSG_DT_BLANKwFORMULA	14
ESSG_DT_STRINGwFORMULA	15
ESSG_DT_MISSINGwFORMULA	16
ESSG_DT_NOACCESSwFORMULA	17
ESSG_DT_STRINGEX	18
ESSG_DT_MEMBEREX	19
ESSG_DT_STRINGEXwFORMULA	20
ESSG_DT_FORMULAEX	21
ESSG_DT_MEMBERwKEY(1207 ページの「ESSG_DT_MEMBERwKEY の例」を参照)	23
ESSG_DT_SMARTLIST	24
ESSG_DT_MNGLESS	25
ESSG_DT_DATE	26

注： ESSG_DATA_T 構造体の usType フィールドが ESSG_DT_MEMBERwKEY に設定されている場合、Value(ESSG_DATA_VALUE)フィールドの pszStr フィールドは次のように解釈されます: <メンバー名の長さ><メンバー名><キーの長さ><キー>。ここで長さ要素はサイズが 2 バイトです。<メンバー名>が NULL 文字で終了することに注意してください。

EssGBeginRetrieve の ulOptions パラメータの値として使用する定数

定数	定義
ESSG_RET_RETRIEVE	0

定数	定義
ESSG_RET_RETRIEVELOCK	1
ESSG_RET_LOCKONLY	2

EssGBeginUpdate の ulOptions パラメータで使用する定数

定数	定義
ESSG_RET_REQUIRELOCK	0
ESSG_RET_LOCKIFNEEDED	1

EssGBeginConditionalRetrieve、**EssGBeginConditionalZoomIn**、**EssGBeginReport** および **EssGBeginReportFile** の ulOptions パラメータで使用するビットマスク定数

定数	定義
ESSG_NOATTRIBUTES	0x00001000

EssGBeginZoomIn および **EssGBeginConditionalZoomIn** の ulOptions パラメータで使用するビットマスク定数

定数	定義
ESSG_ZOOM_DOWN	0x00000080
ESSG_ZOOM_ACROSS	0x00000100

接続オプションを表す定数

定数	定義
ESSG_CONNECT_DEFAULT	0
ESSG_CONNECT_NODIALOG	1

各ズーム・レベルを表す定数

定数	定義
ESSG_OPTIONS	0
ESSG_NEXTLEVEL	1
ESSG_ALLLEVELS	2
ESSG_BOTTOMLEVEL	3
ESSG_SIBLEVEL	4
ESSG_SAMELEVEL	5

定数	定義
ESSG_SAMEGENERATION	6
ESSG_CALCLEVEL	7
ESSG_PARENTLEVEL	8
ESSG_TOPLEVEL	9

グリッド・オプションの設定と取得に使用する定数

定数	定義
ESSG_OP_DRILLLEVEL	1
ESSG_OP_INCSEL	2
ESSG_OP_SELONLY	3
ESSG_OP_SELGROUP	4
ESSG_OP_INDENT	5
ESSG_OP_SUPMISSING	6
ESSG_OP_SUPZEROS	7
ESSG_OP_SUPUNDER	8
ESSG_OP_UPDATEMODE	9
ESSG_OP_ALIASNAMES	10
ESSG_OP_ALIASTABLE	11
ESSG_OP_USERGRIDDATA	12
ESSG_OP_RETAINTHREAD	20
ESSG_OP_EMPTYGRIDERROR	21
ESSG_OP_DRILLONLEAF	22
ESSG_OP_DATALESS	23
ESSG_OP_SPANHYBRIDANALYSIS	24
ESSG_OP_UNIQUENAMEONLY	32
ESSG_OP_MEMBERANDUNIQUENAME (1205 ページの「ESSG_OP_MEMBERANDUNIQUENAME の例」を参照)	33
ESSG_OP_GET_ME_CELLS	36 基本メンバーと属性の組合せを持たないセルに対して#ME (無意味)値を戻します デフォルト: オフです

定数	定義
ESSG_OP_GET_FORMATTED_VALUE	38 フォーマットされたセルに対するフォーマット値を含めます デフォルト: セル値のみを戻します
ESSG_OP_GET_VALUE	39 非数値タイプのセルに対する元の値を要求します デフォルト: オンです
ESSG_OP_GET_FORMATTED_MISSING	40 欠落した値を持つセルに対してフォーマット値を含めます デフォルト: オフです
ESSG_OP_GET_DRILLTHRU_URLS	41 セルのドリルスルー・フラグを移入します

各インデント・スタイルを定義する定数

定数	定義
ESSG_INDENTNONE	1
ESSG_INDENTSUBITEMS	2
ESSG_INDENTTOTALS	3

結果を取得する呼出しでプロセスの状態を判断するために使用する定数

定数	定義
ESSG_STATE_DONE	1
ESSG_STATE_INPROGRESS	2

バッファの長さ(末尾の NULL 文字を含む)を表す定数

定数	定義
ESSG_USERNAMELEN	31
ESSG_PASSWORDLEN	101
ESSG_SERVERLEN	31
ESSG_APPLICATIONLEN	9
ESSG_DATABASELEN	9

グリッド API のドリルスルー関数で使用する定数(EssGDTxxx())

定数	定義
ESSG_DESCRIPTION_LEN	レポート・データ用のバッファの最大長(255)
ESSG_DTINPUTOPTION_PROMPT_HISNAME	ESSG_DTINFO_T の **uInputOption** の値。Essbase Studio へ接続してドリルスルー・セッションを開始するために必要なすべてのデフォルト値が、ユーザー側に揃っていることを意味します
ESSG_DTINPUTOPTION_PROMPT_LOGIN	ESSG_DTINFO_T の **uInputOption** の値。Essbase Studio へ接続してドリルスルー・セッションを開始するためのパスワードをユーザー側で設定するよう指定します
ESSG_DTREPORT_NAME	ドリルスルーに使用する文字列の最大長(80)
ESSG_ERR_INVALIDDTHANDLE	指定したドリルスルー・インスタンス・ハンドルが無効な場合に戻されるエラー・メッセージ定数
ESSG_ERR_NODTREPORTS	指定したドリルスルー・インスタンス・ハンドルに対して定義されたドリルスルー・レポートがない場合に戻されるエラー・メッセージ定数
ESSG_FIELDLEN	ドリルスルーに使用する文字列の最大長(30)
ESSG_HISDT	ドリルスルー・エントリに使用する値(5)

構造体 ESSG_LRODESC_T の LRO API 呼出しで使用する定数

定数	定義
ESSG_PARTITIONTYPE	1
ESSG_CELLNOTETYPE	2
ESSG_WINAPPTYPE	3
ESSG_URLTYPE	4

グリッドのパースペクティブ・タイプ

EssGGetGridPerspective および EssGSetGridPerspective によって使用されます。

定数	定義
ESSG_PERSP_EXPLICIT	0 タプルの指定が必要です
ESSG_PERSP_REALITY	1 属性次元の実コンテキストを使用します

テキスト・リスト(スマートリスト)タイプ

テキスト・リスト(スマートリスト)属性。

定数	定義
ESSG_CA_MISSINGCELL	0x00000100 セルに値#Missing がある場合、スマートリストのセル・タイプを設定します。これは、スマートリスト・セルの#Missing 値がテキスト値にマッピングされる場合に発生します。
ESSG_CA_OUTOFRANGE	0x00000200 数値を持つスマートリスト・タイプのセルがこのテキスト・リストのコンテキストの範囲外の場合に設定します

Unicode モードのタイプ

Unicode モードの ESSG_INIT_T の usApiType フィールドの値として使用されます。

定数	定義	説明
ESSG_API_UTF8	0x0003	この値によって、Essbase サーバーは Unicode モードのアプリケーションの作成または移行が行えるようになります。
ESSG_API_NONUNICODE	0x0002	この値にすると、Essbase サーバーで Unicode モード・アプリケーションを作成および移行できなくなります。

C グリッド API のデータ型

データ型	Essbase 型
typedef char ESSG_APPLICATION_T[ESSG_APPLICATIONLEN];	ESSG_APPLICATION_T
typedef unsigned char ESSG_BOOL_T;	ESSG_BOOL_T
typedef char ESSG_CHAR_T;	ESSG_CHAR_T
typedef char ESSG_DATABASE_T[ESSG_DATABASELEN];	ESSG_DATABASE_T
typedef double ESSG_DOUBLE_T;	ESSG_DOUBLE_T
typedef ESSG_PVOID_T ESSG_DTHINST_T, *ESSG_PDTHINST_T	ESSG_DTHINST_T, ESSG_PDTHINST_T
typedef float ESSG_FLOAT_T;	ESSG_FLOAT_T
typedef ESSG_PVOID_T ESSG_HANDLE_T, *ESSG_PHANDLE_T;	ESSG_HANDLE_T, ESSG_PHANDLE_T
typedef ESSG_PVOID_T ESSG_HGRID_T,*ESSG_PHGRID_T;	ESSG_HGRID_T, ESSG_PHGRID_T
typedef long ESSG_LONG_T;	ESSG_LONG_T
typedef char ESSG_PASSWORD_T[ESSG_PASSWORDLEN];	ESSG_PASSWORD_T
typedef char *ESSG_PSTR_T;	ESSG_PSTR_T
typedef ESSG_VOID_T *ESSG_PVOID_T;	ESSG_PVOID_T

データ型	Essbase 型
typedef char ESSG_SERVER_T[ESSG_SERVERLEN];	ESSG_SERVER_T
typedef short ESSG_SHORT_T;	ESSG_SHORT_T
typedef char *ESSG_STR_T;	ESSG_STR_T
typedef long ESSG_STS_T;	ESSG_STS_T
typedef unsigned char ESSG_UCHAR_T;	ESSG_UCHAR_T
typedef unsigned long ESSG_ULONG_T;	ESSG_ULONG_T
typedef char ESSG_USERNAME_T[ESSG_USERNAMELEN];	ESSG_USERNAME_T
typedef unsigned short ESSG_USHORT_T;	ESSG_USHORT_T
typedef void ESSG_VOID_T;	ESSG_VOID_T
typedef unsigned short ESSG_WORD_T;	ESSG_WORD_T

ESSG_PFUNC_T, ESSG_PFUNC_M

これらの型を使用して、ユーザーのメッセージ・コールバック関数のプロトタイプを定義します。

```
   #ifdef WIN32
#define ESSG_CALLBACK _export
#define ESSG_FUNC_M  ESSG_STS_T ESSG_CALLBACK /* for Win32 */
#else
#define ESSG_CALLBACK _export
#define ESSG_FUNC_M  ESSG_STS_T ESSG_CALLBACK /* for other platforms */
#endif

#ifdef WIN32
/* function pointer (Win32) */
typedef
ESSG_STS_T (ESSG_CALLBACK *ESSG_PFUNC_T)(ESSG_PVOID_T,ESSG_LONG_T,
 SSG_USHORT_T, ESSG_STR_T, ESSG_STR_T);
#else
/* function pointer (other) */
typedef
ESSG_STS_T (ESSG_CALLBACK *ESSG_PFUNC_T)(ESSG_PVOID_T, ESSG_LONG_T,
 ESSG_USHORT_T, ESSG_STR_T, ESSG_STR_T);
#endif
```

C グリッド API の構造体

このセクションでは、グリッド API で使用される構造体について説明します。次の構造体名のいずれかをクリックすると、説明に移動します。

- 1094 ページの「ESSG_CONNECTINFO_T」
- 1094 ページの「ESSG_DATA_T」

ESSG_CONNECTINFO_T

リンクされた各パーティションのデータベース接続についての情報が含まれます。次にフィールドについて説明します:

```
   typedef struct ESSG_CONNECTINFO_T
{
 ESSG_SERVER_T    Server;
 ESSG_APPLICATION_T  Application;
 ESSG_DATABASE_T   Database;
 ESSG_USERNAME_T   Username;
 ESSG_PASSWORD_T   Password;
} ESSG_CONNECTINFO_T, * ESSG_PCONNECTINFO_T, ** ESSG_PPCONNECTINFO_T;
```

データ型	フィールド	説明
ESSG_SERVER_T	Server	サーバー名
ESSG_APPLICATION_T	Application	アプリケーション名
ESSG_DATABASE_T	DatabaseNNN	Essbase データベース名
ESSG_USERNAME_T	Username	ユーザー名
ESSG_PASSWORD_T	Password	ユーザーのパスワード

ESSG_DATA_T

Essbase グリッド API で送受信するデータのフォーマットを記述します。この構造体を戻す呼出しでは、メンバー構造体内のメンバー名が戻されます。データ型が ESSG_DT_MEMBER の場合、呼出し側は pszSt フィールドではなくメンバー構造体を使用して API に同じ構造体を戻すことができます。

ESSG_DATA_T データ構造体は、グリッド API を介して送信または戻される各セルを定義します。この構造体が呼出し側に戻される場合、pszStr に文字列データが含まれ dblData に数値データが含まれます。usType フィールドは、セルがメンバー、数値、またはテキストのいずれかを判別するために使用します。同様に、

構造体を API に渡す場合は、pszStr にメンバー名またはテキストを含める必要があり、dblData に数値データを含める必要があります。usType フィールドは、セルのデータ型にあわせて設定します。セルのデータ型が不明な場合は、データ型をテキスト(ESSG_DT_STRING)に設定すると、サーバー側でデータがメンバーかどうかが判別されます。

```
   typedef struct ESSG_DATA_T
{
 ESSG_PVOID_T   pAttributes;
 ESSG_DATA_VALUE Value;
 ESSG_USHORT_T  usType;
 ESSG_PVOID_T   pCellProps;
} ESSG_DATA_T;

ESS_TSA_API_typedef(ESSG_DATA_T *, ESSG_PDATA_T);
ESS_TSA_API_typedef(ESSG_DATA_T **, ESSG_PPDATA_T);
```

データ型	フィールド	説明
ESSG_PVOID_T	pAttributes	セル・タイプまたはメンバー・タイプを表す次の Long 型の整定数のいずれかになります(OUT)
ESSG_DATA_VALUE_T	Value	戻されたグリッド文字列の値
ESSG_USHORT_T	usType	データ型を表す次のタグ定数のいずれかになります(IN/OUT)
ESSG_PVOID_T	pCellProps	セルがドリルスルー URL に関連付けられているかどうかなどの、セル・プロパティを格納します

ESSG_DATA_T の定数

次の定数は、セルのデータ型を指定するための定数で、ESSG_DATA_T 構造体の pAttributes フィールドで使用します:

```
   ESSG_CA_READONLY
ESSG_CA_READWRITE
ESSG_CA_LINKEDOBJ
ESSG_CA_LINKPARTITION
ESSG_CA_LINKCELLNOTE
ESSG_CA_LINKWINAPP
ESSG_CA_LINKURL
ESSG_CA_AISDT
ESSG_CA_GLDT
```

次の定数は、メンバーのデータ型を指定するための定数で、ESSG_DATA_T 構造体の pAttributes フィールドで使用します:

```
   ESSG_MA_DIMTOP
ESSG_MA_ZOOMINABLE
ESSG_MA_NEVERSHARE
ESSG_MA_LABELONLY
```

```
ESSG_MA_STOREDATA
ESSG_MA_EXPSHARE
ESSG_MA_IMPSHARE
ESSG_MA_DYNCALC
ESSG_MA_FORMULA
ESSG_MA_ATTRIBUTE
ESSG_MA_DIMNUMBITS
```

次の定数は、ESSG_DATA_T 構造体の usType フィールドで使用します:

```
   ESSG_DT_UNUSED
ESSG_DT_STRING
ESSG_DT_LONG
ESSG_DT_DOUBLE
ESSG_DT_BLANK
ESSG_DT_RESERVED
ESSG_DT_ERROR
ESSG_DT_MISSING
ESSG_DT_ZERO
ESSG_DT_NOACCESS
ESSG_DT_MEMBER
ESSG_DT_FORMULA
ESSG_DT_ZEROwFORMULA
ESSG_DT_DOUBLEwFORMULA
ESSG_DT_BLANKwFORMULA
ESSG_DT_STRINGwFORMULA
ESSG_DT_MISSINGwFORMULA
ESSG_DT_NOACCESSwFORMULA
ESSG_DT_STRINGEX
ESSG_DT_MEMBEREX
ESSG_DT_STRINGEXwFORMULA
ESSG_DT_FORMULAEX
ESSG_DT_MEMBERwKEY
```

次の定数は、Unicode モードで操作するための usType フィールドの追加値です。

定数	定義	説明
ESSG_DT_STRINGEX	0x0018	この値は、Unicode モードのために拡張された文字列を指定します。
ESSG_DT_MEMBEREX	0x0019	この値は、Unicode モードのために拡張されたメンバー名を指定します。
ESSG_DT_STRINGEXwFORMULA	0x0002	この値は、Unicode モードのために拡張された式文字列を指定します。
ESSG_DT_FORMULASEX	0x0021	この値は、Unicode モードのために拡張された式を指定します。

ESSG_DRILLDATA_T

特定のセル・アドレスとリンクされたオブジェクトに関連する情報を含みます。
次にフィールドについて説明します:

```
   typedef struct ESSG_DRILLDATA_T
```

```
{
 ESSG_HLRO_T    hLRO;
 ESSG_USHORT_T   usLinkObjType;
 ESSG_LINKOBJDESC  Description;
 ESSG_PSTR_T    pMbrCombos;
 ESSG_ULONG_T    ulNumMbrCombos;
} ESSG_DRILLDATA_T, * ESSG_PDRILLDATA_T, ** ESSG_PPDRILLDATA_T;
```

データ型	フィールド	説明
ESSG_HLRO_T	hLRO	リンクされたオブジェクトの一意のハンドル
ESSG_USHORT_T	usLinkObjType	オブジェクト・タイプ
ESSG_LINKOBJDESC	Description)OOo	オブジェクトの説明
ESSG_PSTR_T	pMbrCombos	メンバー名の配列
ESSG_ULONG_T	ulNumMbrCombos	pMbrCombos に格納するメンバー名の数

ESSG_DTDATA_T

レポート・データ・セルを定義します。

```
   typedef struct ESSG_DTDATA_T
{
 ESSG_ULONG_T row;
 ESSG_ULONG_T column;
 ESSG_CHAR_T  data[ESSG_DESCRIPTION_LEN + 1];
} ESSG_DTDATA_T, *ESSG_PDTDATA_T, **ESSG_PPDTDATA_T;
```

データ型	フィールド	説明
ESSG_ULONG_T	row	指定したデータ・ブロックで 0 から始まるインデックスを付けた行番号
ESSG_ULONG_T	column	指定したデータ・ブロックで 0 から始まるインデックスを付けた列番号
ESSG_CHAR_T	data[ESSG_ DESCRIPTION_LEN + 1]	指定したデータ・ブロックのデータの値

ESSG_DTHEADER_T

特定の列に関するヘッダー情報を定義します。

```
   typedef struct ESSG_DTHEADER_T
{
 ESSG_ULONG_T     colIndex;
 ESSG_CHAR_T      viewName[ESSG_DESCLEN + 1];
 ESSG_CHAR_T      data[ESSG_DESCLEN + 1];
 ESSGDTREPORTDATATYPE dataType;
```

```
} ESSG_DTHEADER_T, *ESSG_PDTHEADER_T, **ESSG_PPDTHEADER_T;
```

データ型	フィールド	説明
ESSG_ULONG_T	colIndex	列の位置を示す 0 から始まるインデックス
ESSG_CHAR_T	viewName[ESSG_DESCLEN_+ 1]	
ESSG_CHAR_T	data[ESSG_DESCLEN_+ 1]	指定したデータ列のヘッダー・テキスト
ESSGDTREPORTDATATYPE	dataType	指定したデータ列のデータ型を表す次の定数のいずれかになります

ESSG_DTHEADER_T の定数

次の定数は、ESSG_DTHEADER_T 構造体のデータ型フィールドで使用します:

```
   ESSGDTINT
ESSGDTFLOAT
ESSGDTSTRING
```

ESSG_DTINFO_T

データ・セルの範囲に関する接続情報を定義します。

```
   typedef struct ESSG_DTINFO_T
{
 ESSG_CHAR_T  hisName[ESSG_FIELDLEN + 1];
 ESSG_CHAR_T  dataSource[ESSG_FIELDLEN + 1];
 ESSG_CHAR_T  username[ESSG_FIELDLEN + 1];
 ESSG_CHAR_T  password[ESSG_FIELDLEN + 1];
 ESSG_USHORT_T inputOption;
} ESSG_DTINFO_T, *ESSG_PDTINFO_T, **ESSG_PPDTINFO_T;
```

データ型	フィールド	説明
ESSG_CHAR_T	hisName[ESSG_FIELDLEN + 1]	
ESSG_CHAR_T	dataSource[ESSG_FIELDLEN + 1]	(読取り専用)
ESSG_CHAR_T	username[ESSG_FIELDLEN + 1]	
ESSG_CHAR_T	password[ESSG_FIELDLEN + 1]	(書込み専用)
ESSG_USHORT_T	inputOption	(読取り専用)

ESSG_DTREPORT_T

レポート定義を定義します。

```
   typedef struct ESSG_DTREPORT_T
{
 ESSG_LONG_T reportId;
 ESSG_CHAR_T name[ESSG_DESCLEN + 1];
 ESSG_LONG_T customize;
 ESSG_LONG_T rowGoverner;
 ESSG_LONG_T timeGoverner;
} ESSG_DTREPORT_T, *ESSG_PDTREPORT_T, **ESSG_PPDTREPORT_T;
```

データ型	フィールド	説明
ESSG_LONG_T	reportId	
ESSG_CHAR_T	name[ESSG_DESCLEN + 1]	
ESSG_LONG_T	customize	
ESSG_LONG_T	rowGoverner	
ESSG_LONG_T	timeGoverner	

ESSG_INIT_T

EssGInit 呼出しへ渡す情報を記述します。

```
   typedef struct
{
 ESSG_ULONG_T ulVersion;
 ESSG_ULONG_T ulMaxRows;
 ESSG_ULONG_T ulMaxColumns;
 ESSG_PFUNC_T pfnMessageFunc;
 ESSG_PVOID_T pUserdata;
 ESSG_USHORT_T usApiType;
} ESSG_INIT_T, *ESSG_PINIT_T;
```

データ型	フィールド	説明
ESSG_ULONG_T	ulVersion	ESSG_VERSION に設定します
ESSG_ULONG_T	ulMaxRows	グリッドの最大行数 制限: 65535 行
ESSG_ULONG_T	ulMaxColumns	グリッドの最大列数 制限: 256 列
ESSG_PFUNC_T	pfnMessageFunc	ユーザー定義のメッセージ・コールバック関数へのポインタ
ESSG_PVOID_T	pUserdata	メッセージ・コールバック関数に渡すユーザー・データへのポインタ
ESSG_UGSHORT_T	usApiType	グリッド API のエンコードのタイプ。有効な値は、1092 ページの「Unicode モードのタイプ」を参照してください。

ESSG_LRODESC_T

Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを説明する情報を含んでいます。次にフィールドについて説明します:

```
   typedef struct ESSG_LRODESC_T
{
 ESSG_USHORT_T  usLinkObjType;
 ESSG_USERNAME_T Username;
 ESSG_TIME_T   LastUpdate;
 union
  {
   ESSG_LROINFO_T lroInfo;
   ESSG_CHAR_T   Note[ESSG_LRONOTELEN];
  } lro;
} ESSG_LRODESC_T, *ESSG_LPLRODESC_T;
```

データ型	フィールド	説明
ESSG_ULONG_T	usLinkObjType	オブジェクト・タイプ
ESSG_USERNAME_T	userName	オブジェクトを最後に変更したユーザー名
ESSG_TIME_T	LastUpdate	オブジェクトが最後に変更された日付 ESSG_TIME_T は符号なし long 型として定義されます
ESSG_LROINFO_T	lroInfo	ユニオンによって関連付けられた LRO 情報構造体
ESSG_CHAR_T	Note[ESSG_LRONOTELEN]	ユニオンによって関連付けられたセル・ノート ESSG_LRONOTELEN によって指定されるデフォルトのノートの長さは 599 です。

ESSG_LROINFO_T

Essbase データベースのデータ・セルにリンクされている特定のオブジェクトに関する情報を含んでいます。次にフィールドについて説明します:

```
   typedef struct ESSG_LROINFO_T
{
 ESSG_CHAR_T  ObjName[ESSG_ONAMELEN];
 ESSG_CHAR_T  Desc[ESS_DESCLEN];
} ESSG_LROINFO_T, *ESSG_LPLROINFO_T;
```

データ型	フィールド	説明
ESSG_CHAR_T	objName[ESSG_ONAMELEN]	データ・セルにリンクされているオブジェクトのソース・ファイル名 ESSG_ONAMELEN はオブジェクト名の最大長を指定します; デフォルト値は 511 です。

データ型	フィールド	説明
ESSG_CHAR_T	Desc[ESS_DESCLEN]	データ・セルにリンクされているオブジェクトの説明 ESS_DESCLEN は説明の最大長を指定します; デフォルト値は 79 です。

ESSG_RANGE_T

受け渡しするデータ範囲を記述します。

```
   typedef struct
{
  ESSG_ULONG_T ulRowStart;
  ESSG_ULONG_T ulColumnStart;
  ESSG_ULONG_T ulNumRows;
  ESSG_ULONG_T ulNumColumns;
} ESSG_RANGE_T, *ESSG_PRANGE_T;
```

データ型	フィールド	説明
ESSG_ULONG_T	ulRowStart	レポートの 1 行目(0 から始まる)
ESSG_ULONG_T	ulColumnStart	レポートの 1 列目(0 から始まる)
ESSG_ULONG_T	ulNumRows	レポートの行数(最大 16370)
ESSG_ULONG_T	ulNumColumns	レポートの列数(最大 256)

15 CグリッドAPI関数リファレンス

「目次」ペインで、EssG が前に付いた C グリッド API 関数のアルファベット順リストを参照してください。

EssGBeginConditionalRetrieve

この関数は、条件付き取得操作を開始します。

構文

```
      ESSG_FUNC_M
      EssGBeginConditionalRetrieve
       (
      hGrid, pszConditions,
ulOptions
      );
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
pszConditions	ESSG_STR_T	取得条件に関連する Essbase レポート指定コマンドが含まれた文字列(64K 以下)。レポート・ライターのメンバー名/別名/一意の名前処理フォーマット・コマンドを pszConditions パラメータに使用しないでください。EssGSetGridOption 関数で利用可能なオプションを使用してください。
ulOptions	ESSG_ULONG_T	取得タイプを示す定数。次のいずれかの値を使用する必要があります: ● ESSG_RET_RETRIEVE 取得のみ ● ESSG_RET_RETRIEVELOCK 取得とロック ● ESSG_RET_LOCKONLY ロックのみ(データを取得しない) 次の値は、ビット OR (|)を使用して ulOptions に追加できます: ESSG_NOATT IBUTES では、pAttributes の値なしでグリッドが戻されます。

備考

- レポート指定の一部に定義した条件は、グリッドに適用されます。
- 戻されたセル値の属性は、第 2 のサーバー要求を使用して入手します。ESSG_NOATTRIBUTES を ulOptions パラメータに渡すことで、サーバーへの要求数が 1 つ少なくなるため、大規模なグリッドの生成では、高速化につながります。

- SmartList、Date、または Format 文字列を使用したアプリケーションなど、タイプに対応したアプリケーションの場合は、テキストにエンコードされたデータが提供されます。ただし、ESSG_NOATTRIBUTES を指定した場合はタイプ情報は提供されません。タイプ情報はメンバー属性と類似の機能をするので、タイプ情報が必要な場合は ESSG_NOATTRIBUTES は使用しないでください。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginConditionalRetrieve(ESSG_HGRID_T hGrid)
{
 ESSG_STS_T   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T  ppDataIn;
 ESSG_PPDATA_T  ppDataOut;
 ESSG_RANGE_T  rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T  ulOptions;
 ESSG_USHORT_T  usState;
 ESSG_STR_T  pszConditions;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic", ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  ulOptions = ESSG_RET_RETRIEVE;
  pszConditions = "<TOP(Scenario,3,@Datacol(3))";
  /* start the conditional retrieve operation */
  sts = EssGBeginConditionalRetrieve(hGrid,
     pszConditions, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn, pDataIn);
 }

 if(sts == 0)
 {
  /* perform the retrieval */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }
 if(sts == 0)
 {
```

```
 /* determine the results of the retrieve */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
    &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
    &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 /* display the results */
 DisplayOutput(ppDataOut, rDataRangeOut);
 /* free the returned data */
 EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
}

if(!sts)
{
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
}
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginConditionalZoomIn

この関数は、条件付きズームインを開始します。

構文

```
      ESSG_FUNC_M
      EssGBeginConditionalZoomIn
       (
      hGrid, pZoomCell,
pszConditions, ulOptions
      );
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
pZoomCell	1101 ページの「ESSG_RANGE_T」	ズーム・インするセルを示します。条件付きズーム・インでは単一のセルである必要があります。

パラメータ	データ型	説明
pszConditions	ESSG_STR_T	ズームイン条件に関する Essbase レポート指定コマンドを含む文字列(64K 以下)。 レポート・ライターのメンバー名/別名/一意の名前処理フォーマット・コマンドを pszConditions パラメータに使用しないでください。EssGSetGridOption で利用可能なオプションを使用してください。
ulOptions	ESSG_ULONG_T	ズームインのタイプ(横または下)を示すビットマスク。次の 2 つの値は互いに排他的です: • **ESSG_ZOOM_DOWN** 選択されたページ次元またはタイトル次元が下方向へズームされます • **ESSG_ZOOM_ACROSS** 選択されたページ次元が左右方向へズームされます 次のオプションを、ビット OR (|)を使用して、ulOptions に追加できます: ESSG_NOATTRIBUTES は、pAttributes 値のないグリッドを戻します。

備考

- ズーム・インするセルは、1 つの範囲で指定します。適用する条件は、Essbase レポート指定コマンドを含む文字列として渡されます。
- 戻されたセル値の属性は、別のサーバー要求を使用して取得されます。ulOptions パラメータに **ESSG_NOATTRIBUTES** を渡すと、サーバーの発行する要求が 1 つ少なくなり、大規模な結果グリッドではより高速になります。
- 条件付きズームインは、一度に複数のズーム・セルでは実行できません。
- 条件付きズームインを行うとき、有効なズーム・レベルは、**ESSG_NEXTLEVEL**、**ESSG_BOTTOMLEVEL** または **ESSG_ALLLEVELS** の 3 つしかありません。条件付きズームインのズーム・レベルは、**EssGSetGridOption** によって、有効な 3 レベルの 1 つに設定する必要があります。条件付きズームインを実行するときに、無効なレベルが設定されると、API はデフォルトの **ESSG_NEXTLEVEL** を使用します。
- ズーム・レベルが **ESSG_BOTTOMLEVEL** である場合、メンバーは、ズーム・インしている次元における 0 レベルのすべてのメンバーから、条件に基づいて選択されます。たとえば、ズーム・セルが Market 次元からの East を含んでおり、ズーム・レベルが **ESS_BOTTOMLEVEL** である場合、選択されるメンバーは、East の子孫のみでなく、Market のリーフ・メンバーのいずれかの可能性があります。
- SmartList、Date または Format 文字列を使用するアプリケーションなど、Type 対応のアプリケーションの場合には、**ESSG_NOATTRIBUTES** を指定すると、タイプ情報なしのテキスト・エンコードされたデータを取得します。タイプ情報は、メンバー属性と同様に機能します。したがって、タイプ情報が必要な場合は、**ESSG_NOATTRIBUTES** を使用しないでください。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginConditionalZoomIn(ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T     ppDataIn;
 ESSG_PPDATA_T     ppDataOut;
 ESSG_RANGE_T     rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_RANGE_T     pZoomCells;
 ESSG_USHORT_T     usState;
 ESSG_STR_T      pszConditions;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic", ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  ulOptions = ESSG_ZOOM_DOWN | ESSG_ALLLEVELS;

  pZoomCells.ulRowStart = 0;
  pZoomCells.ulColumnStart = 2;
  pZoomCells.ulNumRows = 1;
  pZoomCells.ulNumColumns = 1;
  pszConditions = "<TOP("Scenario",3,@Datacol(3))";

  /* start the conditional zoom-in  operation */
  sts = EssGBeginConditionalZoomIn(hGrid,
     &pZoomCells, pszConditions, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn,
     ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the conditional zoom-in */
  sts = EssGPerformOperation(hGrid, 0);

  /* Free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }
 if(sts == 0)
 {
  /* determine the results of conditional zoom-in */
```

```
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
 }
 if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }

 if(sts == 0)
 {
  DisplayOutput(ppDataOut, rDataRangeOut);
  /* free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 }

if(sts == 0)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }

}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginCreateLRO

Essbase データベースのデータ・セルに対してリンク・オブジェクトを作成する操作を開始します。

構文

```
      ESSG_FUNC_M
      EssGBeginCreateLRO
       (
      hGrid, usCells, pCells, pLroDesc,
ulOption
      );
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
usCells;	ESSG_USHORT_T	pCells で指定されたセル範囲の数。
pCells;	1101 ページの「ESSG_RANGE_T」	リンクを作成するセル範囲の配列。
pLroDesc;	1100 ページの「ESSG_LRODESC_T」	新規オブジェクトの LRO 記述情報。

パラメータ	データ型	説明
ulOption;	ESSG_ULONG_T	オブジェクトをサーバーに保管するかどうかを指定するオプション。サーバーに winapp や URL オブジェクトを保管するには ESS_STORE_OBJECT_API を使用します。セル・ノートをサーバー以外(インデックス・ファイル)に保管するには ESS_NOSTORE_OBJECT を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRODesc
- EssGGetLRO
- EssGUpdateLRO

EssGBeginDataPoint

この関数は、データ・ポイント操作を開始します。

構文

```
ESSG_FUNC_M
EssGBeginDataPoint
 (
hGrid, ulRow, ulColumn, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulRow	ESSG_ULONG_T	データ・ポイントの行。
ulColumn	ESSG_ULONG_T	データ・ポイントの列。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

- この関数では、グリッドの特定のセルに対応するメンバーの組合せを表すメンバーが各次元から 1 つずつ戻されます。

- 呼出し元は **EssGSendRows** に、Essbase がセルのメンバーを判断できる十分な情報を渡します。ulRow パラメータおよびすべての列以下のすべての行を渡すのが最も安全です。ulRow および ulColumn の値はゼロベースです。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginDataPoint(ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M  sts = ESS_STS_NOERR;
 ESSG_ULONG_T     ulRow;
 ESSG_ULONG_T     ulColumn;
 ESSG_ULONG_T     ulOptions;
 ESSG_PPDATA_T    ppDataIn;
 ESSG_RANGE_T     rDataRangeIn;
 ESSG_ULONG_T     ulMembers, i;
 ESSG_PSTR_T     ppszMembers;
 ESSG_USHORT_T    usState;

  /* connect the grid to a database on the server */
  sts = EssGConnect(hGrid, "Rainbow", "Admin",
     "Password", "Demo", "Basic", ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  ulRow = 1;
  ulColumn = 2;
  ulOptions = 0;

  /* start the data point operation */
  sts = EssGBeginDataPoint(hGrid, ulRow, ulColumn, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn,
     ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the data point operation */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
```

```
 }
 if(sts == 0)
 {
  /* determine the results of the data point operation */
  sts = EssGGetDataPointResults(hGrid, &ulMembers,
     &ppszMembers, &usState);
 }

 if(!sts && ulMembers)
 {
  printf("\nMembers:");
   for (i = 0; i<ulMembers; i++)
  printf("\n\t%s", ppszMembers[i]);

  EssGFreeMemberInfo(hGrid, ulMembers, ppszMembers);
 }
 if(!sts)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginDeleteLROs

Essbase データベースのデータ・セルにリンクされたオブジェクトをすべて削除する操作を開始します。

構文

```
ESSG_FUNC_M
EssGBeginDeleteLROs
 (
hGrid, usCells, pCells
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
usCells;	ESSG_USHORT_T	pCells で指定されたセル範囲の数。
pCells;	1101 ページの「ESSG_RANGE_T」	リンク・オブジェクトを削除するセル範囲の配列。

備考

単一の LRO を削除するには、EssGDeleteLRO を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginCreateLRO
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRODesc
- EssGGetLRO
- EssGUpdateLRO

EssGBeginDrillAcross

リンク・パーティションからセルを取得するためにドリルアクロスを開始します。

構文

```
      ESSG_FUNC_M
      EssGBeginDrillAcross
       (
      hGrid, hDAGrid, hLRO, usCells,
pDrillCells, usOption
      );
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるオリジナル・グリッドのハンドル。
hDAGrid;	ESSG_HGRID_T	ドリル結果を取得する新規グリッドのハンドル。
hLRO;	ESSG_HLRO_T	リンク・パーティションのハンドル。
usCells;	ESSG_USHORT_T	pDrillCells で指定されたセル範囲の数。
pDrillCells;	1101 ページの「ESSG_RANGE_T」	リンク・パーティションに関連付けられたセル範囲の配列。
ulOption;	ESSG_ULONG_T	サーバーからズームイン結果が送信された場合、これを戻すかどうかを指定するオプション。次のオプションを使用します: • ESSG_OPT_ZOOM: ズームイン結果を戻します。 • ESSG_OPT_NOZOOM: ズームイン結果を戻しません。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」

- 1093 ページの「C グリッド API の構造体」
- EssGBeginRemoveOnly
- EssGGetCellLinkResults
- EssGBeginDrillOrLink

EssGBeginDrillOrLink

Essbase データベースの 1 つ以上のデータ・セルに関連付けられたリンクのクエリー操作を開始します。

構文

```
      ESSG_FUNC_M
      EssGBeginDrillOrLink
       (
      hGrid, usCells, pDrillCells,
ulOptions
      );
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
usCells;	ESSG_USHORT_T	pDrillCells で指定した範囲の配列にあるセル範囲の数。
pDrillCells;	1101 ページの「ESSG_RANGE_T」	リンクについてクエリーを行うセル範囲の配列。
ulOptions;	ESSG_ULONG_T	サーバーからズームイン結果が送信された場合、これを戻すかどうかを指定するオプション。次のオプションを使用します: • ESSG_OPT_ZOOM: ズームイン結果を戻します。 • ESSG_OPT_NOZOOM: ズームイン結果を戻しません。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginRemoveOnly
- EssGGetCellLinkResults
- EssGBeginDrillAcross
- EssGFreeCellLinkResults
- EssGGetCellLinkResults

EssGBeginKeepOnly

選択項目のみ保持の操作を開始し、保持対象のセルを分離します。これにより、その他のセルはすべて削除されます。

構文

```
ESSG_FUNC_M
EssGBeginKeepOnly
(
hGrid, usCells, pKeepCells, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
usCells	ESSG_USHORT_T	pKeepCells のセル範囲の数(配列のサイズ)。
pKeepCells	1101 ページの「ESSG_RANGE_T」	保持するセルを指定します。保持するメンバーの指定は次元ごとに行います。したがって、たとえば"Qtr1"を保持する場合、Time 次元の他のすべてのメンバーが削除され、"Qtr1"が Time 次元を唯一示すものになります。レポート内の他の次元は変更されません。これはセル範囲の 1 次元配列です。 1 つの次元のメンバーを 1 つ以上指定できます。複数の次元も指定できます。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

保持するセルはセル範囲の 1 次元配列として指定します。保持するアイテムの指定は、次元ごとに行われます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginKeepOnly (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T    ppDataIn;
 ESSG_PPDATA_T    ppDataOut;
 ESSG_RANGE_T    rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T    ulOptions;
 ESSG_USHORT_T    usCells;
 ESSG_RANGE_T    pKeepCells;
 ESSG_USHORT_T    usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);
```

```
if(sts == 0)
{
 ppDataIn = BuildTable(&rDataRangeIn);

 pKeepCells.ulRowStart = 1;
 pKeepCells.ulColumnStart = 0;
 pKeepCells.ulNumRows = 1;
 pKeepCells.ulNumColumns = 1;
 ulOptions = 0;
 usCells = 1;

 /* start the keep-only operation */
 sts = EssGBeginKeepOnly(hGrid, usCells,
    &pKeepCells, ulOptions);
}

if(sts == 0)
{
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn,
    ppDataIn);
}

if(sts == 0)
{
 /* perform the keep-only operation */
 sts = EssGPerformOperation(hGrid, 0);

 /* free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
 /* determine the results of the keep-only operation */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
    &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
    &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 DisplayOutput(ppDataOut, rDataRangeOut);
 /* Free the returned data */
 EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
}

if(!sts)
{
  EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
}
```

```
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginLock

この関数はデータベースのブロックをロックします。

構文

```
ESSG_FUNC_M
EssGBeginLock
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

- ulOptions パラメータに **ESSG_RET_LOCKONLY** を指定して **EssGRetrieve** 関数を呼び出す場合と同一の機能を果します。
- 呼出し元にはデータが戻されません。
- この操作では、行を取得する必要はありません。**EssGSendRows** と **EssGPerformOperation** を呼び出すのみで十分です。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginLock (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T     ppDataIn;
 ESSG_RANGE_T      rDataRangeIn;
 ESSG_ULONG_T      ulOptions;

  /* connect the grid to a database on the server */
  sts = EssGConnect(hGrid, "Rainbow", "Admin",
```

```
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);

if(sts == 0)
{
 ppDataIn = BuildTable(&rDataRangeIn);

 /* start the lock operation */
 ulOptions = 0;
 sts = EssGBeginLock(hGrid, ulOptions);
}

if(sts == 0)
{
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn,
    ppDataIn);
}

if(sts == 0)
{
 /* perform the lock operation */
 sts = EssGPerformOperation(hGrid, 0);

 /* Free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}

if(!sts)
{
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
}
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginPivot

この関数はピボットを開始します。

構文

```
ESSG_FUNC_M
EssGBeginPivot
 (
hGrid, pStartCell, pEndCell, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
pStartCell	1101 ページの「ESSG_RANGE_T」	ピボットが発生するセルを説明します。このセルのメンバーは、ピボットが行われる次元を示します。このパラメータは NULL にはできません。
pEndCell	1101 ページの「ESSG_RANGE_T」	次元を配置するセルを説明します。このパラメータの値が NULL の場合、次元メンバーについての行から列へのピボット、または列から行へのピボットを示します。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

ピボットの開始セルと宛先セルは、呼出し元で指定します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginPivot (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T    ppDataIn;
 ESSG_PPDATA_T    ppDataOut;
 ESSG_RANGE_T     rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_RANGE_T     pStartCell;
 ESSG_RANGE_T     pEndCell;
 ESSG_USHORT_T    usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  pStartCell.ulRowStart = 0;
  pStartCell.ulColumnStart = 3;
  pStartCell.ulNumRows = 1;
  pStartCell.ulNumColumns = 1;

  pEndCell.ulRowStart = 1;
  pEndCell.ulColumnStart = 1;
  pEndCell.ulNumRows = 1;
  pEndCell.ulNumColumns = 1;
```

```
  ulOptions = 0;

  /* start the pivot operation */
  sts = EssGBeginPivot(hGrid,&pStartCell, &pEndCell, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the pivot operation */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }
 if(sts == 0)
 {
  /* determine the results of the pivot operation */
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
 }
 if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }

 if(sts == 0)
 {
  DisplayOutput(ppDataOut, rDataRangeOut);
  /* free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 }

 if(!sts)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginRemoveOnly

メンバーの削除を開始し、削除対象のセルを切り離します。

構文

```
ESSG_FUNC_M
EssGBeginRemoveOnly
 (
hGrid, usCells, pRemoveCells, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
usCells	ESSG_USHORT_T	pRemoveCells のセル範囲の数(配列のサイズ)。
pRemoveCells	1101 ページの「ESSG_RANGE_T」	削除するセルを指定します。削除するメンバーの指定は次元ごとに行われます。したがって、「Qtr1」を削除する場合、Time 次元の他のすべてのメンバーは保持されます。レポート内の他の次元は変更されません。これはセル範囲の 1 次元配列です。 1 つの次元のメンバーを 1 つ以上指定できます。複数の次元も指定できます。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

削除対象のセルは、セル範囲の 1 次元配列で指定されます。削除するアイテムの指定は次元ごとに行われます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

例

```
    ESSG_VOID_T ESSG_BeginRemoveOnly (ESSG_HGRID_T hGrid)

{
 ESSG_STS_T   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T    ppDataIn;
 ESSG_PPDATA_T    ppDataOut;
 ESSG_RANGE_T     rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_USHORT_T    usCells;
 ESSG_RANGE_T     pRemoveCells;
 ESSG_USHORT_T    usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  pRemoveCells.ulRowStart = 1;
```

```
  pRemoveCells.ulColumnStart = 0;
  pRemoveCells.ulNumRows = 1;
  pRemoveCells.ulNumColumns = 1;
  ulOptions = 0;
  usCells = 1;

  /* start the remove-only operation */
  sts = EssGBeginRemoveOnly(hGrid, usCells,
     &pRemoveCells, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn,
     ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the remove-only operation */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }
 if (sts == 0)
 {
  /* determine the results of the remove-only operation */
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
     &usState);
 }
 if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }

 if(sts == 0)
 {
  DisplayOutput(ppDataOut, rDataRangeOut);
  /* Free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 }

 if(!sts)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」

- 1093 ページの「C グリッド API の構造体」
- EssGGetCellLinkResults
- EssGBeginDrillAcross
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetCellLinkResults

EssGBeginReport

サーバーでレポート・スクリプトを実行します。

構文

```
 ESSG_FUNC_M
EssGBeginReport
 (
hGrid, pszReportIn, ulOptions
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
pszReportIn	ESSG_STR_T	Essbase のレポート指定が含まれた文字列(64K 以下)。
ulOptions	ESSG_ULONG_T	戻されるグリッド・オプションを記述したビットマスク。有効な値は次のとおりです: **ESSG_NOATTRIBUTES** は pAttributes の値なしでグリッドを戻します。

備考

- 結果は 2 次元のセル配列で戻されます。
- この操作では、行を送信する必要はありません。**EssGPerformOperation**、**EssGGetResults**、および **EssGGetRows** を呼び出すだけで十分です。
- 戻されたセル値の属性は、別のサーバー要求を使用して取得されます。ulOptions パラメータに **ESSG_NOATTRIBUTES** を渡すと、サーバーの発行する要求が 1 つ少なくなり、大規模な結果グリッドではより高速になります。
- グリッド API を介してサーバーに渡されたレポートでは、必ずタブで区切られたレポート・フォーマットを戻す要求をする必要があります{TABDELIM}。タブで区切られていないレポート・フォーマットが戻されると、グリッド API が生成されたレポートをグリッドに変換できない場合があります。
- #Missing の別名に使用される文字列がレポート指定により変更されると、欠落セルは文字列タイプ(**ESSG_DT_STRING**)として戻されます。またこのとき、新しい#Missing の別名は、**ESSG_DT_MISSING** セルではなくテキストとして一緒に戻されます。

- **EssGBeginReport()**および他のレポート関数を呼び出すクライアント・プログラムでは、新しい 1093 ページの「C グリッド API の構造体」および 1092 ページの「C グリッド API のデータ型」(特に StringEx と MemberEx)を考慮する必要があります。古いプログラムを新しいサーバーで使用するには、変更する必要があります。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginReport (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T    ppDataOut;
 ESSG_RANGE_T     rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_STR_T      pszReportIn;
 ESSG_USHORT_T    usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin", "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);
  if(sts == 0)
 {   pszReportIn = "{TabDelim}<idesc Year !";
  ulOptions = ESSG_NOATTRIBUTES;
    sts = EssGBeginReport(hGrid, pszReportIn,
        ulOptions);
 }
 if(sts == 0)
 {
  /* perform the report */
  sts = EssGPerformOperation(hGrid, 0);
 }
 if(sts == 0)
 {
  /* determine the results of the report */
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
     &usState);
 }
  if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }
  if(sts == 0)
 {
  DisplayOutput (ppDataOut, rDataRangeOut);
  /* Free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
```

```
 }
   if(!sts)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginReportFile

サーバーでレポート・ファイルを実行します。

構文

```
ESSG_FUNC_M
EssGBeginReportFile
 (
hGrid, pszReportName, bLocal, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
pszReportName	ESSG_STR_T	実行するレポートの名前。このレポートがサーバー上にある場合、**APPLICATION¥DATABASE** ディレクトリにあります。ローカルの場合は、この文字列にはレポートの絶対パス名が含まれます。
bLocal	ESSG_BOOL_T	レポートがローカルに存在しているのかどうかを示すブール値。TRUE の値はローカルに、FALSE の値はサーバー上に存在していることを示します。
ulOptions	ESSG_ULONG_T	戻されるグリッド・オプションを記述したビットマスク。有効な値は次のとおりです: **ESSG_NOATTRIBUTES** は pAttributes の値なしでグリッドを戻します

備考

- 結果は 2 次元のセル配列で戻されます。
- この操作では、行を送信する必要はありません。**EssGPerformOperation**、**EssGGetResults**、および **EssGGetRows** を呼び出すだけで十分です。
- 戻されたセル値の属性は、別のサーバー要求を使用して取得されます。ulOptions パラメータに **ESSG_NOATTRIBUTES** を渡すと、サーバーの発行する要求が 1 つ少なくなり、大規模な結果グリッドではより高速になります。
- グリッド API を介してサーバーに渡されたレポートでは、必ずタブで区切られたレポート・フォーマットを戻す要求をする必要があります{TABDELIM}。

タブで区切られていないレポート・フォーマットが戻されると、グリッド API が生成されたレポートをグリッドに変換できない場合があります。

- #Missing の別名に使用される文字列がレポート指定により変更されると、欠落セルは文字列タイプ(**ESSG_DT_STRING**)として戻されます。また、このとき、新しい#Missing セルの別名は、**ESSG_DT_MISSING** セルではなくテキストとして一緒に戻されます。
- ローカル以外(サーバー上)のレポート・ファイル・オブジェクトでは、pszReportName パラメータにファイル拡張子を使用しないでください。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginReportFile (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T     ppDataOut;
 ESSG_RANGE_T      rDataRangeOut;
 ESSG_ULONG_T      ulOptions;
 ESSG_STR_T       pszReportName;
 ESSG_BOOL_T       bLocal;
 ESSG_USHORT_T     usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);

 if(sts == 0)
 {
  pszReportName = "DescYear";
  bLocal = ESSG_FALSE;
  ulOptions = ESSG_NOATTRIBUTES;

  /*start the report file operation */
  sts = EssGBeginReportFile(hGrid,
     pszReportName,bLocal, ulOptions);
 }

 if(sts == 0)
 {
  /* perform the report operation */
  sts = EssGPerformOperation(hGrid, 0);
 }

 if (sts == 0)
 {
  /* determine the results of the report operation */
```

```
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
 }

 if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }

 if(sts == 0)
 {
  DisplayOutput(ppDataOut, rDataRangeOut);
  /* Free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 }

 if(!sts)
 {
  EssGEndOperation(hGrid, 0);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginRetrieve

この関数は、基本的な取得操作を開始します。

構文

```
ESSG_FUNC_M
EssGBeginRetrieve
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	取得タイプを示す定数。次のいずれかの値を使用する必要があります: • ESSG_RET_RETRIEVE 取得のみ • ESSG_RET_RETRIEVELOCK 取得とロック • ESSG_RET_LOCKONLY ロックのみ(データを取得しません)

備考

- オプションで、後で行が EssGSendRows を介して渡されるときの更新のためにサーバーのブロックをロックします。
- 行を送信しなくても、メンバーとして使用される次元名のみでデフォルトのグリッドを取得できます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_VOID_T ESSG_BeginRetrieve(ESSG_HGRID_T hGrid)
 {
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T  pDataIn;
 ESSG_PPDATA_T  ppDataOut;
 ESSG_RANGE_T  rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T  ulOptions;
 ESSG_USHORT_T  usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic", ESSG_CONNECT_NODIALOG);

if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);
  ulOptions = ESSG_RET_RETRIEVE;
  /* start the retrieve operation */
  sts = EssGBeginRetrieve(hGrid, ulOptions);
 }

if(sts == 0)
  {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);
 }

if(sts == 0)
 {
  /* perform the retrieval */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
   FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }

if(sts == 0)
```

```
  {
   /* determine the results of the retrieve */
   sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
  }

if(!sts && usState == ESSG_STATE_DONE)
  {
   /* get all the data */
   sts = EssGGetRows(hGrid, 0, &rDataRangeOut, &rDataRangeOut,
      &ppDataOut);
  }

if(sts == 0)
  {
   DisplayOutput (ppDataOut, rDataRangeOut);
   /* free the returned data */
   EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
  }

if(!sts)
  {
   EssGEndOperation(hGrid, 0);
   EssGDisconnect(hGrid, 0);
  }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginSamplingZoomIn

ランダムにサンプリングされたズームイン操作を始めます。

構文

```
ESSG_FUNC_M
EssGBeginSamplingZoomIn
 (
hGrid, usCells, pZoomCells, ulSamplingPercentage, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
usCells	ESSG_USHORT_T	pZoomCells のセル範囲の数(配列のサイズ)。
pZoomCells	1101 ページの「ESSG_RANGE_T」	ズーム・インするセルを指定します。これはセル範囲の 1 次元配列です。

パラメータ	データ型	説明
ulSamplingPercentage	ESSG_ULONG_T	サンプリング・レートのパーセンテージ。この数は、1 以上 100 以下の整数です。100 パーセントの深さの場合、次元のすべてのメンバーを取得します。これにより、効果的にサンプリングをオフにして、すべてのメンバーを取得します。ulSamplingPercentage が 50 の場合、メンバーの半分を取得します。
ulOptions	ESSG_ULONG_T	ズームインのタイプ(横または下)およびズームのレベルについて示すビットマスク。次の 2 つの値は互いに排他的です: ● **ESSG_ZOOM_DOWN** 選択されたページ次元またはタイトル次元が下方向へズームされます ● **ESSG_ZOOM_ACROSS** 選択されたページ次元が左右方向へズームされます ulOptions の次のレベル値自身は相互に排他的です: ● **ESSG_NEXTLEVEL** 子 ● **ESSG_ALLLEVELS** すべてのメンバー ● **ESSG_BOTTOMLEVEL** 最下位レベル ● **ESSG_SIBLEVEL** 兄弟レベル ● **ESSG_SAMELEVEL** 同一レベル ● **ESSG_SAMEGENERATION** 同世代 ● **ESSG_CALCLEVEL** 計算 ● **ESSG_OPTIONS** グリッド・オプションの設定を使用 ビット単位 OR (|)を使用して、ulOptions を指定します。たとえば、ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL とします

備考

- ズーム・インするセルは、セル範囲の 1 次元配列として指定します。
- この関数は、標準のグリッド・ズームイン関数 **EssGBeginZoomIn()**とは異なります。この関数には、パーセンテージでサンプリングの深さを設定する引数があります。100 パーセントの深さは、次元のすべてのメンバーを取得します。また、50 パーセントの深さは、メンバーの半分を取得します。この関数は、大きな次元または非常に密な次元でズーム・インするのに特に役立ちます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginZoomIn
- EssGBeginConditionalZoomIn

- EssGBeginConditionalZoomIn

EssGBeginUpdate

サーバーのデータの更新を開始します。この関数は呼出し元にデータを戻しません。

構文

```
ESSG_FUNC_M
EssGBeginUpdate
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
ulOptions	ESSG_ULONG_T	ブロックを更新前にロックする必要があるかどうかを指定する定数。次の相互に排他的な値のいずれかの値を使用する必要があります: • **ESSG_REQUIRELOCK** ブロックが先にロックされていない場合、更新できません。 • **ESSG_LOCKIFNEEDED** ブロックが先にロックされていない場合、ロックしてから更新します。

備考

EssGPerformOperation を呼び出して操作を完了すると、このブロックのロックが解除されます。ブロックのロックを維持する場合は、**EssGSetGridOptions** の「更新モード」オプションを TRUE に設定します。この操作では行を取得する必要はありません。**EssGSendRows** と **EssGPerformOperation** を呼び出すのみで十分です。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginUpdate (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T     ppDataIn;
 ESSG_RANGE_T      rDataRangeIn;
 ESSG_ULONG_T      ulOptions;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic", ESSG_CONNECT_NODIALOG);
```

```
 if(sts == 0)
 {
  ppDataIn = BuildTable (&rDataRangeIn);

  ulOptions = ESSG_LOCKIFNEEDED;
  /* start the update operation */
  sts = EssGBeginUpdate(hGrid, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the update */
  sts = EssGPerformOperation(hGrid, 0);

  /* free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }

 if(!sts)
 {
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginZoomIn

この関数は、ズームインを開始します。

構文

```
ESSG_FUNC_M
EssGBeginZoomIn
 (
hGrid, usCells, pZoomCells, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。

パラメータ	データ型	説明
usCells	ESSG_USHORT_T	pZoomCells のセル範囲の数(配列のサイズ)。
pZoomCells	1101 ページの「ESSG_RANGE_T」	ズーム・インするセルを指定します。これはセル範囲の 1 次元配列です。
ulOptions	ESSG_ULONG_T	ズームインのタイプ(横または下)およびズームのレベルについて示すビットマスク。次の 2 つの値は互いに排他的です: ● **ESSG_ZOOM_DOWN**- 選択したページ/タイトルが下方向にズームされます ● **ESSG_ZOOM_ACROSS**- 選択したページ次元が左右方向にズームされます ulOptions の次のレベル値自身は相互に排他的です: ● **ESSG_NEXTLEVEL**- 子 ● **ESSG_ALLLEVELS**- すべてのメンバー ● **ESSG_BOTTOMLEVEL**- 最下位レベル ● **ESSG_SIBLEVEL**- 兄弟レベル ● **ESSG_SAMELEVEL**- 同一レベル ● **ESSG_SAMEGENERATION**- 同世代 ● **ESSG_CALCLEVEL**- 計算 ● **ESSG_OPTIONS**- グリッド・オプションの設定を使用 ビット単位 OR (|)を使用して、ulOptions を指定します。たとえば、ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL とします

備考

ズーム・インするセルは、セル範囲の 1 次元配列として指定します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginZoomIn (ESSG_HGRID_T hGrid)
{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T    ppDataIn;
 ESSG_PPDATA_T    ppDataOut;
 ESSG_RANGE_T     rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_USHORT_T    usCells;
 ESSG_RANGE_T     pZoomCells;
 ESSG_USHORT_T    usState;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic", ESSG_CONNECT_DEFAULT);
```

```
 if(sts == 0)
 {
  ppDataIn = BuildTable(&rDataRangeIn);

  ulOptions = ESSG_ZOOM_DOWN | ESSG_ALLLEVELS;

  pZoomCells.ulRowStart = 0;
  pZoomCells.ulColumnStart = 2;
  pZoomCells.ulNumRows = 1;
  pZoomCells.ulNumColumns = 1;
  usCells = 1;

  /* start the zoom in operation */
  sts = EssGBeginZoomIn(hGrid, usCells, &pZoomCells, ulOptions);
 }

 if(sts == 0)
 {
  /* send the entire grid to define the query */
  sts = EssGSendRows(hGrid, &rDataRangeIn,
     ppDataIn);
 }

 if(sts == 0)
 {
  /* perform the zoom-in */
  sts = EssGPerformOperation(hGrid, 0);

  /* Free the built data */
  FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
 }

if (sts == 0)
 {
  /* determine the results of the zoom-in */
  sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
     &usState);
 }

 if(sts ==0)
 {
  /* get all the data */
  sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
     &rDataRangeOut, &ppDataOut);
 }

 if(sts == 0)
 {
  DisplayOutput(ppDataOut, rDataRangeOut);
  /* Free the returned data */
  EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 }

 if( !sts)
 {
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
```

```
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGBeginZoomOut

ズームアウトを開始します。

構文

```
ESSG_FUNC_M
EssGBeginZoomOut
 (
hGrid, usCells, pZoomCells, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
usCells	ESSG_USHORT_T	pZoomCells のセル範囲の数のカウント(配列のサイズ)。
pZoomCells	1101 ページの「ESSG_RANGE_T」	ズーム・アウトするセルを指定します。これはセル範囲の 1 次元配列です。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_BeginZoomOut (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_PPDATA_T     ppDataIn;
 ESSG_PPDATA_T     ppDataOut;
 ESSG_RANGE_T      rDataRangeIn, rDataRangeOut;
 ESSG_ULONG_T     ulOptions;
 ESSG_USHORT_T     usCells;
 ESSG_RANGE_T     pZoomCells;
 ESSG_USHORT_T     usState;

 /* connect the grid to a database on the server */
```

```
sts = EssGConnect(hGrid, "Rainbow", "Admin",
   "Password", "Demo", "Basic",
   ESSG_CONNECT_DEFAULT);

if(sts == 0)
{
 ppDataIn = BuildTable(&rDataRangeIn);

 pZoomCells.ulRowStart = 1;
 pZoomCells.ulColumnStart = 1;
 pZoomCells.ulNumRows = 1;
 pZoomCells.ulNumColumns = 1;
 ulOptions = 0;
 usCells = 1;

 /* start the zoom out operation */
 sts = EssGBeginZoomOut(hGrid, usCells,
    &pZoomCells, ulOptions);
}

if(sts == 0)
{
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn,
    ppDataIn);
}

if(sts == 0)
{
 /* perform the zoom-out */
 sts = EssGPerformOperation(hGrid, 0);

 /* Free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
 /* determine the results of the zoom-out */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut,
    &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
    &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 DisplayOutput(ppDataOut, rDataRangeOut);
 /* free the returned data */
 EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
}

if(!sts)
{
```

```
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGCancelOperation

この関数は、操作のどの段階でも操作を取り消します。

構文

```
ESSG_FUNC_M
EssGCancelOperation
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

- この関数は EssGBeginXxx を呼び出した後、いつでも呼び出せます。
- 現在の操作が取り消され、すべてのリソースが解放されます。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T EssGCancelOperation (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_ULONG_T     ulOptions;
 ESSG_STR_T      pszReportIn;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);
```

```
 if(sts == 0)
 {
  pszReportIn = "{TabDelim}<idesc Year !";
  ulOptions = ESSG_NOATTRIBUTES;
  sts = EssGBeginReport(hGrid, pszReportIn,
     ulOptions);
 }

 if(sts == 0)
 {
  ulOptions = 0;
  sts = EssGCancelOperation(hGrid, ulOptions);
 }

 if(!sts)
 {
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGCell

この関数は、単独のデータポイントを表現する 1 つの値をサーバーから取得します。

構文

```
ESSG_FUNC_M
EssGCell
 (
hGrid, usCount, pszMbrs, pDataCell
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
usCount	ESSG_USHORT_T	送信対象メンバーの数。**EssGCell** がレポートできる次元の最大数は 20 です。
pszMbrs	ESSG_PSTR_T	クエリー対象のメンバー名の配列。次元ごとに 1 つのみ代表が許可されます。
pDataCell	1094 ページの「ESSG_DATA_T」	サーバーから戻される値。

備考

- 指定できるメンバーの最大数は、次のとおりです:
  - 20 メンバー。
  - 1 次元につき 1 メンバー。
- 次元に対してメンバーを指定しない場合、デフォルトとして最上位レベル(次元)のメンバーが使用されます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_Cell (ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_USHORT_T    usCount;
 ESSG_DATA_T     DataCell;
 ESSG_CHAR_T     *pszMbrs[5] = { "Actual", "Jan",
                "West", "Audio",
                "Sales"};

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
         "Password", "Demo", "Basic",
         ESSG_CONNECT_NODIALOG);

 /* retrieve cell value */
 usCount = 5;
 if(sts == 0)
  sts = EssGCell(hGrid, usCount, pszMbrs,&DataCell);

 if(!sts)
 {
  switch(DataCell.usType)
  {
   case(ESSG_DT_STRING):
    printf("%s", DataCell.Value.pszStr+1);
    break;
   case(ESSG_DT_LONG):
    printf("%ld", DataCell.Value.lData);
    break;
   case(ESSG_DT_DOUBLE):
    printf("%g", DataCell.Value.dblData);
    break;
   case(ESSG_DT_BLANK):
    break;
   case(ESSG_DT_RESERVED):
    printf("#Reserved");
    break;
```

```
    case(ESSG_DT_ERROR):
     printf("#Error");
     break;
    case(ESSG_DT_MISSING):
     printf("#Missing");
     break;
    case(ESSG_DT_ZERO):
     printf("%ld", DataCell.Value.lData);
     break;
    case(ESSG_DT_NOACCESS):
     printf("#NoAccess");
     break;
    case(ESSG_DT_MEMBER):
     printf("%s", DataCell.Value.pszStr+1);
     break;
    default:
     break;
   }
 }
 if(!sts)
   EssGDisconnect(hGrid, 0);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGCreateMemberwKeyStr

入力としてメンバー名およびメンバー・キーを使用して結合文字列を作成します。キーは Essbase が生成した値で、アウトライン内のメンバー名を一意に特定します。

構文

```
ESSG_FUNC_M
EssGCreateMemberwKeyStr
 (
pszMember, pszKey, *pszOutStr
);
```

パラメータ	データ型	説明
pszMember	ESSG_STR_T	メンバー名(入力)。
pszKey	ESSG_STR_T	メンバー・キー(入力)。

パラメータ データ型 説明

*pszOutStr ESSG_STR_T フォーマットの出力文字列:

<member-name length><member-name><key length><key>

ここで、長さ要素のサイズは 2 バイトです。<member-name>は NULL で終了することに注意してください。

備考

EssGFreeMemberwKeyStr を使用して文字列*pszOutStr を解放する必要があります。

例

```
    ESSG_VOID_T ESSG_BeginZoomIn(ESSG_HGRID_T hGrid)
{
ESSG_STS_T   sts = ESS_STS_NOERR;
ESSG_DATA_T  **ppDataIn;
ESSG_DATA_T  **ppDataOut;
ESSG_RANGE_T rDataRangeIn, rDataRangeOut;
ESSG_ULONG_T ulOptions;
ESSG_USHORT_T usCells;
ESSG_RANGE_T pZoomCells;
ESSG_USHORT_T usState;
ESSG_USHORT_T usMember2Len, usKey2Len;
ESSG_SHORT_T    sOption, sOptionGet;
  ESSG_SHORT_T    tmpShort, tmpShortGet, i;
  ESSG_PVOID_T    pOption, pOptionGet;
ESSG_STR_T  pMember, pKey, pOutStr;
ESSG_STR_T   pMember2, pKey2;


/* connect the grid to a database on the server */
sts = EssGConnect(hGrid, server, "essexer", pwd, app, db, ESSG_CONNECT_NODIALOG);

/* set grid option*/
tmpShort = ESSG_TRUE;
sOption = ESSG_OP_MEMBERANDUNIQUENAME ;
pOption = (ESSG_PVOID_T)tmpShort;   // pOption holds the actual value not a pointer

  sts = EssGSetGridOption(hGrid, sOption, pOption);
printf("EssGSetGridOption sts %ld\n",sts);

sOptionGet = ESSG_OP_MEMBERANDUNIQUENAME ;
  pOptionGet = &tmpShortGet;
  if(!sts)
{
   sts = EssGGetGridOption(hGrid, sOptionGet, pOptionGet);
  printf("EssGGetGridOption sts %ld\n",sts);
  printf("EssGSetGridOption set ESSG_OP_MEMBERANDUNIQUENAME TO %d\n",
(int)tmpShortGet);
  }
```

```
if(sts == 0)
{
 ppDataIn = BuildTable(&rDataRangeIn);

 ulOptions = ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL;

 pZoomCells.ulRowStart = 0;
 pZoomCells.ulColumnStart = 2;
 pZoomCells.ulNumRows = 1;
 pZoomCells.ulNumColumns = 1;
 usCells = 1;

 /* start the zoom in operation */
 sts = EssGBeginZoomIn(hGrid, usCells, &pZoomCells, ulOptions);
 printf("EssGBeginZoomIn  sts:  %ld\n",sts);
}

//Display Input
DisplayOutput(ppDataIn, rDataRangeIn);
printf("\n\n");
if(sts == 0)
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);

if(sts == 0)
{
 /* perform the zoom-in */
 sts = EssGPerformOperation(hGrid, 0);

 /* Free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
 /* determine the results of the zoom-in */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut, &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 DisplayOutput(ppDataOut, rDataRangeOut);


 /* Retreive member and key from cell */
 sts = EssGGetFromMemberwKey  (((ppDataOut[1][0]).Value).pszStr, &pMember, &pKey);
    printf("After EssGGetFromMemberwKey\n Member: %s, Key: %s \n\n",
   pMember+2,
  pKey+2);

 //Member is  "Qtr1", Key is "[2004].[Qtr1]", pOutStr is in the format
 //nn<member-name>nn<'key> - where nn is string length
```

```
usMember2Len = strlen("Qtr1");
pMember2 = malloc(usMember2Len+3);
   memset(pMember2, 0, usMember2Len+3);
usKey2Len = strlen("[2004].[Qtr1]");
  pKey2 = malloc(usKey2Len+3);
   memset(pKey2, 0, usKey2Len+3);

memcpy(pMember2, &usMember2Len, 2);
memcpy(pMember2+2, "Qtr1", usMember2Len);

memcpy(pKey2, &usKey2Len, 2);
memcpy(pKey2+2, "[2004].[Qtr1]", usKey2Len);

sts = EssGCreateMemberwKeyStr(pMember2, pKey2, &pOutStr);

/*Note: because not all elements in pOutStr are actual characters,
  e.g. the 2 bytes for the size of Member and size of Key, plus the
  \0 ending characters, the printf below does not display the actual
  contents of the array */
for (i=0;i < usMember2Len + usKey2Len + 4 + 2; ++i)
printf("%c", pOutStr[i]);


/* Free the returned data */
EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
sts = EssGFreeMemberwKeyStr (pOutStr);

}

if( sts == 0)
{
EssGEndOperation(hGrid, 0);
EssGDisconnect(hGrid, 0);
}
}
```

関連トピック

- EssGFreeMemberwKeyStr
- EssGGetFromMemberwKey

EssGConnect

グリッドを Essbase データベースに接続します。

構文

```
     ESSG_FUNC_M
     EssGConnect
      (
     hGrid, Server, Username, Password, Application,
Database, ulOptions
     );
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
Server	ESSG_SERVER_T	ネットワーク・サーバー名の文字列。 サーバー名は hostname または hostname:port で表すことができます。
Username	ESSG_USERNAME_T	サーバー上の有効なユーザー名。
Password	ESSG_PASSWORD_T	ユーザーのパスワード。
Application	ESSG_APPLICATION_T	サーバー上の有効なアプリケーション名。
Database	ESSG_DATABASE_T	サーバー上にあるアプリケーションにおいて有効なデータベース名。
ulOptions	ESSG_ULONG_T	オプションのフラグ。値は、ダイアログを表示せずデフォルト、または渡された設定を使用してログインと接続を試行する場合は ESSG_CONNECT_NODIALOG になります。ログインおよび選択のダイアログを表示する場合は ESSG_CONNECT_DEFAULT になります。

備考

- **EssAutoLogin** が呼び出されるため、**EssAutoLogin** に適用されるすべてのルールがこの関数にも適用されます。たとえば、すべてのパラメータは大文字と小文字を区別しません。
- ulOptions が **ESSG_CONNECT_NODIALOG** に設定されている場合、接続に関連するパラメータを NULL または空にできません。ulOptions が **ESSG_CONNECT_DEFAULT** に設定され、バッファが接続用パラメータに渡されると、ユーザーがダイアログで選択した内容がこれらのバッファに戻されます。
- すべてのセキュリティ情報が使用されています。そのため、読取りアクセス権限のないデータベースに接続すると、すべての読取り操作が失敗します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M     sts = ESS_STS_NOERR;
ESSG_INIT_T     InitStruct;
ESSG_HANDLE_T    Handle;
ESSG_SERVER_T    Server;
ESSG_USERNAME_T   UserName;
ESSG_PASSWORD_T   Password;
ESSG_APPLICATION_T Application;
ESSG_DATABASE_T   Database;
ESSG_ULONG_T    ulOptions;
```

```
ESSG_HGRID_T    hGrid;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, &Handle);

/* initializes a specific grid */
if(!sts)
 sts = EssGNewGrid(Handle, &hGrid);

strcpy(Server, "Rainbow");
strcpy(UserName, "Admin");
strcpy(Password, "Password");
strcpy(Application, "Demo");
strcpy(Database, "Basic");
ulOptions = ESSG_CONNECT_NODIALOG;

/* connects the grid to a database on the server */
if(!sts)
 sts = EssGConnect(hGrid, Server, UserName, Password, Application,
    Database, ulOptions);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGConnectEx

ユーザー名とパスワードではなくユーザー認証トークンを使用して、グリッドを Essbase データベースに接続します。

構文

```
ESSG_FUNC_M
EssGConnectEx
 (
hGrid, Server, Token, Application, Database, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
Server	ESSG_SERVER_T	ネットワーク・サーバー名の文字列。 サーバー名は hostname または hostname:port で表すことができます。

パラメータ	データ型	説明
Token	ESSG_TOKEN_T	認証されるユーザーのユーザー名とパスワードを表すトークン。
Username	ESSG_USERNAME_T	サーバー上の有効なユーザー名。
Password	ESSG_PASSWORD_T	ユーザーのパスワード。
Application	ESSG_APPLICATION_T	サーバー上の有効なアプリケーション名。
Database	ESSG_DATABASE_T	サーバー上にあるアプリケーションにおいて有効なデータベース名。
ulOptions	ESSG_ULONG_T	オプションのフラグ。値は、ダイアログを表示せずデフォルト、または渡された設定を使用してログインと接続を試行する場合は **ESSG_CONNECT_NODIALOG** になります。ログインおよび選択のダイアログを表示する場合は **ESSG_CONNECT_DEFAULT** になります。

備考

- この関数が失敗した場合は、ユーザーのユーザー名とパスワードを確認するために、対応する EssGConnect()関数が自動的に呼び出されます。
- **EssAutoLogin** が呼び出されるため、**EssAutoLogin** に適用されるすべてのルールがこの関数にも適用されます。たとえば、すべてのパラメータは大文字と小文字を区別しません。
- ulOptions が **ESSG_CONNECT_NODIALOG** に設定されている場合、接続に関連するパラメータを NULL または空にできません。ulOptions が **ESSG_CONNECT_DEFAULT** に設定され、バッファが接続用パラメータに渡されると、ユーザーがダイアログで選択した内容がこれらのバッファに戻されます。
- すべてのセキュリティ情報が使用されています。そのため、読取りアクセス権限のないデータベースに接続すると、すべての読取り操作が失敗します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

関連トピック

- EssGConnect
- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGDeleteLRO

指定された LRO を Essbase データベースから削除します。

構文

```
ESSG_FUNC_M
EssGDeleteLRO
```

```
(
hGrid, hLRO
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	**EssGNewGrid()**から戻されるグリッド・ハンドル。
hLRO;	ESSG_HLRO_T	リンク・オブジェクトへのハンドル(**EssGGetCellLinkResults()**関数によりDRILLDATA 構造体に戻されます)。

備考

特定の範囲のセルへリンクされたオブジェクトをすべて削除するには、EssGBeginDeleteLROs を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginCreateLRO
- EssGBeginDeleteLROs
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRODesc
- EssGGetLRO
- EssGUpdateLRO

EssGDestroyGrid

グリッド・インスタンスを破棄します。

構文

```
ESSG_FUNC_M
EssGDestroyGrid
(
hGrid
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。

備考

渡されたグリッド・ハンドルに関連付けられたメモリーを解放し、このグリッド・ハンドルを無効にします。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;
ESSG_HGRID_T   hGrid;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, &Handle);

/* initializes a specific grid */
if(!sts)
 sts = EssGNewGrid(Handle, &hGrid);

/* destroys a grid instance */
if(!sts)
 sts = EssGDestroyGrid(hGrid);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGNewGrid

EssGDisconnect

サーバー上のデータベースからグリッドを切断します。

構文

```
ESSG_FUNC_M
EssGDisconnect
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_ULONG_T  ulOptions = 0;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;
ESSG_HGRID_T  hGrid;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, &Handle);

/* initializes a specific grid */
if(!sts)
 sts = EssGNewGrid(&Handle, &hGrid);

/* connects the grid to a database on the server */
if(!sts)
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo", "Basic",
    ESSG_CONNECT_DEFAULT);

/* disconnects a grid from database at server */
if(!sts)
 sts = EssGDisconnect(hGrid, ulOptions);

/* terminate the EGAPI */
if(!sts)
 sts = EssGTerm(Handle);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGDTBeginDrillThrough

指定したデータ・セル範囲のドリルスルー・インスタンス・ハンドルを戻します。

構文

```
ESSG_FUNC_M
EssGDTBeginDrillThrough
(
hGrid, usCells, pCells, ppDTInst
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるオリジナル・グリッドのハンドル。
usCells;	ESSG_USHORT_T	pCells 配列のセル範囲数。
pCells;	1101 ページの「ESSG_RANGE_T」	ドリルスルー・レポート・データを受け取るために選択されたセル範囲の配列。
ppDTInst;	ESSG_PPDTHINST_T	指定したデータ・セル範囲について戻されるドリルスルー・インスタンス・ハンドル。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTEndDrillThrough
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTGetReportData
- EssGInit
- EssGDTListReports
- EssGDTReportCount

EssGDTConnect

指定したドリルスルー・ハンドルのドリルスルー接続情報を取得し、Oracle Essbase Studio に接続します。

構文

```
ESSG_FUNC_M
EssGDTConnect
(
pDTInst
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo

EssGDTEndDrillThrough

ドリルスルー・セッションを終了し、指定したドリルスルー・インスタンス・ハンドルのメモリーを解放します。

構文

```
ESSG_FUNC_M
EssGDTEndDrillThrough
(
pDTInst
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	指定したデータ・セル範囲における初期化済ドリルスルー・インスタンス・ハンドル。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTExecuteReport
- EssGDTGetData

- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo

EssGDTExecuteReport

レポート構造体の配列に対するインデックスによって指定されたレポートを実行します。

構文

```
 ESSG_FUNC_M
EssGDTExecuteReport
 (
pDTInst
,
Index
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル
Index;	ESSG_ULONG_T	実行するレポートのインデックス

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo

EssGDTGetData

指定したドリルスルー・インスタンス・ハンドルに対するレポート・データの配列を取得します。

構文

```
ESSG_FUNC_M
EssGDTGetData
(
pDTInst
,
ppData
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
ppData;	1097 ページの「ESSG_DTDATA_T」	指定したデータ・セルに対するレポート・データ構造体の配列。
pulCount;	ESSG_PULONG_T	ppData 配列のデータ・ブロック数。

備考

- pulCount が 0 になるまで **EssGDTGetData()**を呼び出します。
- **EssGDTGetData()**を呼び出した後で、**EssFree()**を使用して ppData(**ESSG_DTDATA_T**)に使用したメモリーを解放します。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo
- EssFree

EssGDTGetHeader

指定したドリルスルー・インスタンス・ハンドルのレポート・データ・ヘッダー情報を取得します。

構文

```
ESSG_FUNC_M
EssGDTGetHeader
(
pDTInst
,
ppHeader
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル。
ppHeader;	1097 ページの「ESSG_DTHEADER_T」	指定した列のヘッダー情報構造体の配列。
pulCount;	ESSG_PULONG_T	ppHeader ヘッダー情報配列のデータ・ブロック数。

備考

EssGDTGetHeader()を呼び出した後で、**EssFree()**を使用して ppHeader(**ESSG_DTHEADER_T**)に使用したメモリーを解放します。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo
- EssFree

EssGDTGetInfo

指定されたドリルスルー・ハンドルのドリルスルー接続情報を取得します。

構文

```
ESSG_FUNC_M
EssGDTGetInfo
(
```

```
pDTInst
,
pDTInfo
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル
pDTInfo;	1098 ページの「ESSG_DTINFO_T」	指定したデータ・セル範囲の接続情報構造体へのポインタ

備考

- **EssGDTGetInfo()**を呼び出す前に **ESSG_DTINFO_T** へメモリーを割り当てます。
- password は pDTInfo へ戻されません。つまり、**ESSG_DTINFO_T** の password フィールドの値は戻されません。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetHeader
- EssGDTListReports
- EssGDTRequestDrillThrough
- EssGDTSetInfo

EssGDTGetReportData

指定のデータ・セル範囲について事前に定義されたデフォルトのドリルスルー・レポート処理を実行して、指定のグリッド・ハンドル hDAGrid を介してレポート・データを戻します。

構文

```
ESSG_FUNC_M
EssGDTGetReportData
 (
hGrid, hDAGrid, usCells, pCells
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	**EssGNewGrid()**から戻されるオリジナル・グリッドのハンドル。
hDAGrid;	ESSG_HGRID_T	ドリルスルー・レポート・データを受け取る新規グリッドのハンドル。
usCells;	ESSG_USHORT_T	pCells 配列のセル範囲数。
pCells;	1101 ページの「ESSG_RANGE_T」	ドリルスルー・レポート・データを受け取るために選択されたセル範囲の配列。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTEndDrillThrough
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGInit
- EssGDTListReports
- EssGDTReportCount
- EssGDTRequestDrillThrough

EssGDTListReports

指定されたドリルスルー・インスタンス・ハンドルのレポート構造体の配列を戻します。

構文

```
ESSG_FUNC_M
EssGDTListReports
 (
pDTInst
,
ppReports
,
pulCount
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル
ppReports;	1098 ページの「ESSG_DTREPORT_T」	指定したドリルスルー・インスタンス・ハンドルのレポート構造体の配列
pulCount;	ESSG_PULONG_T	ppReports ヘッダー情報配列のブロック数

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTRequestDrillThrough
- EssGDTSetInfo

EssGDTReportCount

指定したデータ・セル範囲に対して定義されたレポートの数を戻します。

構文

```
ESSG_FUNC_M
EssGDTReportCount
 (
hGrid, usCells, pCells, uspReportNum
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるオリジナル・グリッドのハンドル。
usCells;	ESSG_USHORT_T	pCells 配列のセル範囲数。
pCells;	1101 ページの「ESSG_RANGE_T」	ドリルスルー・レポート・データを受け取るために選択されたセル範囲の配列。
uspReportNum;	ESSG_PUSHORT_T	指定したデータ・セル範囲に対して定義されたレポートの数。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTEndDrillThrough
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTGetReportData
- EssGInit
- EssGDTListReports

- EssGDTRequestDrillThrough

EssGDTRequestDrillThrough

指定されたデータ・セル範囲のドリルスルー・インスタンス・ハンドルを戻します。

構文

```
ESSG_FUNC_M
EssGDTRequestDrillThrough
(
hGrid
,
usCells
,
pCells
,
ppDTInst
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるオリジナル・グリッドのハンドル。
usCells;	ESSG_USHORT_T	pCells 配列のセル範囲数。
pCells;	1101 ページの「ESSG_RANGE_T」	ドリルスルー・レポート・データを受け取るために選択されたセル範囲の配列。
ppDTInst;	ESSG_PPDTHINST_T	指定したデータ・セル範囲について戻されるドリルスルー・インスタンス・ハンドル。

備考

- Essbase サーバーに最適化された拡張メンバー・コメントの要求を送信します
- 指定した拡張メンバー・コメントでドリルスルー・セッションを初期化します
- ドリルスルー・インスタンス・ハンドル ppDTInst を戻します。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData

- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTSetInfo
- EssOtlGetMemberCommentEx
- EssOtlSetMemberCommentEx

EssGDTSetInfo

指定したドリルスルー・ハンドルのドリルスルー接続情報を設定します。

構文

```
 ESSG_FUNC_M
EssGDTSetInfo
 (
pDTInst
,
pDTInfo
);
```

パラメータ	データ型	説明
pDTInst;	ESSG_PDTHINST_T	初期化済ドリルスルー・インスタンス・ハンドル
pDTInfo;	1098 ページの「ESSG_DTINFO_T」d	指定したデータ・セル範囲の接続情報構造体へのポインタ

備考

ESSG_DTINFO_T の inputOption フィールドは無視されます。

例

コード例については、1203 ページの「C グリッド API ドリルスルーの例」を参照してください。

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssGDTConnect
- EssGDTEndDrillThrough
- EssGDTExecuteReport
- EssGDTGetData
- EssGDTGetHeader
- EssGDTGetInfo
- EssGDTListReports
- EssGDTRequestDrillThrough

EssGEndOperation

操作が完了してすべての行が戻された後で、使用していた内部リソースを解放します。

構文

```
ESSG_FUNC_M
EssGEndOperation
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

備考

この呼出しはオプションで、操作が完了した後に内部リソースの解放に使用できます。この呼出しを行わない場合、内部リソースは、次の操作が開始した際、または呼出し元がグリッドを切断した際のうち、先に生じた条件の場合に解放されます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
EssGEndOperation(hGrid, 0);
```

このコードを使用した例については、EssGBeginRetrieve の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGFreeCellLinkResults

EssGGetCellLinkResults()を呼び出した結果作成されたリンクを保管するための予約済リソースをすべて解放します。

構文

```
ESSG_FUNC_M
EssGFreeCellLinkResults
(
hGrid, pDrillData)
;
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
pDrillData;	1096 ページの「ESSG_DRILLDATA_T」	リンク・オブジェクトの情報を含む ESSG_DRILLDATA_T 構造体の配列への参照。

備考

EssGGetCellLinkResults()では、ESSG_DRILLDATA_T へのポインタが参照されますが、この関数では ESSG_DRILLDATA_T へのポインタのみを必要とします。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginCreateLRO
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGBeginRemoveOnly
- EssGDeleteLRO
- EssGGetCellLinkResults
- EssGGetLRO
- EssGUpdateLRO

EssGFreeMemberInfo

この関数は EssGGetMemberInfo、EssGGetDataPointResults などのメンバー情報を戻す関数呼出しにより戻されたデータをすべて解放します。

構文

```
ESSG_FUNC_M
EssGFreeMemberInfo
(
hGrid, ulMembers, pszMembers
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
ulMembers	ESSG_ULONG_T	解放する ppszMembers 配列の要素数を指定します。
pszMembers	ESSG_PSTR_T	解放するメンバー名の 1 次元配列へのポインタ。

備考

この関数のパラメータには、1 次元配列の要素数とメンバー名自体の 1 次元配列が含まれます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
EssGFreeMemberInfo(hGrid, ulMembers, pszMembers);
```

このコードを使用した例については、EssGGetMemberInfo を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGFreeMemberwKeyStr

EssGCreateMemberwKeyStr によって作成されたメンバー名とメンバー・キーの結合文字列を解放します。

構文

```
ESSG_FUNC_M
EssGFreeMemberwKeyStr
 (
pszStr
);
```

パラメータ	データ型	説明
pszStr ；	ESSG_STR_T	入力。次のフォーマットのメンバー/キーの結合文字列: <メンバー名の長さ><メンバー名><キーの長さ><キー>

例

```
    ESSG_VOID_T ESSG_BeginZoomIn(ESSG_HGRID_T hGrid)
{
```

```
ESSG_STS_T   sts = ESS_STS_NOERR;
ESSG_DATA_T  **ppDataIn;
ESSG_DATA_T  **ppDataOut;
ESSG_RANGE_T rDataRangeIn, rDataRangeOut;
ESSG_ULONG_T ulOptions;
ESSG_USHORT_T usCells;
ESSG_RANGE_T pZoomCells;
ESSG_USHORT_T usState;
ESSG_USHORT_T usMember2Len, usKey2Len;
ESSG_SHORT_T     sOption, sOptionGet;
 ESSG_SHORT_T     tmpShort, tmpShortGet, i;
 ESSG_PVOID_T     pOption, pOptionGet;
ESSG_STR_T  pMember, pKey, pOutStr;
ESSG_STR_T   pMember2, pKey2;


/* connect the grid to a database on the server */
sts = EssGConnect(hGrid, server, "essexer", pwd, app, db, ESSG_CONNECT_NODIALOG);

/* set grid option*/
tmpShort = ESSG_TRUE;
sOption = ESSG_OP_MEMBERANDUNIQUENAME ;
pOption = (ESSG_PVOID_T)tmpShort;   // pOption holds the actual value not a pointer

 sts = EssGSetGridOption(hGrid, sOption, pOption);
printf("EssGSetGridOption sts %ld\n",sts);

sOptionGet = ESSG_OP_MEMBERANDUNIQUENAME ;
 pOptionGet = &tmpShortGet;
 if(!sts)
{
   sts = EssGGetGridOption(hGrid, sOptionGet, pOptionGet);
 printf("EssGGetGridOption sts %ld\n",sts);
 printf("EssGSetGridOption set ESSG_OP_MEMBERANDUNIQUENAME TO %d\n",
(int)tmpShortGet);
 }


if(sts == 0)
{
ppDataIn = BuildTable(&rDataRangeIn);

ulOptions = ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL;

pZoomCells.ulRowStart = 0;
pZoomCells.ulColumnStart = 2;
pZoomCells.ulNumRows = 1;
pZoomCells.ulNumColumns = 1;
usCells = 1;

/* start the zoom in operation */
sts = EssGBeginZoomIn(hGrid, usCells, &pZoomCells, ulOptions);
printf("EssGBeginZoomIn sts: %ld\n",sts);
}

//Display Input
DisplayOutput(ppDataIn, rDataRangeIn);
```

```
printf("\n\n");
if(sts == 0)
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);

if(sts == 0)
{
 /* perform the zoom-in */
 sts = EssGPerformOperation(hGrid, 0);

 /* Free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
 /* determine the results of the zoom-in */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut, &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 DisplayOutput(ppDataOut, rDataRangeOut);


 /* Retreive member and key from cell */
 sts = EssGGetFromMemberwKey (((ppDataOut[1][0]).Value).pszStr, &pMember, &pKey);
    printf("After EssGGetFromMemberwKey\n Member: %s, Key: %s \n\n",
   pMember+2,
  pKey+2);

 //Member is  "Qtr1", Key is "[2004].[Qtr1]", pOutStr is in the format
 //nn<member-name>nn<'key> - where nn is string length

 usMember2Len = strlen("Qtr1");
 pMember2 = malloc(usMember2Len+3);
    memset(pMember2, 0, usMember2Len+3);
 usKey2Len = strlen("[2004].[Qtr1]");
  pKey2 = malloc(usKey2Len+3);
    memset(pKey2, 0, usKey2Len+3);

 memcpy(pMember2, &usMember2Len, 2);
 memcpy(pMember2+2, "Qtr1", usMember2Len);

 memcpy(pKey2, &usKey2Len, 2);
 memcpy(pKey2+2, "[2004].[Qtr1]", usKey2Len);

 sts = EssGCreateMemberwKeyStr(pMember2, pKey2, &pOutStr);

 /*Note: because not all elements in pOutStr are actual characters,
  e.g. the 2 bytes for the size of Member and size of Key, plus the
  \0 ending characters, the printf below does not display the actual
  contents of the array */
```

```
 for (i=0;i < usMember2Len + usKey2Len + 4 + 2; ++i)
 printf("%c", pOutStr[i]);


 /* Free the returned data */
 EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
 sts = EssGFreeMemberwKeyStr (pOutStr);

 }

 if( sts == 0)
 {
 EssGEndOperation(hGrid, 0);
 EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- EssGCreateMemberwKeyStr
- EssGGetFromMemberwKey

EssGFreeRows

この関数は **EssGGetRows** から戻されたデータを解放します。

構文

```
ESSG_FUNC_M
EssGFreeRows
 (
hGrid, pRange, ppData
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
pRange	1101 ページの「ESSG_RANGE_T」	データ範囲を指定します。
ppData	1094 ページの「ESSG_DATA_T」	解放するデータの 2 次元配列。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
```

このコードを使用した例については、EssGBeginRetrieve の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetAPIContext

この関数は、指定したグリッドの API ログイン・コンテキスト・ハンドルを取得します。

構文

```
ESSG_FUNC_M
EssGGetAPIContext
 (
hGrid, pEssHctx
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
pEssHctx	ESSG_PPVOID_T	接続したグリッドの API コンテキスト・ハンドルが戻される変数。

備考

- そのため、ログイン・コンテキスト・ハンドルを必要とするグリッド以外の API 関数を呼出し元で呼び出せます。
- サーバーとの接続が無効な場合、有効な API のコンテキスト・ハンドルを取得できないため、呼出しが失敗し、*pEssHctx は ESS_INVALID_HCTX に設定されます。
- コンテキスト情報を変更する可能性のある API 関数では、戻されたログイン・コンテキストを使用しないでください。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;
ESSG_PVOID_T   EssHctx;
ESSG_HGRID_T   hGrid;

InitStruct.ulVersion = ESSG_VERSION;
```

```
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, Handle);

if(!sts)
 sts = EssGNewGrid(Handle, &hGrid);

/* connect the grid to a database on the server */
if(!sts)
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
    "Password", "Demo",
    "Basic", ESSG_CONNECT_DEFAULT);

/* Get API context handle for the specified grid */
if(!sts)
 sts = EssGGetAPIContext(hGrid, &EssHctx);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetAPIInstance

この関数は、API 初期化インスタンス・ハンドルを取得します。

構文

```
ESSG_FUNC_M
EssGGetAPIInstance
 (
Handle, pEssHinst
);
```

パラメータ	データ型	説明
Handle	ESSG_HANDLE_T	**EssGInit** から戻されるハンドル。
pEssHinst	ESSG_PPVOID_T	グリッド API で使用される API インスタンス・ハンドルが戻される変数。

備考

これにより、インスタンス・ハンドルを必要とするグリッドでない API 関数を呼出し元から呼び出せます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_PVOID_T  EssHinst;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T Handle;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, Handle);

/* get API initialization instance handle */
if(!sts)
 sts = EssGGetAPIInstance(Handle, &EssHinst);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetCellLinkResults

前に呼び出した **EssGBeginDrillOrLink()**により作成されたリンクのリストを取得します。

構文

```
     ESSG_FUNC_M
     EssGGetCellLinkResults
      (
     hGrid, pfCanDrill, pNumLROs,
ppDrillData, pRangeOut, pState
     );
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	**EssGNewGrid()**から戻されるグリッド・ハンドル。
pfCanDrill;	ESSG_PBOOL_T	セルにリンク済オブジェクトがある場合は TRUE が戻されます。**EssGBeginDrillOrLink()**に ESSG_OPT_ZOOM オプションを指定して、結果のズームインを要求した場合は、pfCanDrill から FALSE が戻されます。

パラメータ	データ型	説明
pNumLROs;	ESSG_PULONG_T	取得したリンクの数が戻されます。
ppDrillData;	1096 ページの「ESSG_DRILLDATA_T」	リンク・オブジェクトに関する情報が含まれる ESSG_DRILLDATA_T 構造体の配列への参照を戻します。
pRangeOut;	1101 ページの「ESSG_RANGE_T」	LRO が検出されず、**EssGBeginDrillOrLink()**呼出しに ESSG_OPT_ZOOM オプションを指定した場合には、ズームインするセル範囲が戻されます。
pState;	ESSG_PUSHORT_T	操作の状態を表す次のいずれかの値が戻されます: ● In progress - すべてのセルの取得が終了していません ● Done - すべてのセルの取得が終了しました

備考

- この関数を使用すると、ESSG_DRILLDATA_T にメモリーが割り当てられます。割り当てられているメモリーを解放する場合は、EssGFreeCellLinkResults を呼び出してください。
- この関数により取得されたハンドルは、次のグリッド API 関数のいずれかを呼び出すまで有効です:
  - **EssGBeginDrillOrLink()**
  - **EssGBeginCreateLRO()**
  - **EssGUpdateLRO()**
  - **EssGBeginDeleteLRO()**
  - **EssGDeleteLRO()**

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginRemoveOnly
- EssGGetGridOption
- EssGPerformOperation
- EssGSetGridOption
- EssGBeginCreateLRO
- EssGBeginDrillAcross
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetLRODesc
- EssGGetLinkedPartitionDesc
- EssGGetLRO
- EssGUpdateLRO

EssGGetDataPointResults

EssGBeginDataPoint 呼出し(**EssGBeginDataPoint**、**EssGSendRows**、**EssGPerformOperation**)から情報を取得します。

構文

```
      ESSG_FUNC_M
      EssGGetDataPointResults
       (
      hGrid, pulMembers, ppszMembers,
pState
      );
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
pulMembers	ESSG_PULONG_T	戻されるメンバーの数。
*ppszMembers	ESSG_PSTR_T	サーバーから戻された pulMembers のサイズのメンバーの 1 次元配列を指すポインタ。API はこのメモリーを割り当て、**EssGFreeMemberInfo** を使用して呼出し元から解放する必要があります。 **注：** ppszMembers パラメータは、**EssGFreeMemberInfo** を使用して呼出し元が解放する必要があります。
pState	ESSG_PUSHORT_T	操作状態を戻す変数。これは、次のいずれかの値にできます: • **ESSG_STATE_DONE** 操作終了 • **ESSG_STATE_INPROGRESS** 操作中

備考

pState 変数に **ESSG_STATE_DONE** が戻されるまで、この呼出しを何回か繰り返します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
      sts = EssGGetDataPointResults(hGrid, &ulMembers,
```

このコードを使用した例については、EssGBeginDataPoint の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetFormattedValue

指定したセルのフォーマットされた値を戻します。

構文

```
ESS_FUNC_M EssGGetFormattedValue(
hGrid
,
pData
,
*fmtVal
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	グリッド・ハンドル
pData	ESSG_PDATA_T	セルの ESSG_DATA_T 構造体へのポインタ。
*fmtVal	ESSG_STR_T	このセルのフォーマットされた値へのポインタ

備考

- フォーマットされた値を取得するには、グリッド・オプション ESSG_OP_GET_FORMATTED_VALUE を有効にする必要があります。
- API で管理されるため、戻されたポインタを解放する必要はありません。

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

EssGGetFromMemberwKey

メンバー名とキーを戻します。キーは Essbase が生成した値で、アウトライン内のメンバー名を一意に特定します。

構文

```
ESSG_FUNC_M
EssGGetFromMemberwKey
 (
pszOutStr, pszMember, pszKey
);
```

パラメータ　データ型　説明

pszOutStr ; ESSG_STR_T 次のフォーマットの入力文字列: <メンバー名の長さ><メンバー名><キーの長さ><キー>。ここで、長さ要素のサイズは 2 バイトです。<メンバー名>は NULL で終了します。文字列は API から戻されるか、または EssGCreateMemberwKeyStr を使用して作成できます。

pszMember; ESSG_STR_T メンバー名(出力)。

pszKey; ESSG_STR_T メンバー・キー(出力)。

備考

ESSG_DATA_T 構造体の usType フィールドが ESSG_DT_MEMBERwKEY に設定されているとき、Value(ESSG_DATA_VALUE)フィールドの pszStr フィールドは pszOutStr に必要なフォーマットとして解釈されます。

例

```
    ESSG_VOID_T ESSG_BeginZoomIn(ESSG_HGRID_T hGrid)
{
ESSG_STS_T   sts = ESS_STS_NOERR;
ESSG_DATA_T  **ppDataIn;
ESSG_DATA_T  **ppDataOut;
ESSG_RANGE_T rDataRangeIn, rDataRangeOut;
ESSG_ULONG_T ulOptions;
ESSG_USHORT_T usCells;
ESSG_RANGE_T pZoomCells;
ESSG_USHORT_T usState;
ESSG_USHORT_T usMember2Len, usKey2Len;
ESSG_SHORT_T     sOption, sOptionGet;
 ESSG_SHORT_T     tmpShort, tmpShortGet, i;
 ESSG_PVOID_T     pOption, pOptionGet;
ESSG_STR_T  pMember, pKey, pOutStr;
ESSG_STR_T   pMember2, pKey2;


/* connect the grid to a database on the server */
sts = EssGConnect(hGrid, server, "essexer", pwd, app, db, ESSG_CONNECT_NODIALOG);

/* set grid option*/
tmpShort = ESSG_TRUE;
sOption = ESSG_OP_MEMBERANDUNIQUENAME ;
pOption = (ESSG_PVOID_T)tmpShort;   // pOption holds the actual value not a pointer

 sts = EssGSetGridOption(hGrid, sOption, pOption);
printf("EssGSetGridOption  sts  %ld\n",sts);

sOptionGet = ESSG_OP_MEMBERANDUNIQUENAME ;
 pOptionGet = &tmpShortGet;
 if(!sts)
{
   sts = EssGGetGridOption(hGrid, sOptionGet, pOptionGet);
 printf("EssGGetGridOption  sts  %ld\n",sts);
 printf("EssGSetGridOption set ESSG_OP_MEMBERANDUNIQUENAME TO %d\n",
(int)tmpShortGet);
```

```
 }


if(sts == 0)
{
 ppDataIn = BuildTable(&rDataRangeIn);

 ulOptions = ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL;

 pZoomCells.ulRowStart = 0;
 pZoomCells.ulColumnStart = 2;
 pZoomCells.ulNumRows = 1;
 pZoomCells.ulNumColumns = 1;
 usCells = 1;

 /* start the zoom in operation */
 sts = EssGBeginZoomIn(hGrid, usCells, &pZoomCells, ulOptions);
 printf("EssGBeginZoomIn  sts:  %ld\n",sts);
}

//Display Input
DisplayOutput(ppDataIn, rDataRangeIn);
printf("\n\n");
if(sts == 0)
 /* send the entire grid to define the query */
 sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);

if(sts == 0)
{
 /* perform the zoom-in */
 sts = EssGPerformOperation(hGrid, 0);

 /* Free the built data */
 FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
 /* determine the results of the zoom-in */
 sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
}
if(sts ==0)
{
 /* get all the data */
 sts = EssGGetRows(hGrid, 0, &rDataRangeOut, &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
 DisplayOutput(ppDataOut, rDataRangeOut);


 /* Retreive member and key from cell */
 sts = EssGGetFromMemberwKey  (((ppDataOut[1][0]).Value).pszStr, &pMember, &pKey);
    printf("After EssGGetFromMemberwKey\n Member: %s, Key: %s \n\n",
   pMember+2,
  pKey+2);
```

```
//Member is  "Qtr1", Key is "[2004].[Qtr1]", pOutStr is in the format
//nn<member-name>nn<'key> - where nn is string length

usMember2Len = strlen("Qtr1");
pMember2 = malloc(usMember2Len+3);
   memset(pMember2, 0, usMember2Len+3);
usKey2Len = strlen("[2004].[Qtr1]");
 pKey2 = malloc(usKey2Len+3);
   memset(pKey2, 0, usKey2Len+3);

memcpy(pMember2, &usMember2Len, 2);
memcpy(pMember2+2, "Qtr1", usMember2Len);

memcpy(pKey2, &usKey2Len, 2);
memcpy(pKey2+2, "[2004].[Qtr1]", usKey2Len);

sts = EssGCreateMemberwKeyStr(pMember2, pKey2, &pOutStr);

/*Note: because not all elements in pOutStr are actual characters,
 e.g. the 2 bytes for the size of Member and size of Key, plus the
 \0 ending characters, the printf below does not display the actual
 contents of the array */
for (i=0;i < usMember2Len + usKey2Len + 4 + 2; ++i)
printf("%c", pOutStr[i]);


/* Free the returned data */
EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
sts = EssGFreeMemberwKeyStr (pOutStr);

}

if( sts == 0)
{
EssGEndOperation(hGrid, 0);
EssGDisconnect(hGrid, 0);
}
}
```

関連トピック

- EssGCreateMemberwKeyStr
- EssGFreeMemberwKeyStr

EssGGetGridOption

この関数は個々のグリッド・オプションを取得します。

構文

```
ESSG_FUNC_M
EssGGetGridOption
 (
hGrid, sOption, pOption
```

```
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるハンドル。
sOption	ESSG_SHORT_T	取得するオプションを表す番号。
pOption	ESSG_PVOID_T	取得したオプションを指すポインタ。ESSG_OP_USERGRIDDATA ポインタを除いて、このデータは読取り専用で呼出し元からは解放できません。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

例

```
    ESSG_VOID_T EssG_GetGridOption(ESSG_HGRID_T hGrid)
{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_SHORT_T     sOption;
 ESSG_SHORT_T     tmpShort;
 ESSG_PVOID_T     pOption;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin", "Password", "Demo", "Basic",
        ESSG_CONNECT_DEFAULT);
 /* get grid option */
 sOption = ESSG_OP_DRILLLEVEL;
 pOption = &tmpShort;
 if(!sts)
  sts = EssGGetGridOption(hGrid, sOption, pOption);
 if(!sts)
 {
  printf("\n%s: %d", "DRILLLEVEL", tmpShort);
  EssGDisconnect(hGrid, 0);
 }
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGSetGridOption

EssGGetGridPerspective

グリッドのパースペクティブを戻します。

構文

```
ESSG_FUNC_M EssGGetGridPerspective(
hGrid
```

```
,
sAttrdim
,
*pPerspectiveType
,
*pPerspectiveString
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるハンドル。
sAttrdim	ESSG_STR_T	パースペクティブに対してクエリーが行われる属性次元名
*pPerspectiveType	ESSG_SHORT_T	パースペクティブのタイプ。1091 ページの「グリッドのパースペクティブ・タイプ」を参照してください。
*pPerspectiveString	ESSG_STR_T	戻されるパースペクティブのタプル・セットへのポインタ。 ● ESSG_PERSP_EXPLICIT 以外のパースペクティブ・タイプの場合は NULL です。 ● ESSG_PERSP_EXPLICIT に対しては、この値は明示的に解放する必要があります。

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

関連トピック

- EssGSetGridPerspective

EssGGetIsCellDrillable

セルがドリルスルー URL に関連付けられているかどうかをチェックします。

構文

```
ESS_FUNC_M EssGGetIsCellDrillable (
hGrid, pData, pIsDrillable
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル
pData	ESS_PDATA_T	セルの ESSG_DATA_T 構造体へのポインタ
pIsDrillable	ESS_PBOOL_T	セルがドリルスルー URL に関連付けられている場合は TRUE、それ以外の場合は FALSE

戻り値

- 正常に処理されると、pIsDrillable が適宜設定されます。

- 処理に失敗すると、エラー・コードが戻されます。

例

```
    #define  ESSG_OP_GET_DRILLTHRU_URLS      41

ESSG_STS_T  sts = EssGInit(&InitStruct, &Handle);
sts = EssGNewGrid(Handle, &hGrid);
sts = EssGConnect(hGrid,Server,UserName,Password,Application,Database,ulOptions);
sts = EssGSetGridOption(hGrid, ESSG_OP_GET_DRILLTHRU_URLS ,(ESSG_PVOID_T)
(ESSG_TRUE));

ppDataIn = BuildQuery(&rRangeDataIn);

sts = EssGBeginRetrieve(hGrid,ESSG_RET_RETRIEVE);
sts = EssGSendRows(hGrid, &rRangeDataIn, ppDataIn);
sts = EssGPerformOperation(hGrid, 0);

/*To retrieve the cell drillable property of a cell*/

EssGGetIsCellDrillable(hGrid, &(cells[ulRow][ulCol]), &bIsDrillable);
 if (bIsDrillable)
  printf("bIsDrillable: true");
 else
  printf("bIsDrillable: false");
```

EssGGetLinkedPartitionDesc

リンクしたパーティションの説明を取得します。**EssGGetCellLinkResults()**関数呼出しから戻された一意のハンドルを持つパーティションを指定します。

構文

```
ESSG_FUNC_M
EssGGetLinkedPartitionDesc
 (
hGrid, hLRO, pConnectInfo
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	**EssGNewGrid()**から戻されるグリッド・ハンドル。
hLRO;	ESSG_HLRO_T	リンク・パーティションへのハンドル(**EssGGetCellLinkResults()**関数により DRILLDATA 構造体に戻されます)。ハンドルはタイプ ESSG_PARTITIONTYPE のリンク・オブジェクトを指定する必要があります。
pConnectInfo;	1094 ページの「ESSG_CONNECTINFO_T」	リンク・パーティションの接続情報が戻されます。

備考

パーティション以外のリンク・オブジェクトの記述を取得するには、EssGGetLRODesc を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginRemoveOnly
- EssGGetCellLinkResults
- EssGGetGridOption
- EssGPerformOperation
- EssGSetGridOption
- EssGBeginCreateLRO
- EssGBeginDrillAcross
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRODesc
- EssGGetLRO
- EssGUpdateLRO

EssGGetLRO

LRO を Essbase データベースから取得します。

構文

```
ESSG_FUNC_M
EssGGetLRO
 (
hGrid, hLRO, szTargetFile, ulOption
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
hLRO	ESSG_HLRO_T	リンク・オブジェクトへのハンドル(EssGGetCellLinkResults()関数により DRILLDATA 構造体に戻されます)。
szTargetFile	ESSG_STR_T	オブジェクトを格納するターゲット・ファイルの名前。パスも含まれます。

パラメータ	データ型	説明
ulOption	ESSG_ULONG_T	オブジェクトとカタログ・エントリの一方またはその両方を取得するかどうかを指定するオプション。次のいずれかを使用します: ● ESS_LRO_OBJ_API オブジェクトのみを取得します。 ● ESS_LRO_CATALOG_API カタログ・エントリのみを取得します。 ● ESS_LRO_BOTH_API オブジェクトとカタログ・エントリの両方を取得します。

備考

セル・ノートを取得するには、EssGGetLRODesc を使用します。**EssGGetLRO** はセル・ノート情報を取得しません。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginRemoveOnly
- EssGGetCellLinkResults
- EssGGetGridOption
- EssGPerformOperation
- EssGSetGridOption
- EssGBeginCreateLRO
- EssGBeginDrillAcross
- EssGBeginDeleteLROs
- EssGBeginDrillOrLink
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRODesc
- EssGGetLinkedPartitionDesc
- EssGUpdateLRO

EssGGetLRODesc

リンク・オブジェクトの説明情報を取得します。**EssGGetCellLinkResults**()関数呼出しから戻された一意のハンドルを持つオブジェクトを指定します。

構文

```
ESSG_FUNC_M
EssGGetLRODesc
 (
hGrid, hLRO, pLroDesc
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	**EssGNewGrid()**から戻されるグリッド・ハンドル。
hLRO;	ESSG_HLRO_T	リンク・パーティションへのハンドル(**EssGGetCellLinkResults()**関数により DRILLDATA 構造体に戻されます)。ハンドルは ESSG_PARTITIONTYPE 以外のタイプのリンク・オブジェクトを指定できます。
pLroDesc;	ESSG_LPLRODESC_T	指定したオブジェクトの情報を含む LRO 記述構造体が戻されます。

備考

パーティション情報(オブジェクト・タイプ ESSG_PARTITIONTYPE)を取得するには、EssGGetLinkedPartitionDesc を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGGetCellLinkResults
- EssGBeginCreateLRO
- EssGBeginDeleteLROs
- EssGDeleteLRO
- EssGFreeCellLinkResults
- EssGGetCellLinkResults
- EssGGetLRO
- EssGUpdateLRO
- EssGUpdateLRO
- EssGUpdateLRO
- EssGUpdateLRO
- EssGUpdateLRO

EssGGetMemberInfo

1 つの次元からメンバーの関係情報が戻されます。

構文

```
      ESSG_FUNC_M
      EssGGetMemberInfo
       (
      hGrid, pszMbrName, sAction,
bAliases, pulMembers, ppszMembers
      );
```

パラメータ	データ型	説明
hGrid ;	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。

パラメータ	データ型	説明
pszMbrName;	ESSG_STR_T	関係情報を取得するメンバーの名前。
sAction;	ESSG_SHORT_T	どのようなタイプの関係情報を戻すのかを示す番号。このパラメータに有効な値(相互に排他的): ● ESSG_NEXTLEVEL 子 ● ESSG_ALLLEVELS すべてのメンバー ● ESSG_BOTTOMLEVEL 最下位レベル ● ESSG_SIBLEVEL 兄弟レベル ● ESSG_SAMELEVEL 同一レベル ● ESSG_SAMEGENERATION 同世代 ● ESSG_CALCLEVEL 計算 ● ESSG_PARENTLEVEL メンバーの親 ● ESSG_TOPLEVEL メンバーが属する次元
bAliases ;	ESSG_BOOL_T	別名が戻されるかどうかを指定します。
pulMembers;	ESSG_PULONG_T	戻されるメンバーの数。
*ppszMembers;	ESSG_PSTR_T	サーバーから戻される pulMembers のサイズのメンバーの 1 次元配列を指すポインタ。API がこのメモリーを割り当て、呼出し元が解放する必要があります。

備考

- pszMbrName には、NULL を指定できません。
- **EssGFreeMemberInfo** を使用して ppszMembers パラメータを解放します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_GetMemberInfo(ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M  sts = ESS_STS_NOERR;
 ESSG_STR_T     pszMbrName;
 ESSG_SHORT_T    sAction;
 ESSG_BOOL_T    bAliases;
 ESSG_ULONG_T    ulMembers, ind;
 ESSG_PSTR_T    pszMembers;
 char tmp[5] = "Year";

 pszMbrName = tmp;
 sAction = ESSG_NEXTLEVEL;
 bAliases = ESSG_FALSE;

 /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin",
```

```
    "Password", "Demo", "Basic",
    ESSG_CONNECT_NODIALOG);

 /* get member information */
 if(sts == 0)
   sts = EssGGetMemberInfo(hGrid,pszMbrName, sAction, bAliases,
      &ulMembers, &pszMembers);

 if (sts == 0)
 {
   printf("\nNext Level of %s:\n", pszMbrName);
for (ind = 0; ind < ulMembers; ind++)
   printf("\t%s\n", *(pszMembers + ind));

   EssGFreeMemberInfo(hGrid, ulMembers, pszMembers);
 }
 if(!sts)
   sts = EssGDisconnect(hGrid, 0);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetResults

操作の完了後に戻されるデータの情報を取得します(EssGBeginXxx、EssGSendRows、 EssGPerformOperation)。

構文

```
ESSG_FUNC_M
EssGGetResults
 (
hGrid, ulOptions, pRangeOut, pState
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。
pRangeOut	1101 ページの「ESSG_RANGE_T」	サーバーから戻されるデータの範囲を指定します。このパラメータは戻されるデータの総量を示します。呼出し元は EssGGetRows を複数回呼び出して、何度かに分けて取得できます。
pState	ESSG_PUSHORT_T	操作状態を戻す変数。これは、次のいずれかの値にできます: • ESSG_STATE_DONE 操作終了 • ESSG_STATE_INPROGRESS 操作中

備考

呼出し後、pState 変数に **ESSG_STATE_DONE** が含まれた場合、呼出し元は **EssGGetRows** を呼び出してサーバーから実際のデータを取得する必要があります。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
```

このコードを使用した例については、EssGBeginRetrieve の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetRows

操作の完了後にデータを取得します(**EssGBegin**、**EssGSendRows**、**EssGPerformOperation**、**EssGGetResults**)。

構文

```
ESSG_FUNC_M
EssGGetRows
(
hGrid, ulOptions, pRangeRequested,
pRangeOut, pppDataOut
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。
pRangeRequested	1101 ページの「ESSG_RANGE_T」	要求するデータ範囲を指定します。EssGGetResults の呼出しで戻される行数や列数以下になります。
pRangeOut	1101 ページの「ESSG_RANGE_T」	戻されるデータ範囲を指定します。

パラメータ	データ型	説明
*pppDataOut	1094 ページの「ESSG_DATA_T」	データの 2 次元配列アドレス。この配列のメモリーは API によって割り当てられ、呼出し元が **EssGFreeRows** を使用して解放する必要があります。

備考

- **EssGGetRows** を複数回呼び出すことは可能ですが、各呼出しの pRangeRequested->ulStartRow の値は、直前に受け取った行より大きくする必要があります。
- pRangeRequested 変数では、戻す必要がある行数を定義します。複数のデータ・バッファを戻す場合は、以降に **EssGGetRows** を呼び出すたびに、pRangeRequested パラメータ内の行が更新されます。
- 呼出し元が要求した行のうち、一部の行は有効で他の行は範囲外である場合は、有効な行に値が入力されます。無効な行は未定義のままになります。
- 呼出し元で使用可能な情報の範囲外にある行を要求すると、エラーが戻されます。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    sts = EssGGetRows(hGrid, 0, &rDataRangeOut,
&rDataRangeOut, &ppDataOut);
```

このコードを使用した例については、EssGBeginRetrieve の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGGetSmartlistforCell

セル・タイプが ESSG_DT_SMARTLIST の場合に、セルに関連付けられたスマートリスト(テキスト・リスト)オブジェクトの名前を戻します。

- Essbase データベースには、複数の TextList オブジェクトとメンバーを保持できます。
- この API 呼出しでは、セルが関連付けられている TextList オブジェクトを特定できます。
- API で管理されるため、戻されたポインタを解放する必要はありません。

- グリッドはステートレスなので、戻された名前は EssGEndOperation を実行するまで有効です。

構文

```
ESS_FUNC_M EssGGetSmartlistforCell (
hGrid
,
pData
,
*pSmartlistname
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	グリッド・ハンドル
pData	ESSG_PDATA_T	セルの ESSG_DATA_T 構造体へのポインタ。
*pSmartlistname	ESSG_STR_T	セルが関連付けられている TextList オブジェクトの名前へのポインタ

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

関連トピック

- EssOtlFreeSmartListInfo
- EssOtlGetMemberSmartList
- EssOtlGetSmartListInfo
- EssOtlPutSmartList

EssGInit

グリッド API を初期化します。

構文

```
ESSG_FUNC_M
EssGInit
 (
pInitStruct, pHandle
);
```

パラメータ	データ型	説明
pInitStruct	1099 ページの「ESSG_INIT_T」	EGAPI の有効な情報を含む構造体へのポインタ。
pHandle	ESSG_PHANDLE_T	EGAPI から戻されたハンドルへのポインタ。

備考

- **EssGVersion** 以外の他の EGAPI 関数を呼び出す前に、この関数を呼び出す必要があります。
- この関数は、セッションの開始時に 1 回のみ呼び出します。
- この関数は、使用する各グリッドの **EssGNewGrid** へ渡すハンドル、またはグリッドに固有でないハンドルを必要とする他の EGAPI 呼出しへ渡すハンドルを戻します。
- 他のスレッドのネットワーク・ステータス情報が上書きされないようにするには、スレッドに独自のハンドル(pHandle)が必要です。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

sts = EssGInit(&InitStruct, &Handle);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGLoginSetPass

グリッドを Essbase データベースに接続し、ユーザーのパスワードを変更します。

構文

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるハンドル
Server;	ESSG_SERVER_T	有効なサーバーの名前

パラメータ	データ型	説明
Username;	ESSG_USERNAME_T	サーバーで有効なユーザー名
Password;	ESSG_PASSWORD_T	ユーザーのパスワード
NewPassword;	ESSG_PASSWORD_T	ユーザーの新パスワード

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

例

```
    #include
#include

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_INIT_T   InitStruct;
 ESSG_HANDLE_T  Handle;
 ESSG_SERVER_T  Server;
 ESSG_USERNAME_T UserName;
 ESSG_PASSWORD_T Password;
 ESSG_PASSWORD_T NewPassword;
 ESSG_HGRID_T  hGrid;

 InitStruct.ulVersion    = ESSG_VERSION;
 InitStruct.ulMaxRows    = 1000;
 InitStruct.ulMaxColumns  = 200;
 InitStruct.pfnMessageFunc = ESS_NULL;
 InitStruct.pUserdata    = ESS_NULL;

 /* initializes EGAPI */
 sts = EssGInit(&InitStruct, Handle);

 /* initializes a specific grid */
 if(!sts)
  sts = EssGNewGrid(Handle, &hGrid);

  strcpy(Server, "Rainbow");
  strcpy(UserName, "Admin");
  strcpy(Password, "Password");
  strcpy(Password, "NewPassword");

 /* connects the grid to a database on the server */
 if(!sts)
  sts = EssGLoginSetPass(hGrid, Server, UserName, Password, NewPassword);
}
```

関連トピック

- 1085 ページの「C グリッド API の定数」
- 1093 ページの「C グリッド API の構造体」
- EssAutoLogin
- EssGConnect

- EssGInit
- EssInit
- EssLogout

EssGNewGrid

この関数は、指定されたグリッドを初期化します。

構文

```
ESSG_FUNC_M
EssGNewGrid
(
Handle, phGrid
);
```

パラメータ	データ型	説明
Handle; /* IN */	ESSG_HANDLE_T	EssGInit から戻されるハンドル。
phGrid; /* OUT */	ESSG_PHGRID_T	EGAPI から戻されたグリッド固有のハンドルへのポインタ。

備考

- この関数の呼出しは、グリッド固有の API を呼び出す前に実行します。
- 戻されたハンドルは、指定したグリッドを操作する、後続のグリッド固有の API 呼出しへ渡す必要があります。
- またこの関数は、グリッド API を使用するグリッドごとに 1 回ずつ呼び出すことが必要です。
- マルチスレッド環境の各スレッドでは、各ハンドル(phGrid)を使用して EssGSendRows()や EssGBeginOperation()などのグリッド固有の API を呼び出す必要があります。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;
ESSG_HGRID_T   hGrid;

InitStruct.ulVersion = ESSG_VERSION;
```

```
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initializes EGAPI */
sts = EssGInit(&InitStruct, &Handle);

/* initializes a specific grid */
if(!sts)
 sts = EssGNewGrid(Handle, &hGrid);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGPerformOperation

EssGBeginXxx と **EssGSendRows** を使用してすべての行がサーバーへ送信された後に、操作を実行します。

構文

```
ESSG_FUNC_M
EssGPerformOperation
 (
hGrid, ulOptions
);
```

パラメータ　データ型　　説明

hGrid;　　ESSG_HGRID_T **EssGNewGrid()**から戻されるハンドル。

ulOptions; ESSG_ULONG_T 今後使用するために予約されています。ゼロに設定します。

備考

戻されたデータの情報を取得するには、この関数を呼び出した後に、**EssGGetResults** を呼び出します。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

例

```
sts = EssGPerformOperation(hGrid, 0);
```

このコードを使用した例については、EssGBeginPivot の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGSendRows

この関数は操作が開始されると、サーバーへ行を送信します。

構文

```
ESSG_FUNC_M
EssGSendRows
 (
hGrid, pRangeIn, ppDataIn
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid から戻されるハンドル。
pRangeIn	1101 ページの「ESSG_RANGE_T」	ppDataIn のデータ範囲を説明します。
ppDataIn	1094 ページの「ESSG_DATA_T」	データを記述するセルの 2 次元配列。

備考

- EssGSendRows を複数回呼び出すことは可能ですが、後続の各呼出しの pRangeIn->ulStartRow の値は直前に送信された行より大きくする必要があります。
- pRangeIn 変数はグリッド内の行を定義します。
- 複数のデータ・バッファを送信する場合、EssGSendRows を呼び出すたびに pRangeIn パラメータの行を更新する必要があります。
- すべての行が送信されると、EssGPerformOperation を呼び出せます。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);
```

このコードを使用した例については、EssGBeginRetrieve の「例」の項を参照してください。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」

- 1093 ページの「C グリッド API の構造体」

EssGSetGridOption

この関数は、個別のグリッド・オプションを設定します。

構文

```
ESSG_FUNC_M
EssGSetGridOption
(
hGrid, sOption, pOption
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるハンドル。 Description
sOption	ESSG_SHORT_T	設定するオプションを表す値。有効な値の表については、「注意」を参照してください。
pOption	ESSG_PVOID_T	ESSG_PVOID_T へキャストを設定するオプションの値。

備考

- このアプリケーションのみで使用するグリッド固有の情報を保管するには、ESSG_OP_USERGRIDDATA ポインタを使用してください。
- 次の表に、sOption の有効なオプションと対応するデータ型の説明を示します:

値	説明	予期されるデータ型	デフォルト
ESSG_OP_ALIASNAMES	別名	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_ALIASTABLE	別名テーブル	ESSG_STR_T	
ESSG_OP_DATALESS	データベース・ナビゲーションを使用可能にします。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_DRILLLEVEL	ドリルレベル	ESSG_SHORT_T	ESSG_NEXTLEVEL
ESSG_OP_EMPTYGRIDERROR	FALSE の場合、データが見つからずグリッド・ヘッダーのみが戻されたクエリーについてエラーを発行しません。	ESSG_BOOL_T	ESSG_TRUE
ESSG_OP_INCSEL	選択を含めます	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_INDENT	インデント・スタイル	ESSG_SHORT_T	ESSG_INDENTTOTALS
ESSG_OP_LATEST	最新メンバーを指定する機能を有効にします。	ESSG_BOOL_T	ESSG_FALSE

値	説明	予期されるデータ型	デフォルト
ESSG_OP_LATESTMEMBER	最新メンバーを指定します。	ESSG_STR_T	NULL
ESSG_OP_REPEATMBRNAMES	メンバー名を繰り返します。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_RETAINTHREAD	TRUE に設定されている場合、グリッド操作の終わりにサーバー・スレッドの接続を解除しないでください。順にいくつかの操作を送信すると性能を向上させる場合があります。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SELGROUP	選択されていないグループの削除。選択したメンバーのすべてのオカレンスにズームしますが、同じ次元にある他のメンバーは、選択したメンバーそのものも含めてすべて削除されます。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SELONLY	選択したグループ内。選択したメンバーのインスタンスそのものにのみズームします。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SUPMISSING	欠落した行を抑制します。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SUPUNDER	アンダースコアをスペースに置換します。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SUPZEROS	0 の行を抑制します。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_UPDATEMODE	更新モード	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_USEBOTHFORROWDIMS	行次元にメンバー名と別名の両方を使用します。	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_USERGRIDDATA	ユーザー・データへのポインタ	ESSG_PVOID_T	NULL
ESSG_OP_RETAINTHREAD	スレッドを保持	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_EMPTYGRIDERROR	空のグリッド・エラーを発行	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_DATALESS	データなしで移動	ESSG_BOOL_T	ESSG_FALSE
ESSG_OP_SPANHYBRIDANALYSIS	リレーショナル・ソースまでドリルをスパン	ESSG_BOOL_T	ESSG_FALSE

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

例

```
    ESSG_VOID_T ESSG_SetGridOption (ESSG_HGRID_T hGrid)

{
 ESSG_STS_T   sts = ESS_STS_NOERR;
 ESSG_SHORT_T     sOption;
 ESSG_SHORT_T     tmpShort;
 ESSG_PVOID_T     pOption;

 /* connect the grid to a database on the server */
```

```
 sts = EssGConnect(hGrid, "Rainbow", "Admin", "Password", "Demo", "Basic",
         ESSG_CONNECT_DEFAULT);

 tmpShort = 2;
 sOption = ESSG_OP_DRILLLEVEL;
 pOption = (ESSG_PVOID_T)tmpShort;

 /* set grid option */
 if(!sts)
   sts = EssGSetGridOption(hGrid, sOption, pOption);

 if(!sts)
   EssGDisconnect(hGrid, 0);
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGSetGridPerspective

この関数はグリッドのパースペクティブを設定します。パースペクティブはグリッド・オプションに似ています。設定したパースペクティブが有効な場合、グリッド・コンテキストは同一になります。

構文

```
ESSG_FUNC_M EssGSetGridPerspective(
hGrid
,
sAttrdim
,
sPerspectiveType
,
pPerspectiveString
)
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	EssGNewGrid()から戻されるハンドル。
sAttrdim	ESSG_STR_T	パースペクティブによる設定が必要な属性次元名
sPerspectiveType	ESSG_SHORT_T	パースペクティブのタイプ。1091 ページの「グリッドのパースペクティブ・タイプ」を参照してください。

パラメータ　　データ型　　説明

pPerspectiveString ESSG_STR_T

pPerspectiveString　　(m1,m2,m3,.....)

指定した属性次元への適用が必要なパースペクティブ・タプルです。1 つ以上の「独立」次元(attrDim の場合)のレベル 0 メンバーが入力タプルに含まれます。

パースペクティブ・タプルに 1 つの「独立」次元メンバーが含まれていない場合、現在のクエリー/計算コンテキストの同一次元メンバーが使用されます。

属性次元の明示的なパースペクティブが欠落している場合、パースペクティブに ESSG_PERSP_REALITY をデフォルトで使用します。この引数には、有効なタプルを必要とする ESSG_PERSP_EXPLICIT 以外の PerspectiveType では NULL を指定できます。

戻り値

- 0 - 正常終了の場合
- エラー番号 - 失敗した場合

関連トピック

- EssGGetGridPerspective

EssGSetPath

現在のプロセスに対して、ESSBASEPATH 環境変数を設定します。

構文

```
ESSG_FUNC_M
EssGSetPath
(
pszPath
);
```

パラメータ　データ型　　説明

pszPath;　ESSG_STR_T ESSBASEPATH 環境変数を説明する文字列へのポインタ

備考

- EssGInit()を呼び出す前に EssGSetPath()を呼び出します。
- pszPath の値は ESSG_PATHLEN で定義されているように、シングル・バイト文字換算で 120 文字以内で指定します。
- pszPath は、現在のプロセスにのみ適用されます。
- Essbase DLL はシステム・パスからアクセスできる必要があります。EssGSetPath()は Essbase DLL のパスを解決しません。

戻り値

- 正常終了の場合は、ESSG_STS_NOERR が戻されます。
- pszPath が長すぎる場合は、API_NAME_TOO_LONG(1030009)が戻されます。

例

```
    ESS_STS_T
ESSG_SetPath(ESS_STR_T pszPath)
{
 ESS_STS_T  sts
 ESSG_STR_T pszPath = "C:\Hyperion\products\Essbase";
 sts = EssGSetPath (pszPath);
 return sts;
}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGTerm

この関数は、グリッド API を終了します。

構文

```
ESSG_FUNC_M
EssGTerm
 (
Handle
);
```

パラメータ データ型 説明

Handle	ESSG_HANDLE_T	EGAPI のインスタンスのハンドル。

備考

- この呼出しは必須です。
- この関数を実行すると、グリッド API の使用が終了します。
- この呼出しは、セッションごとに 1 回のみ実行します。呼出しできる場所はセッションの最後のみです。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_INIT_T   InitStruct;
ESSG_HANDLE_T  Handle;

InitStruct.ulVersion = ESSG_VERSION;
InitStruct.ulMaxRows = 1000;
InitStruct.ulMaxColumns = 200;
InitStruct.pfnMessageFunc = ESS_NULL;
InitStruct.pUserdata = ESS_NULL;

/* initialize EGAPI */
sts = EssGInit(&InitStruct, &Handle);

/* terminate the EGAPI */
if(!sts)
  sts = EssGTerm(Handle);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGUnlock

この関数は、サーバーでロックされたブロックのロックを解除します。

構文

```
ESSG_FUNC_M
EssGUnlock
 (
hGrid, ulOptions
);
```

パラメータ	データ型	説明
hGrid	ESSG_HGRID_T	**EssGNewGrid** から戻されるハンドル。
ulOptions	ESSG_ULONG_T	今後使用するために予約されています。ゼロに設定する必要があります。

戻り値

正常終了の場合は、**ESSG_STS_NOERR** が戻されます。

アクセス

なし。

例

```
    ESSG_VOID_T ESSG_Unlock(ESSG_HGRID_T hGrid)

{
 ESSG_FUNC_M   sts = ESS_STS_NOERR;
 ESSG_ULONG_T     ulOptions = 0;

  /* connect the grid to a database on the server */
 sts = EssGConnect(hGrid, "Rainbow", "Admin", "Password", "Demo", "Basic",
    ESSG_CONNECT_NODIALOG);

 /* unlock the locked blocks at server */
 if(!sts)
   sts = EssGUnlock(hGrid, ulOptions);

 if(!sts)
   EssGDisconnect(hGrid, 0);

}
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

EssGUpdateLRO

リンク・オブジェクトの記述とコンテンツを更新します。

構文

```
ESSG_FUNC_M
EssGUpdateLRO
 (
hGrid, hLRO, pLroDesc, ulOption
);
```

パラメータ	データ型	説明
hGrid;	ESSG_HGRID_T	EssGNewGrid()から戻されるグリッド・ハンドル。
hLRO;	ESSG_HLRO_T	リンク・オブジェクトへのハンドル(EssGGetCellLinkResults()関数により DRILLDATA 構造体に戻されます)。
pLroDesc;	1100 ページの「ESSG_LRODESC_T」	更新する LRO の情報を含む構造体。
ulOption;	ESSG_ULONG_T	オブジェクトをサーバーに保管するかどうかを指定するオプション。サーバーに winapp や URL オブジェクトを保管するには ESS_STORE_OBJECT_API を使用します。セル・ノートをサーバー以外(インデックス・ファイル)に保管するには ESS_NOSTORE_OBJECT を使用します。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」
- EssGBeginCreateLRO
- EssGBeginDeleteLROs
- EssGDeleteLRO
- EssGGetLRODesc

EssGVersion

この関数は、API のバージョン番号を戻します。

構文

```
ESSG_FUNC_M
EssGVersion
 (
pulVersion
);
```

パラメータ　データ型　　説明

pulVersion ESSG_PULONG_T グリッド API の現在のバージョン番号へのポインタ。

備考

- クライアントによる再コンパイルまたは再リンクを要求する変更が発生するたびに番号が大きくなります。
- この関数を使用する前にグリッド API の初期化は不要です。

戻り値

正常終了の場合は、ESSG_STS_NOERR が戻されます。

アクセス

なし。

例

```
    #include <essapin.h>
#include <essgapin.h>

ESSG_FUNC_M   sts = ESS_STS_NOERR;
ESSG_ULONG_T  ulVersion;

/* get version number for the API */
sts = EssGVersion(&ulVersion);
```

関連トピック

- 1082 ページの「C グリッド API 関数の使用」
- 1093 ページの「C グリッド API の構造体」

16 CグリッドAPIの例

この章の内容

C グリッド API の例

この例では、基本の取得の実行に必要な手順を説明しています。次のグリッドでは、1 つのデータポイントが示された 5 つの次元テンプレートを示しています。

		Year	Product	Market
Actual	Sales	123.45		

次のコードは、データ構造体の設定方法と取得操作を実行するために必要な関数呼出しを示しています。

```
   /* This function allocates the necessary data to send to the server */

ESSG_PPDATA_T AllocTwoDims(ESSG_ULONG_T ulRows, ESSG_ULONG_T ulCols)
{
ESSG_PPDATA_T ppTemp;
ESSG_ULONG_T  ulIndex;

if(ulRows)
ppTemp = (ESSG_PPDATA_T) malloc(sizeof(ESSG_DATA_T*) * ulRows);
if(ppTemp == NULL)
return ppTemp;

memset(ppTemp, 0, (sizeof(ESSG_PDATA_T) * ulRows));

for (ulIndex = 0; ulIndex < ulRows; ulIndex++)
{
ppTemp[ulIndex] = (ESSG_PDATA_T)malloc(sizeof(ESSG_DATA_T) * ulCols);
if(ppTemp[ulIndex])
memset(ppTemp[ulIndex], 0, (sizeof(ESSG_DATA_T) * ulCols));
}
```

```
return ppTemp;
}


/* This function frees the memory allocated by AllocTwoDims */
void FreeTwoDim(ESSG_PPDATA_T ppDataToFree, ESS_ULONG_T ulRows)
{
    ESS_ULONG_T ulIndex;

    for (ulIndex = 0; ulIndex < ulRows; ulIndex++)
    {
        if(ppDataToFree[ulIndex]->usType == ESSG_DT_STRING)
        {
            free(ppDataToFree[ulIndex]->Value.pszStr);
        }
        free(ppDataToFree[ulIndex]);
    }
    free(ppDataToFree);
}

/* This function builds a table based on the above grid. */
/* Note: The items in the grid are hard coded.      */
ESSG_PPDATA_T BuildTable(ESSG_PRANGE_T pRange)
{
    ESSG_PPDATA_T  ppTable;
    ESS_ULONG_T   ulRow, ulCol;

    /* Your code would probably not be hard-coded here... */
    pRange->ulRowStart    = 0;
    pRange->ulColumnStart  = 0;
    pRange->ulNumRows    = 2;
    pRange->ulNumColumns   = 5;
    ppTable = AllocTwoDims(2, 5);

    /* ROW 1 */
    ppTable[0][0].usType = ESSG_DT_BLANK;
    ppTable[0][1].usType = ESSG_DT_BLANK;
    ppTable[0][2].usType = ESSG_DT_STRING;
    /* Some compilers allow you to specify \p to indicate */
    /* the length of the string */
    ppTable[0][2].Value.pszStr = "\pYear";
    ppTable[0][3].usType = ESSG_DT_STRING;
    ppTable[0][3].Value.pszStr = "\pProduct";
    ppTable[0][4].usType = ESSG_DT_STRING;
    ppTable[0][4].Value.pszStr = "\pMarket";

    /* ROW 2 */
    ppTable[1][0].usType = ESSG_DT_STRING;
    ppTable[1][0].Value.pszStr = "\pActual";
    ppTable[1][1].usType = ESSG_DT_STRING;
    ppTable[1][1].Value.pszStr = "\pSales";
    ppTable[1][2].usType = ESSG_DT_DOUBLE;
    ppTable[1][2].dblData = 123.45;
    ppTable[1][3].usType = ESSG_DT_BLANK;
    ppTable[1][4].usType = ESSG_DT_BLANK;

    return (ppTable);
```

```
}

/* This function makes the necessary calls to the */
/* EGAPI to perform a basic retrieval.      */
/* NOTE:  This example does not show the          */
/* initialization of the EGAPI or the grid.    */
/* Also, the hGrid is assumed to be external.   */
void CallEGAPI(void)
{
    ESSG_PPDATA_T  ppDataIn,
    ESSG_PPDATA_T  ppDataOut;
    ESSG_RANGE_T   rRangeDataIn,rRangeDataOut;
    ESSG_STS_T    sts;
    ESSG_ULONG_T   ulRow, ulCol;
    ESSG_USHORT_T  usState;

    /* Connect the grid to a database on the server */
    sts = EssGConnect(hGrid, "Server", "User", "Password",
                  "App", "Db", ESSG_CONNECT_DEFAULT);
    if (sts == 0)
    {
        ppDataIn = BuildTable(rRangeDatain);
        /* Start the retrieve operation */
        sts = EssGBeginRetrieve(hGrid, ESSG_RET_RETRIEVE);
    }
    if (sts == 0)
    {
        /* Send the entire grid to define the query */
        sts = EssGSendRows(hGrid, rRangeDatain, ppDataIn);
    }
    if (sts == 0)
    {
        /* We're done sending rows, perform the retrieval */
        sts = EssGPerformOperation(hGrid, 0);

        /* Free the data we built */
        FreeTwoDim(ppDataIn, rRangeDataIn.ulNumRows);
    }
    if (sts == 0)
    {
        /* Determine the results of the retrieve and how much data
         * is being returned.
         */
        sts = EssGGetResults(hGrid, 0, rRangeDataOut, usState);
    }
    if (sts == 0)
    {
        /* Get all of the data */
        sts = EssGGetRows(hGrid,0, rRangeDataOut,
            rRangeDateOut, ppDataOut);
    }
    if (sts == 0)
    {
        /* Interate though the data ... */
        /* First the rows */
        for (ulRow = rRangeDataOut.ulRowStart;
               ulRow < rRangeDataOut.ulNumRows;
```

```
            ulRow++)
        {
            /* Then the columns */
            for (ulCol = rRangeDataOut.ulColumnStart;
                   ulCol < rRangeDataOut.ulNumColumns;
                   ulCol++)
            {
                /* Here's a cell ... just render it. */
                switch (ppDataOut[ulRow][ulCol].usType)
                {
                    case (ESSG_DT_STRING):
                        DisplayString(ppDataOut[ulRow][ulCol].Value.pszStr);
                        break;
                    case (ESSG_DT_LONG):
                        DisplayValue(ppDataOut[ulRow][ulCol].Value.lData);
                        break;
                    case (ESSG_DT_DOUBLE):
                        DisplayValue(ppDataOut[ulRow][ulCol].Value.dblData);
                        break;
                    case (ESSG_DT_BLANK):
                        DisplayBlank();
                        break;
                    case (ESSG_DT_MISSING):
                        DisplayMissing();
                        break;
                    case (ESSG_DT_ZERO):
                        DisplayValue(0);
                        break;
                    case (ESSG_DT_NOACCESS):
                        DisplayNoAccess();
                        break;
                    case (ESSG_DT_MEMBEREX):
                          DisplayString(ppDataOut[ulRow][ulCol].Value.pszStr+1);
                          break;
                    default:
                        DisplayOops();
                        break;
                }
            }
        }
        /* Tell the API we don't care about this request any more */
        EssGEndOperation(hGrid, 0);
        /* Free the data returned */
        EssGFreeRows(hGrid, rRangeDataOut, ppDataOut);
    }

    /* Disconnect if you wish */
    EssGDisconnect(hGrid, 0);

}
```

C グリッド API ドリルスルーの例

```
void main(int argc, char *argv[])
{
    ESSG_STS_T    sts = ESS_STS_NOERR;
    ESSG_HGRID_T  hGrid;
    ESSG_HANDLE_T  Handle;
    ESSG_INIT_T   InitStruct;

    /* BEGIN: initialize grid handle and create a new grid */
    InitStruct.ulVersion    = ESSG_VERSION;
    InitStruct.ulMaxRows    = 1000;
    InitStruct.ulMaxColumns  = 200;
    InitStruct.pfnMessageFunc = ESS_NULL;
    InitStruct.pUserdata    = ESS_NULL;

    sts = EssGInit(&InitStruct, Handle);
    if (sts != ESS_STS_NOERR)
        return;

    sts = EssGNewGrid(Handle, hGrid);
    if (sts != ESS_STS_NOERR)
        return;
    /* END: initialize grid handle and create a new grid */


    ESSG_DTTest(Handle, hGrid);

    sts = EssGTerm(Handle);
}

void ESSG_DTTest(ESSG_HANDLE_T  Handle, ESSG_HGRID_T hGrid)
{
    ESSG_STS_T    errsts,
            sts       = ESS_STS_NOERR;
    ESSG_HLRO_T    hLRO      = 0;
    ESSG_PPDATA_T   ppDataIn;
    /* ESSG_PPDATA_T  ppDataOut; */
    ESSG_RANGE_T   rDataRangeIn,
            rDataRangeOut;
    ESSG_USHORT_T   usCells;
    ESSG_USHORT_T   usState    = 0;
    ESSG_RANGE_T   Range;
    ESSG_PDTHINST_T  pDTInst;
    ESSG_STR_T    ErrMesg;
    ESSG_ULONG_T   ErrSize    = 255;
    memset(&rDataRangeOut, 0, sizeof(ESSG_RANGE_T));
    ErrMesg = malloc(255);

    /* connect the grid to a database on the server */
    sts = EssGConnect(hGrid, server, user, pwd, app, db, ESSG_CONNECT_DEFAULT);

    if(sts == ESS_STS_NOERR)
    {
```

```
    ppDataIn = BuildTableForDrillThru (&rDataRangeIn);

    DisplayOutput(ppDataIn, rDataRangeIn);

    usCells      = 1;
    Range.ulRowStart   = 1;
    Range.ulColumnStart = 6;
    Range.ulNumRows    = 1;
    Range.ulNumColumns  = 1;
    sts = EssGBeginDrillOrLink(hGrid, usCells, &Range, ESSG_OPT_ZOOM);

}

if(sts == ESS_STS_NOERR)
    /* send the entire grid to define the query */
    sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);

if(sts == ESS_STS_NOERR)
{
    /* perform the drillorlink operation */
    sts = EssGPerformOperation(hGrid, 0);

    /* free the built data */
    FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}

if (sts ==ESS_STS_NOERR)
    sts = EssGDTRequestDrillThrough(hGrid, usCells, &Range, &pDTInst);

if (sts == ESS_STS_NOERR)
{
    /* Get the DT Info corresponding to the DT handle */
    sts = ESSGDTGetInfo(pDTInst);


  /* Set the password info for executing the drill through report */
    sts = ESSGDTSetInfo(pDTInst);

    /* determine the list of reports associated with the data cell range. */
    sts = ESSGDTListReports(pDTInst);

    /* Execute the report. Using index 0 for now as we have only one report */
    sts = EssGDTExecuteReport(pDTInst, 0);
    if ( sts )    /* Error Condition print error mesg */
        errsts = EssDTAPIGetError(pDTInst, &sts, ErrMesg, ErrSize);

    /* Get the headers for the report associated with the data cell range. */
    sts = ESSGDTGetHeader(pDTInst);
    if ( sts )    /* Error Condition print error mesg */
        EssDTAPIGetError(pDTInst, &sts, ErrMesg, ErrSize);

    /* Get the data for the report associated with the data cell range. */
    sts = ESSGDTGetData(pDTInst);
    if ( sts )    /* Error Condition print error mesg */
        EssDTAPIGetError(pDTInst, &sts, ErrMesg, ErrSize);
}
```

```
    sts = EssGDTEndDrillThrough(pDTInst);
}
```

ドリルスルーの詳細は、次の関数を参照してください:

EssGDTConnect

EssGDTEndDrillThrough

EssGDTExecuteReport

EssGDTGetData

EssGDTGetHeader

EssGDTGetInfo

EssGDTListReports

EssGDTRequestDrillThrough

EssGDTSetInfo

ESSG_OP_MEMBERANDUNIQUENAME の例

次の例は、Grid API 定数 ESSG_OP_MEMBERANDUNIQUENAME の使用方法を示しています。

```
   ESSG_VOID_T ESSG_BeginZoomIn(ESSG_HGRID_T hGrid)
{
ESSG_STS_T    sts = ESS_STS_NOERR;
ESSG_DATA_T  **ppDataIn;
ESSG_DATA_T  **ppDataOut;
ESSG_RANGE_T rDataRangeIn, rDataRangeOut;
ESSG_ULONG_T ulOptions;
ESSG_USHORT_T usCells;
ESSG_RANGE_T pZoomCells;
ESSG_USHORT_T usState;
ESSG_USHORT_T usMember2Len, usKey2Len;
ESSG_SHORT_T     sOption, sOptionGet;
  ESSG_SHORT_T     tmpShort, tmpShortGet, i;
  ESSG_PVOID_T     pOption, pOptionGet;
ESSG_STR_T  pMember, pKey, pOutStr;
ESSG_STR_T   pMember2, pKey2;


/* connect the grid to a database on the server */
sts = EssGConnect(hGrid, server, "essexer", pwd, app, db, ESSG_CONNECT_NODIALOG);

/* set grid option*/
tmpShort = ESSG_TRUE;
sOption = ESSG_OP_MEMBERANDUNIQUENAME ;
pOption = (ESSG_PVOID_T)tmpShort;   // pOption holds the actual value not a pointer

  sts = EssGSetGridOption(hGrid, sOption, pOption);
printf("EssGSetGridOption  sts  %ld\n",sts);
```

```
sOptionGet = ESSG_OP_MEMBERANDUNIQUENAME ;
 pOptionGet = &tmpShortGet;
 if(!sts)
{
   sts = EssGGetGridOption(hGrid, sOptionGet, pOptionGet);
 printf("EssGGetGridOption sts %ld\n",sts);
 printf("EssGSetGridOption set ESSG_OP_MEMBERANDUNIQUENAME TO %d\n",
(int)tmpShortGet);
 }


if(sts == 0)
{
ppDataIn = BuildTable(&rDataRangeIn);

ulOptions = ESSG_ZOOM_DOWN | ESSG_NEXTLEVEL;

pZoomCells.ulRowStart = 0;
pZoomCells.ulColumnStart = 2;
pZoomCells.ulNumRows = 1;
pZoomCells.ulNumColumns = 1;
usCells = 1;

/* start the zoom in operation */
sts = EssGBeginZoomIn(hGrid, usCells, &pZoomCells, ulOptions);
printf("EssGBeginZoomIn sts: %ld\n",sts);
}

//Display Input
DisplayOutput(ppDataIn, rDataRangeIn);
printf("\n\n");
if(sts == 0)
/* send the entire grid to define the query */
sts = EssGSendRows(hGrid, &rDataRangeIn, ppDataIn);

if(sts == 0)
{
/* perform the zoom-in */
sts = EssGPerformOperation(hGrid, 0);

/* Free the built data */
FreeTwoDim(ppDataIn, rDataRangeIn.ulNumRows);
}
if (sts == 0)
{
/* determine the results of the zoom-in */
sts = EssGGetResults(hGrid, 0, &rDataRangeOut, &usState);
}
if(sts ==0)
{
/* get all the data */
sts = EssGGetRows(hGrid, 0, &rDataRangeOut, &rDataRangeOut, &ppDataOut);
}

if(sts == 0)
{
```

```
DisplayOutput(ppDataOut, rDataRangeOut);


/* Retreive member and key from cell */
sts = EssGGetFromMemberwKey (((ppDataOut[1][0]).Value).pszStr, &pMember, &pKey);
   printf("After EssGGetFromMemberwKey\n Member: %s, Key: %s \n\n",
  pMember+2,
 pKey+2);

//Member is  "Qtr1", Key is "[2004].[Qtr1]", pOutStr is in the format
//nn<member-name>nn<key> - where nn is string length

usMember2Len = strlen("Qtr1");
pMember2 = malloc(usMember2Len+3);
   memset(pMember2, 0, usMember2Len+3);
usKey2Len = strlen("[2004].[Qtr1]");
 pKey2 = malloc(usKey2Len+3);
   memset(pKey2, 0, usKey2Len+3);

memcpy(pMember2, &usMember2Len, 2);
memcpy(pMember2+2, "Qtr1", usMember2Len);

memcpy(pKey2, &usKey2Len, 2);
memcpy(pKey2+2, "[2004].[Qtr1]", usKey2Len);

sts = EssGCreateMemberwKeyStr(pMember2, pKey2, &pOutStr);

/*Note: because not all elements in pOutStr are actual characters,
 e.g. the 2 bytes for the size of Member and size of Key, plus the
 \0 ending characters, the printf below does not display the actual
 contents of the array */
for (i=0;i < usMember2Len + usKey2Len + 4 + 2; ++i)
printf("%c", pOutStr[i]);


/* Free the returned data */
EssGFreeRows(hGrid, &rDataRangeOut, ppDataOut);
sts = EssGFreeMemberwKeyStr (pOutStr);

}

if( sts == 0)
{
EssGEndOperation(hGrid, 0);
EssGDisconnect(hGrid, 0);
}
}
```

ESSG_DT_MEMBERwKEY の例

次の例は、Grid API 定数 ESSG_DT_MEMBERwKEY の使用方法を示しています。

注： DisplayOutput は、次の ESSG_BeginZoomIn で呼び出される関数です。

```
   ESSG_VOID_T DisplayOutput(ESSG_PPDATA_T ppDataOut, ESSG_RANGE_T pRangeOut)
{
 ESSG_ULONG_T RowIndex, ColumnIndex;
  for (RowIndex = 0; RowIndex < pRangeOut.ulNumRows; RowIndex++)
   {
    for (ColumnIndex = 0; ColumnIndex < pRangeOut.ulNumColumns; ColumnIndex++)
    {
   switch(ppDataOut[RowIndex][ColumnIndex].usType)
     {
      case(ESSG_DT_STRING):
  printf("%s", ppDataOut[RowIndex][ColumnIndex].Value.pszStr+1);
  break;
      case(ESSG_DT_LONG):
     printf("%ld", ppDataOut[RowIndex][ColumnIndex].Value.lData);
  break;
      case(ESSG_DT_DOUBLE):
  printf("%g", ppDataOut[RowIndex][ColumnIndex].Value.dblData);
  break;
      case(ESSG_DT_BLANK):
       break;
      case(ESSG_DT_RESERVED):
       printf("#Reserved");
       break;
      case(ESSG_DT_ERROR):
       printf("#Error");
       break;
      case(ESSG_DT_MISSING):
       printf("#Missing");
  break;
  case(ESSG_DT_ZERO):
       printf("%ld", ppDataOut[RowIndex][ColumnIndex].Value.lData);
       break;
      case(ESSG_DT_NOACCESS):
       printf("#NoAccess");
       break;
      case(ESSG_DT_MEMBER):
  printf("%s", ppDataOut[RowIndex][ColumnIndex].Value.pszStr+1);
  break;
      case(ESSG_DT_MEMBERwKEY):
  printf("%s", ppDataOut[RowIndex][ColumnIndex].Value.pszStr+2);
  printf(" (Key = %s)", ppDataOut[RowIndex][ColumnIndex].Value.pszStr+5+
       strlen(ppDataOut[RowIndex][ColumnIndex].Value.pszStr+2));
  break;
      default:
       break;
     }
    if (ColumnIndex < pRangeOut.ulNumColumns - 1)
     {
      printf(",");
     }
    }
     printf("\n");
 }
 printf("\n");
 printf("\n");
```

```
}
```

BuildTable 例関数

次の例は、この例関数を呼び出します:

```
   ...
ESSG_PPDATA_T BuildTable (ESSG_PRANGE_T pRange)
{
ESSG_PPDATA_T  ppTable;
ESSG_STR_T   current_str;
ESSG_USHORT_T  slen = 0;

 pRange->ulRowStart = 0;
 pRange->ulColumnStart = 0;
 pRange->ulNumRows = 2
 pRange->ulNumColumns = 5;
 ppTable = AllocTwoDims(2, 5);

 /* ROW 1 */
 ppTable[0][0].usType = ESSG_DT_BLANK;
 ppTable[0][1].usType = ESSG_DT_BLANK;

 slen = strlen("Year");
 current_str = malloc(sizeof(ESSG_CHAR_T)*(slen+2));
 *current_str = slen;
 strcpy( (current_str + 1), "Year");
 ppTable[0][2].usType = ESSG_DT_STRING;
 ppTable[0][2].Value.pszStr = current_str;

 slen = strlen("Product");
 current_str = malloc(sizeof(ESSG_CHAR_T)*(slen+2));
 *current_str = slen;
 strcpy( (current_str + 1), "Product");
 ppTable[0][3].usType = ESSG_DT_STRING;
 ppTable[0][3].Value.pszStr = current_str;

 slen = strlen("Market");
 current_str = malloc(sizeof(ESSG_CHAR_T)*(slen+2));
 *current_str = slen;
 strcpy((current_str + 1), "Market");
 ppTable[0][4].usType = ESSG_DT_STRING;
 ppTable[0][4].Value.pszStr = current_str;

 /*** ROW 2 ***/
 slen = strlen("Actual");
 current_str = malloc(sizeof(ESSG_CHAR_T)*(slen+2));
 *current_str = slen;
 strcpy((current_str + 1), "Actual");
 ppTable[1][0].usType = ESSG_DT_STRING;
 ppTable[1][0].Value.pszStr = current_str;
 ppTable[1][1].usType = ESSG_DT_STRING;
```

```
 slen = strlen("Sales");
 current_str = malloc(sizeof(ESSG_CHAR_T)*(slen+2));
 *current_str = slen;
 strcpy( (current_str + 1), "Sales");
 ppTable[1][1].Value.pszStr = current_str;

 ppTable[1][2].usType = ESSG_DT_DOUBLE;
 ppTable[1][2].Value.dblData = 123.45;
 ppTable[1][3].usType = ESSG_DT_BLANK;
 ppTable[1][4].usType = ESSG_DT_BLANK;

 return (ppTable);
}
```

DisplayOutput 例関数

次の例は、この例関数を呼び出します:

```
   ESSG_VOID_T DisplayOutput(
             ESSG_HGRID_T hGrid,
             ESSG_PPDATA_T ppDataOut,
             ESSG_RANGE_T pRangeOut)
{
 if (!ppDataOut)
 {
  printf("Data area is empty !\n");
  return;
 }

 ESSG_ULONG_T  RowIndx, ColIndx;
 printf
        ("---- Row: %d Column: %d startRow: %d, startColumn: %d\n",
            pRangeOut.ulNumRows,
            pRangeOut.ulNumColumns,
            pRangeOut.ulRowStart,
            pRangeOut.ulColumnStart);


 for(RowIndx = 0; RowIndx < pRangeOut.ulNumRows; RowIndx++)
 {
  for (ColIndx = 0; ColIndx < pRangeOut.ulNumColumns; ColIndx++)
  {
   switch(ppDataOut[RowIndx][ColIndx].usType)
   {
   case(ESSG_DT_STRING):
   printf("%s", ppDataOut[RowIndx][ColIndx].Value.pszStr+1);
   break;

   case(ESSG_DT_LONG):
   printf("%ld",ppDataOut[RowIndx][ColIndx].Value.lData);
   break;
```

```
case(ESSG_DT_DOUBLE):
printf("%g", ppDataOut[RowIndx][ColIndx].Value.dblData);
break;

case(ESSG_DT_BLANK):
break;

case(ESSG_DT_RESERVED):
printf("#Reserved");
break;

case(ESSG_DT_ERROR):
printf("#Error");
break;

case(ESSG_DT_MISSING):
printf("#Missing");
break;

case(ESSG_DT_ZERO):
printf("%ld", ppDataOut[RowIndx][ColIndx].Value.lData);
break;

case(ESSG_DT_NOACCESS):
printf("#NoAccess");
break;

case(ESSG_DT_MEMBER):
printf("%s", ppDataOut[RowIndx][ColIndx].Value.pszStr+1);
break;

case(ESSG_DT_STRINGEX):
case(ESSG_DT_MEMBEREX):
printf("%s",  ppDataOut[RowIndx][ColIndx].Value.pszStr+2);
break;

case ESSG_DT_SMARTLIST:
{
ESSG_STR_T val = 0;
printf("SmartList");
EssGGetFormattedValue(hGrid,&ppDataOut[RowIndx][ColIndx],&val);
if(val)printf("-%s",val);
EssGGetSmartlistforCell (hGrid,&ppDataOut[RowIndx][ColIndx],&val);
if(val)printf("Name -%s",val);
}
break;

case ESSG_DT_DATE:
{
ESSG_STR_T val = 0;
printf("Date");
EssGGetFormattedValue(hGrid,&ppDataOut[RowIndx][ColIndx],&val);
if(val)printf("-%s",val);
}
break;

case ESSG_DT_MNGLESS:
```

```
   printf("MeaningLess");
   break;

   default:
   break;
}
              printf("(%d, %x)",ppDataOut[RowIndx][ColIndx].usType, ppDataOut[RowIndx]
[ColIndx].pAttributes);

              if (ColIndx < pRangeOut.ulNumColumns - 1)
                 printf(",");
          }
              printf("\n");
      }
}
```

FreeTwoDim 例関数

```
   ESSG_VOID_T FreeTwoDim(ESSG_PPDATA_T ppDataToFree,
           ESSG_ULONG_T ulRows)
{
 ESSG_ULONG_T ulIndex;

 for (ulIndex = 0; ulIndex < ulRows; ulIndex++)
 {
  if (ppDataToFree[ulIndex]->usType == ESSG_DT_STRING)
  {
   free(ppDataToFree[ulIndex]->Value.pszStr);
  }
  free(ppDataToFree[ulIndex]);
 }
 free(ppDataToFree);
}
```

delete

17 CグリッドAPIのエラー・コード

C グリッド API から戻されるエラー・コードは、次の 3 タイプです:

- **成功**- API から値 0 が戻されます
- **サーバー・エラー**- API から非常に大きな番号が戻されます。番号の説明は `esserror.h` ファイルに記載されています
- **API エラー**- API から 1100001 で始まる番号が戻されます。この種の値は、グリッド API C 言語ヘッダー・ファイルである `essgapi.h` で定義されています。

グリッド API の初期化の際に有効なエラー関数コールバックを提供すると、有効な hGrid が渡された場合にこのコールバックがすべての EGAPI エラーに対して呼び出されます。そしてエラー関数が 0 を戻すことで、EGAPI 内で提供されているデフォルトのエラー処理ユーザー・インタフェースを停止できます。エラー関数から 0 以外の値が戻された場合、またはエラー関数が提供されていない場合は、EGAPI はシステム固有のユーザー・インタフェースを使用してエラー・メッセージを表示します。

次の表は、グリッド API 呼出しが失敗した場合に戻されるエラー・ステータス定数をまとめたものです。

エラー・コード	値
ESSG_ERR_INITREQUIRED	1100001
ESSG_ERR_CONNECTREQUIRED	1100002
ESSG_ERR_INVALIDHANDLE	1100003
ESSG_ERR_INVALIDGRID	1100004
ESSG_ERR_CANNOTINIT	1100005
ESSG_ERR_CANNOTCONNECT	1100006
ESSG_ERR_CANNOTCREATEGRID	1100007
ESSG_ERR_INVALIDVERSION	1100008
ESSG_ERR_CANNOTGETAPIINST	1100009
ESSG_ERR_CANNOTGETAPICTX	1100010
ESSG_ERR_INVALIDOPTION	1100011
ESSG_ERR_INVALIDRANGE	1100012

エラー・コード	値
ESSG_ERR_INVALIDDATA	1100013
ESSG_ERR_INVALIDROWORCOLMAX	1100014
ESSG_ERR_NULLARGUMENT	1100015
ESSG_ERR_CELLSREQUIRED	1100016
ESSG_ERR_RANGEREQUIRED	1100017
ESSG_ERR_INVALIDACTION	1100018
ESSG_ERR_INVALIDGRIDOPTION	1100019
ESSG_ERR_INVALIDFUNCTION	1100020
ESSG_ERR_MEMORY	1100021
ESSG_ERR_INVALIDROW	1100022
ESSG_ERR_INVALIDCOLUMN	1100023
ESSG_ERR_INVALIDPARM	1100024
ESSG_ERR_INVALIDCSLVERSION	1100025
ESSG_ERR_RANGEOVERLAP	1100026
ESSG_ERR_OPERATIONFAILED	1100027
ESSG_ERR_CANNOTSETOPTION	1100028
ESSG_ERR_INVALIDOPTIONVALUE	1100029
ESSG_ERR_EMPTYARGUMENT	1100030
ESSG_ERR_INVALIDLROHANDLE	1100031
ESSG_ERR_NOLROSAVAILABLE	1100032
ESSG_ERR_INVALIDLROTYPE	1100033
ESSG_ERR_GCINITFAIL	1100034
ESSG_ERR_GCSETLOCALEFAIL	1100035

第 V 部

Visual BasicのメインAPI

Visual Basic のメイン API の内容：

18 Visual BasicのメインAPIの使用

この章の内容

Visual Basic のメイン API の表記規則

Visual Basic の空文字列

「空文字列」はスペースで満たされた固定長文字列(ByVal As String *サイズ)または最初の位置に chr$(0)を持つ可変/固定長文字列です。原則として、入力パラメータを「空文字列」として参照できる場合は、前述のいずれかにできます。戻り値が同様の場合は、後者のみ可能です(空文字列のテストを容易にするため)。

Visual Basic での API 宣言の理解

プログラム内で、C 言語での宣言と同等な Visual Basic の宣言を使用します。次の表には C 言語の宣言、Visual Basic の ESB32.BAS ファイルにおける宣言方法、Visual Basic での呼出し方法を記載します。

C 宣言	Visual Basic 宣言	呼出しに使用する変数
文字列へのポインタ(LPSTR)	`ByVal S As String`	任意の文字列変数またはバリアント変数
整数へのポインタ(LPINT)	`I As Integer`	任意の整数変数またはバリアント変数
ロング型へのポインタ(LPDWORD)	`L As Long`	任意の Long 型変数またはバリアント変数
構造体へのポインタ(LPRECTなど)	`S As Rect`	ユーザー定義型の任意の変数

C 宣言	Visual Basic 宣言	呼出しに使用する変数
整数(INT, UINT, WORD, BOOL)	ByVal I As Integer	任意の整数変数またはバリアント変数
ハンドル(hWnd、hDC、hMenuなど)	ByVal h As Integer	任意の整数変数またはバリアント変数
Long型(DWORD, LONG)	ByVal L As Long	任意の Long 型変数またはバリアント変数
整数へのポインタ	I As Integer	I(0)などの配列における最初の要素
void型(void *)へのポインタ	As Any	任意の変数(文字列を持つ ByVal を使用する)
void型(関数の戻り値)	Subプロシージャ	適用外

プログラムのメイン・フォームでは、ブール変数 **ESB_TRUE** および **ESB_FALSE** を設定するために、次の行を挿入します。これらは、初期化構造体 **ESB_INIT_T** などのユーザー定義型で発生します:

```
   ESB_TRUE = 1
ESB_FALSE = 0
```

注： Visual Basic プログラムにおける C 言語 DLL の使用に関する情報は、Microsoft Visual Basic のマニュアルを参照してください。

Excel で使用できる Visual Basic 関数

Oracle Hyperion Smart View for Office では、Excel の Visual Basic Editor を使用して、Visual Basic for Applications (VBA)関数を用いた一般的なタスクのカスタマイズおよび自動化が可能です。

詳細は、Oracle Hyperion Smart View for Office User's Guide を参照してください。

Visual Basic の API ハンドル

- 1218 ページの「インスタンス・ハンドル」
- 1219 ページの「コンテキスト・ハンドル」

インスタンス・ハンドル

インスタンス・ハンドル(概念的にはファイル・ハンドルと同様)は、API へのプログラム・アクセスを示し、API 内で使用されるプログラム固有のリソースと設定を識別します。この種の識別は、複数のプログラムによって同時にアクセスされることのある DLL では必要です。**EsbInit**()の呼出しによって、プログラムが API を初期化すると、インスタンス・ハンドルが戻されます。

アプリケーションにおけるインスタンス・ハンドルの使用

インスタンス・ハンドルは、Visual Basic プログラムにおいて **ESB_HINST_T** 型として宣言されます。

インスタンス・ハンドルは、次の 2 つの呼出しに渡す必要があります。まず、コンテキスト・ハンドルを戻す **EsbLogin**()呼出し、次に API 内で使用されるプログラム固有のリソースを解放する API 終了関数 **EsbTerm**()呼出しです。

インスタンス・ハンドルを他のプログラム、子プロセス、またはスレッドに渡して、同じ API リソースと設定を使用して個別にログインできます。同じインスタンス・ハンドルを使用しているすべてのプログラム、プロセス、またはスレッドが、必ず API の終了前にログアウトすることを確認してください。

注： スレッドは別のスレッドのネットワーク・ステータス情報への上書きを防止するために、固有のインスタンス・ハンドル(phInstance)を必要とすることがあります。

コンテキスト・ハンドル

コンテキスト・ハンドルは、システムへのユーザーによる単一の有効なログインを示します。**EsbLogin**()の呼出しが正常終了すると、コンテキスト・ハンドルが戻されます。それを、引数としてコンテキスト・ハンドルを必要とする他の API 呼出しに渡すことができます。

アプリケーションでのコンテキスト・ハンドルの使用

コンテキスト・ハンドルは、Visual Basic プログラムにおいて **ESB_HCTX_T** タイプとして定義されます。

一般に、ユーザーがそのサーバーにログ・インしている間(つまり、**EsbLogout**()の呼出しが正常終了するまで)、コンテキスト・ハンドルは有効です。ただし、サーバー・シャットダウンなどのような場合、コンテキスト・ハンドルが無効になる場合があります。そのため、プログラムは、セッションの間にユーザーが再びログ・インできる方法を(たとえばメニュー・オプションまたは機能キーを通して)提供する必要があります。

注： コンテキスト・ハンドルは、API のインスタンスに固有であり、適切なインスタンスのリソースおよび設定を暗黙的に参照します。

複数のコンテキスト・ハンドル

API プログラムの単一のインスタンスは、1 つ以上の Essbase サーバー上で同じユーザー名または異なるユーザー名を使用して、**EsbLogin**()を複数回呼び出すことができます。**EsbLogin**()への各呼出しは、独自のコンテキスト・ハンドルを戻します。また、プログラムは、戻された各コンテキスト・ハンドルを追跡する必要があります。使用している 1 つのクライアント・アプリケーション当たり 255 個までのコンテキスト・ハンドルを同時に持つことができます。ただし、プログラムが単一のサーバー上でその処理をすべて実行する場合、一般に、コンテキスト・ハンドルを 1 つのみ使用し、必要に応じて異なるアプリケーションやデータベースの

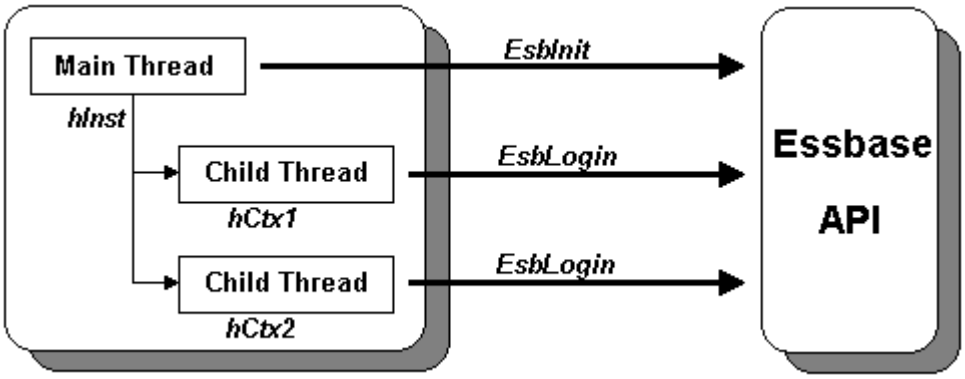
間で切り替える方が簡単です。このとき、**EsbSetActive()**関数または **EsbAutoLogin()** 関数のいずれかを使用します。

ローカル・コンテキスト・ハンドル

ローカルのオブジェクトおよびファイル(クライアント上のオブジェクト)に対する操作は、ローカルのコンテキスト・ハンドルを使用できます。ローカル・コンテキスト・ハンドルの使用およびローカル・コンテキストを参照してください。

コンテキスト・ハンドルの共有

一般に、複数のプログラム、プロセスまたはスレッドの間でコンテキスト・ハンドルを共有することは、その使用が排他的であることが保証される場合を除いて、望ましくありません。同じインスタンス・ハンドルを使用し、各プロセスに別々にログ・インする方がよい方法です。Essbase では、同じサーバー上で同じユーザー名を使用する複数のログインは、そのサーバー上のポートを 1 つのみ使用します。



Visual Basic API ファイル・オブジェクト

Essbase オブジェクトとは、データベース・アウトライン、計算スクリプトまたは他のデータのように、単に Essbase が使用するファイルのことです。Essbase は、API を通して、単に名前、ファイル・タイプ、および関連付けられているアプリケーションやデータベースによって、このようなファイルを参照できる、オブジェクト・システムを備えています。これにより、(Essbase の異なるプラットフォームおよび実装の間で異なる可能性がある)基本となるファイル・システムとは無関係に、オブジェクトを操作できます。

オブジェクトは、どの Essbase サーバーまたはクライアント上にあってもかまいません。サーバーのロック・メカニズムが、オブジェクトへのアクセスを制御し、十分な権限を持ったユーザーは、(**EsbGetObject()**関数を使用して)サーバー・オブジェクトをロックし、クライアントにそれらをコピーできます。また、(**EsbPutObject()**関数を使用して)そのオブジェクトを編集したり、サーバーへ戻せます。サーバー・オブジェクトは、読取り専用のアクセスにはロックなしに開くことができますが、サーバーに戻して保存することはできません。ユーザーはまた、自分で使用するためにクライアント・ワークステーションのオブジェクトを作成または編集したり、他のユーザーと共有するためにこれをサーバーに保存できます。クライアントはサーバー間のコピーはできません。

オブジェクトへのアクセス

API を通してオブジェクトにアクセスするとき、オブジェクト名は、オブジェクトのファイル名(拡張子なし)を表します。オブジェクト・タイプは、**ESB_OBJTYPE_xxx** の形式で API ヘッダー・ファイルで宣言されます(ここで、xxx は、**ESB_OBJTYPE_REPORT** でのように、特定のタイプを示します)。ほとんどのオブジェクトは、アプリケーションおよびデータベースと関連付けられています。しかし、計算スクリプトなどのオブジェクトやルール・ファイルは、アプリケーション・レベルで保管したり、アプリケーション内のデータベースで使用できます。

サーバー・オブジェクト・ファイルは、対応するアプリケーションまたはデータベースのサブディレクトリに物理的に位置します。ただし、サーバー・オブジェクト・ファイルを直接操作することは、通常望ましくありません。必ず、適切な API 関数を使用して、ファイルをローカルにコピーしてください。クライアント・オブジェクト・ファイルも、デフォルトでは、**ESB_INIT_T** の LocalPath 設定が指定するディレクトリのアプリケーションおよびデータベースのサブディレクトリに保管されています。これらのファイルは自由に操作や編集ができます。ただし、クライアントで編集しているサーバー・オブジェクトをロックおよびロック解除するときにプログラムが適切に操作されることを確認してください(必ず、編集前にオブジェクトをロックし、変更が保存されていてもいなくても、後でロックを解除してください)。

ローカル・コンテキスト

API を通してクライアント・マシン上のファイル・オブジェクトにアクセスする場合、使用する API オブジェクト関数用のローカル・コンテキストを作成する必要があります。ローカル・コンテキストを作成するためには、`EsbCreateLocalContext` 関数を使用します。この関数は、コンテキスト・ハンドルを戻します。このハンドルは、ログイン・コンテキスト・ハンドルのかわりに、オブジェクト API 関数のいずれにも渡すことができます。また、サーバーではなくローカルのクライアント・オブジェクト・システム上で、要求された操作を API に実行させます。プログラムが最初に API を初期化した直後に、ローカル・コンテキストを 1 度作成する必要があるだけです。

ローカル・コンテキストを作成する場合、API を終了する前にプログラムで **EsbDeleteLocalContext()**関数を呼び出してクリーン・アップする必要があります。

Visual Basic の API メッセージの処理

プログラムが API を呼び出すと、システム・メッセージおよびエラー・メッセージが生成されます。これらのメッセージの一部は Essbase サーバーによって戻され、その他は API 内部に渡されます。プログラムはなんらかの方法でこれらのメッセージを処理する必要があり、処理中の操作を中止させるエラーがある場合は、ユーザーに通知する必要があります。

この項では、API のメッセージ処理の仕組みを説明します。また、C および Visual Basic の開発者が、プログラム内でカスタム・メッセージのプロセスを実装する方法について説明します:

- 1222 ページの「Essbase API のメッセージの処理方法」

- 1222 ページの「Visual Basic プログラムにおけるメッセージ処理の使用」

Essbase API のメッセージの処理方法

Essbase では次のメッセージ・レベルがサポートされています:

- 通知メッセージ(通知のみ)
- 警告メッセージ(操作は続行)
- エラー・メッセージ(操作は中断)
- 深刻(操作は中断 - システムは不安定)
- 致命的(操作は中断 - システムは停止)

プログラムが、Essbase API のデフォルトのメッセージ処理を使用する場合、エラーまたはエラー以上のレベルのメッセージ(深刻または致命的)は、すべて現行アプリケーションの画面に表示されます。

Visual Basic プログラムにおけるメッセージ処理の使用

➤ Visual Basic プログラムにおいてメッセージ処理を実装するには、最初に次の手順に従ってプログラムをコード化してください:

1 **初期化関数 EsbInit()を呼び出す時点で、ESB_INIT_T 構造体の `ClientError` フィールドを ESB_TRUE に設定し、ESB_INIT_T の `ErrorStack` フィールドの値を定義します。**

2 **カスタム・メッセージ・ハンドラを定義するには、**

- 次の関数シグネチャを使用して、定義します(関数名は変更できます):

```
Public Function
EsbErrorHandler

(ByVal MsgNum As Long, ByVal Level As Long,
ByVal uLog As String, ByVal uMsg As String) As Long
```

- ESB_INIT_T 構造体の vbCallbackFuncAddress フィールドをカスタム・メッセージ・ハンドラ関数のアドレスに初期化します。例:

```
Sub ESB_Init()
Dim Init As ESB_INIT_T
Dim lx As Long

ESB_FALSE = 0
ESB_TRUE = 1

Init.Version = ESB_API_VERSION
Init.MaxHandles = 10
Init.LocalPath = "C:\Hyperion\products\Essbase"
Init.MessageFile = ""
```

```
 'This must be set to True
 Init.ClientError = ESB_TRUE
 Init.ErrorStack = 100
 'This is where the address of the custom function is set for
Init.vbCallbackFuncAddress
 Init.vbCallbackFuncAddress = GetProcAddress(AddressOf EsbErrorHandler)

 sts = EsbInit(Init, hInst)
 Debug.Print "EsbInit: sts = " & sts

End Sub

Public Function GetProcAddress(ByVal lngAddressOf As Long) As Long
 GetProcAddress = lngAddressOf
End Function

Public Function EsbErrorHandler(ByVal MsgNum As Long, ByVal Level As
Long, ByVal uLog As String, ByVal uMsg As String) As Long
 ' [ YOUR CODE GOES HERE ]
 MsgBox " Info " & MsgNum & ": Level: " & Level & ": " & uLog & ": " & uMsg
End Function
```

3 Essbase が提供するメッセージ処理を使用するためには、関数 **EsbGetMessage()**を呼び出し、**EssbaseAPI** を呼び出した後に戻された情報を取得します。情報を取得した後、必要に応じて、プログラムは情報を表示または処理できます。

ClientError フィールドおよび ErrorStack フィールドの設定

次に示すコードの一部は、ClientError および ErrorStack のフィールド・セットを示しています。

```
   Dim Init As ESB_INIT_T

.

.

.

Init.ClientError = ESB_TRUE

Init.ErrorStack = 100
```

EsbGetMessage()の呼出し

API 関数を実行すると、情報、警告および、エラー・メッセージはすべて、メッセージ・スタックに蓄積されます。ClientError が **ESB_TRUE** に設定されると、**EsbGetMessage()**を使用して、スタックから最上位のメッセージを取得できます。成功すると、**EsbGetMessage()**は、メッセージ・レベルへのポインタ、メッセージ番号へのポインタ、およびメッセージ文字列を戻します。また、内部メッセージ・スタック・ポインタを減らします。データが失われることがないように、関数がメッセージ・パラメータで空の文字列および番号パラメータで 0 を戻すまで、

EsbGetMessage()を呼び出す必要があります。詳しくは、EsbGetMessage を参照してください。

Visual Basic プログラムは、API への呼出しによって生成される情報を取得するには、**EsbGetMessage()**を呼び出す必要があります。API 呼出しによって生成された戻りコードが 0 でないとき、戻されたエラーまたはステータス情報を取得するため、これは重要です。さらに、戻りコードが 0 であるとき、追加情報の戻される可能性がある場合は、**EsbGetMessage()**を呼び出す必要があります。たとえば、**EsbLogin()**または **EsbAutoLogin()**への呼出しが正常終了した場合、最後のログ・インに関する役に立つ情報が戻されます。

注： カスタム・エラー処理関数を初期化すると、EsbGetMessage は取得に使用できません。

Visual Basic の API 関数の呼出し

この項では、API 関数の呼出し、インスタントとコンテキスト・ハンドルの使用方法および戻りコードの処理について説明します。

関数の宣言

Visual Basic で API を使用する場合、プログラムに正しい関数と定数の宣言を組み込んでおく必要があります。\ESSBASE\API\INCLUDE のファイル ESB32.BAS には、正しい関数と定数の宣言が含まれています。

32 ビットのプログラムでは 1 つの文字に 1 バイトではなく、2 バイト使用するため、32 ビットの Visual Basic プログラムでは ESB32.BAS ファイルが必要です。Essbase Visual Basic のデータ構造体の中には、1 バイトのデータ型を使用しているものもあるため、ESB32.BAS はこれらも 2 バイトを使用するように変更します。

プロジェクトに ESB32.BAS を追加します。あるいはグローバル宣言用に独自のファイルを使用している場合には、ESB32.BAS に含まれる宣言をそのファイルにコピーします。

インスタンス・ハンドルまたはコンテキスト・ハンドルの提供

EsbInit()への最初の呼出しで戻されたインスタンス・ハンドルを、**EsbLogin()**または **EsbTerm()**の呼出しに渡す必要があります。**EsbLogin()**で戻されたコンテキスト・ハンドルは、特定のログインに関連付けられたすべての関数呼出しに渡す必要があります。

戻りコードの処理

すべての Essbase API 関数では、**ESB_STS_T** のタイプのステータス・コードが戻されます。0 の戻りコードは関数が正常に実行されたことを、0 以外の値はエラーの状態を示します。エラーの戻り定数をすべて網羅したリストは、ヘッダー・ファイル ESSERROR.H および ESBERROR.BAS に含まれています。

注： あらゆる Essbase API 関数の戻りコードを常に確認する必要があります。戻りコードが 0 以外の値だった場合、関数から戻されるポインタや値は未定義となります。

Visual Basic の API 関数の呼出し順序

API では、プログラムによって特定の関数を他の関数より先に呼び出す必要があります。基本的な順序ルールは次のとおりです:

- プログラムは、他の API 関数を呼び出す前に、**EsbInit()**を呼び出す必要があります。
- プログラムは、コンテキスト・ハンドル引数を必要とする API 関数(ほとんどの API 関数)より前に、**EsbLogin()**または **EsbAutoLogin()**を呼び出す必要があります。さらに、使用する API オブジェクト関数のローカル・コンテキストを作成する場合は、コンテキスト・ハンドル引数を必要とする API 関数の前に、**EsbCreateLocalContext()**を呼び出す必要があります。
- 一部の API 関数では、アクティブなアプリケーションとデータベースの設定が必要です。そのためには、先にプログラムで **EsbSetActive()**または **EsbAutoLogin()**を呼び出します。
- プログラムがコンテキスト・ハンドルのために **EsbLogout()**を呼び出した後は、そのハンドルを API 関数に渡さないようにしてください。
- **EsbTerm()**を呼び出した後には、プログラムから、**EsbInit()**を除くいかなる API 関数も呼び出さないようにしてください。

Visual Basic API タスクの順序に関するトピック:

- 1225 ページの「VB API タスクの一般的な順序」
- 1226 ページの「Visual Basic API の初期化」
- 1227 ページの「Essbase サーバーへのログイン」
- 1227 ページの「アクティブなアプリケーションとデータベースの選択」
- 1228 ページの「データの取得と更新」
- 1229 ページの「データベースの再計算」
- 1231 ページの「Essbase サーバーからのログアウトと API の終了」

VB API タスクの一般的な順序

以下は、単純な API アプリケーションの通常の操作順序です:

1. **ESB_INIT_T** 構造体を作成し、初期化します。
2. **EsbInit()**を呼び出して、API を初期化します。
3. **EsbLogin()**または **EsbAutoLogin()**を呼び出して、必要なサーバーにログインします。

4. **EsbSetActive()**または **EsbAutoLogin()**を呼び出して、アクティブ・アプリケーションおよびデータベースを選択します。
5. **EsbReport()**または関連する関数を呼び出して、データを取得(またはロック)します。
6. **EsbUpdate()**または関連する関数を呼び出して、データを更新します。
7. **EsbCalc()**または関連する関数を呼び出して、データベースの再計算を実行します。
8. **EsbReport()**または関連する関数を呼び出して、データに対するレポートを作成します。
9. **EsbLogout()**を呼び出して、サーバーからログアウトします。
10. **EsbTerm()**を呼び出して、API を終了します。

Visual Basic API の初期化

プログラムは、他の EssbaseAPI 関数を呼び出す前に、**EsbInit()**関数を呼び出すことによって、API を初期化する必要があります。**EsbInit()**は、内部 API 状態変数をすべて初期化し、プログラムの要件に API を適合させることができます。

EsbInit()の前に呼び出すことができる唯一の関数は、**EsbGetAPIVersion()**です。

呼出し側プログラムは **EsbInit()**関数に初期化構造体を渡す必要があります。この構造体は、`ESB32.BAS` で **ESB_INIT_T** と定義されています。それには、API をカスタマイズし、API のデフォルトを設定するために使用される、一連のフィールドが含まれます。**EsbInit()**を呼び出す前に、この構造体のインスタンスを宣言し、関連するフィールドを初期化する必要があります。

EsbInit()関数は、インスタンス・ハンドルを戻します。これは、それが引数として API ログイン関数に渡されます。

Visual Basic 初期化構造体の宣言

Visual Basic では、それを使用するプロシージャで初期化構造体を宣言できます。下に示した構造体は、`ESB32.BAS` から取ったものです。

```
  Type ESB_INIT_T
 Version   As Long         ' version of API
 MaxHandles  As Integer        ' maximum number of context handles required
 LocalPath  As String * ESB_PATHLEN  ' local path to use for file operations
 MessageFile As String * ESB_PATHLEN  ' full path name of message database file
 HelpFile   As String * ESB_PATHLEN  ' full path name of help file
 ClientError As Integer        ' allows use of a pseudo client error handler
 ErrorStack  As Integer        ' size of the error message stack
End Type
```

このコードでは、フィールドは次のように定義されています:

- `Version` フィールドは、API の現在のバージョンを示します。

- MaxHandles フィールドは、プログラムが同時に要求できるコンテキスト・ハンドルの最大数を含みます。デフォルトは 255 です。
- LocalPath フィールドには、クライアント上でのファイルおよびオブジェクトの操作に使用するデフォルトのローカルのパス名が含まれています。デフォルトは$ESSBASEPATH\CLIENT です。ここで$ESSBASEPATH は ESSBASEPATH 環境変数によって定義されます。
- MessageFile フィールドには、メッセージ・データベース・ファイル ESSBASE.MDB の完全修飾パス名が含まれています。これが明示的に設定されていない場合、Essbase は最初に、ARBORMSGPATH 環境変数にある完全修飾パスを使用しようとします。それ以外の場合は、$ESSBASEPATH\BIN\ESSBASE.MDB を使用します。ここで$ESSBASEPATH は ESSBASEPATH 環境変数によって定義されています。ESSBASEPATH 変数が設定されていない場合、実行時にエラー・メッセージが戻されます。
- HelpFile フィールドには、アプリケーション・ヘルプ・ファイルの完全修飾名が含まれています。デフォルトでは、「ヘルプ」ボタンをクリックすると、Essbase システムのログインに関するヘルプ・トピックが表示されます。

  ESSBASEPATH が定義されていない場合、ヘルプ・ファイル名は NULL に設定されます。
- ClientError フィールドには値 **ESB_FALSE** または **ESB_TRUE** が含まれ、メッセージを取得するために **EsbGetMessage**()が使用できるかどうかを示します。
- ErrorStack フィールドには、**EsbGetMessage**()によって使用されるメッセージ・スタックのサイズを含みます。デフォルト値は 100 です。

Essbase サーバーへのログイン

一般に、**EsbInit**()を呼び出した後にプログラムが最初に実行すべきことは、ユーザーにサーバー名、ユーザー名、パスワード(または定義済のデフォルトを使用)の入力を求め、**EsbLogin**()を呼び出してサーバーにログインすることです。または、カプセル化されたログイン関数 **EsbAutoLogin**()を使用します。この呼出しが正常に終了した場合、戻されるコンテキスト・ハンドルは保管され、すべての後続の API 呼出しに対して使用されます。

アクティブなアプリケーションとデータベースの選択

コンテキスト・ハンドルに加えて、ログイン関数はログインしたユーザーがアクセスできるアプリケーションとデータベースのリストも戻します(プログラムは **EsbListDatabases**()関数を呼び出していつでもこのリストを取得できます)。プログラムでは、**EsbSetActive**()関数を呼び出して、ユーザーが特定のアプリケーションおよびデータベースを選択できるようにします。

EsbAutoLogin()をログインに使用する場合、アクティブ・アプリケーションおよびデータベースをオプションで設定できます。

Essbase アプリケーションの情報(たとえば、すでにロード済かどうか)を取得するには、**EsbGetApplicationState()**または **EsbGetApplicationInfo()**関数を呼び出します。特定のデータベースの情報を取得するには、**EsbGetDatabaseState()**または **EsbGetDatabaseInfo()**関数を呼び出します。これらの関数は、アクティブなアプリケーションおよびデータベースを設定する前に呼び出せます。

データの取得と更新

- 1228 ページの「データの取得」
- 1229 ページの「データの更新」

データの取得

レポートまたはその後の更新のために Essbase データベースからデータを取得するには、プログラムでレポート指定を使用する必要があります。レポート指定は、単一のテキスト文字列(長さが 64KB 未満の場合)、一連のテキスト文字列、またはファイルの形式を取れます。レポート・ファイルはクライアント・マシンまたは Essbase サーバー上に配置できます。

単一文字列としてのレポート指定の送信

レポート指定を単一文字列として送信するには、プログラムで **EsbReport()**を呼び出して使用して、サイズが 32KB 以下のレポート文字列全体を引数として渡します。`Output` フラグが **EsbReport()**への呼出しで TRUE に設定されている場合、プログラムは **EsbGetString()**を NULL 文字列が戻されるまで繰り返し呼び出して、戻されたレポート・データを読み込む必要もあります。その後、戻されたデータは、必要に応じて、表示、ファイルへの書込みまたは印刷が可能です。

連続文字列としてのレポート指定の送信

レポート指定を一連の文字列として送信するには、まず **EsbBeginReport()**を呼び出してから **EsbSendString()**を繰り返し呼び出して、レポート指定の個別の各文字列を送信します(Windows では、それぞれの文字列の長さは 32KB 以下にする必要があります)。最後に、**EsbEndReport()**を呼び出してレポート指定を終了します。`Output` フラグが **EsbBeginReport()**への呼出しで TRUE に設定されている場合は、プログラムは **EsbGetString()**を NULL 文字列が戻されるまで繰り返し呼び出して、戻されたレポート・データを読み込む必要もあります。戻されたデータはその後必要に応じて表示、ファイルに書き込み、または印刷できます。

レポート指定としてのファイルの送信

ファイルをレポート指定として送信するには、**EsbReportFile()**関数を使用してレポート・ファイル名を渡します。`Output` フラグが **EsbReportFile()**への呼出しで TRUE に設定されている場合、プログラムは **EsbGetString()**を NULL が戻されるまで繰り返し呼び出して、戻されたレポート・データを読み込む必要があります。戻されたデータはその後必要に応じて表示、ファイルに書き込み、または印刷できます。

データの更新

データベース内のデータを更新するには、最初に更新対象となるデータベース内のブロックをロックしてください。これを行うには、次のいずれかの操作を実行します:

- Output フラグを TRUE、Lock フラグを TRUE に設定して、前述のようにレポート指定を送信します。このレポートによるデータ出力を変更し、更新としてデータベースに送信できます。
- または、ロードの準備ができた新規データまたは変更データがある場合、プログラムは最初にそのデータをレポート指定として使用することで、適切なレポート関数を呼び出す時点で Output フラグを FALSE、Lock フラグを TRUE に設定して、データ・ブロックをロックできます。

データベースは、単一文字列、一連の文字列またはファイルのいずれかから更新できます。更新データ・ファイルはクライアント・マシン、Essbase サーバーのいずれかに保存されます。

単一文字列としての更新データの送信

更新を単一文字列として送信するには、**EsbUpdate()**を呼び出して文字列全体を引数として渡します(Windows では、文字列の長さは 32KB 以下にする必要があります)。Store フラグを **EsbUpdate()**への呼出しで TRUE に設定し、データベースが更新されるようにします。Unlock フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックのロックが解除され、他のユーザーがそれらのブロックを更新できます。

連続文字列としての更新データの送信

更新を一連の文字列として送信するには、**EsbBeginUpdate()**を呼び出し、次に **EsbSendString()**を繰り返し呼び出してすべてのデータを送信します(Windows では、個別の各データ文字列の長さは 32KB 以下にする必要があります)。最後に、**EsbEndUpdate()**を呼び出して更新を終了します。Store フラグを **EsbUpdate()**への呼出しで TRUE に設定し、データベースが更新されるようにします。Unlock フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックもロックが解除されます。

ファイルとしての更新データの送信

更新をファイルとして送信するには、**EsbUpdateFile()**関数を使用してデータ・ファイル名を渡します。Store フラグを **EsbUpdate()**への呼出しで TRUE に設定し、データベースが更新されるようにします。Unlock フラグも TRUE に設定されている場合、データが更新されると、データベース内のロック済データ・ブロックもロックが解除されます。

データベースの再計算

データベースのデータを更新した後は、連結した合計が正しくなるように再計算する必要があります。データベースを再計算するためには、デフォルト計算を実

行、または特定の計算スクリプトを送信できます。また、計算スクリプトをデフォルトの計算スクリプトとして設定できます。計算スクリプトは、単一文字列、連続文字列またはファイルとして送信できます。計算スクリプト・ファイルは、クライアント・マシンまたは Essbase サーバーのいずれにあってもかまいません。

単一文字列としての計算スクリプトの送信

単一文字列として計算スクリプトを送信するには、**EsbCalc()**を呼び出し文字列全体を因数として渡します(文字列の長さが 32KB を超えないように注意してください)。**EsbCalc()**の呼出しで `Calculate` フラグを TRUE に設定し、計算スクリプトが実行されるようにします。その後、一定の間隔で計算の進行状況を確認する必要があります。

連続文字列としての計算スクリプトの送信

連続文字列として計算スクリプトを送信するためには、最初に **EsbBeginCalc()**を呼び出し、その後、計算スクリプトの文字列をすべて送信するために **EsbSendString()** を繰り返し呼び出します(各文字列の長さが 32KB を超えないように注意してください)。最後に、**EsbEndCalc()**を呼び出して、スクリプトを終了します。データベースが再計算されるように、`EsbBeginCalc()`への呼出しでは **Calculate** フラグを TRUE に設定します。その後、一定の間隔で計算の進行状況を確認する必要があります。

ファイルとしての計算スクリプトの送信

ファイルとして計算スクリプトを送信するためには、**EsbCalcFile()**関数を使用して、計算スクリプトのファイル名を渡します。データベースが再計算されるように、`EsbCalcFile()`への呼出しにおいて **Calculate** フラグを TRUE に設定します。その後、一定の間隔で計算の進行状況を確認する必要があります。

デフォルトの計算スクリプトの使用

現在のデフォルト計算スクリプトを使用して、データベースを再計算するためには、**EsbDefaultCalc()**関数を使用します。データベースのデフォルトの計算スクリプトを設定するためには、**EsbSetDefaultCalc()**を使用して、単一文字列として計算スクリプトを渡します。ファイルからデフォルト計算スクリプトを設定するためには、**EsbSetDefaultCalcFile()**関数を使用して、計算スクリプト・ファイル名を渡します。**EsbGetProcessState()**を使用して、計算がいつ終了したかを判断します。

計算の進行状況の確認

データベースの計算が始まったら、一定の間隔(推奨は 5 秒)で、**EsbGetProcessState()** 関数を呼び出すことにより、計算の進行状況を確認します。この関数は、計算状態を示す構造体を戻します。計算が終了したことまたはエラーが発生したことが示されるまで、**EsbGetProcessState()**を呼び出します。また、**EsbCancelProcess()**関数を使用して、進行中の計算を取り消せます。

注意 計算が進行中である間、計算操作が正常に完了または取り消されるまで、同じコンテキスト・ハンドルを使用して、**EsbGetProcessState()**または**EsbCancelProcess()**以外の API 関数を呼び出そうとしないでください。計算が終了したことを **EsbGetProcessState()**が示した後、プログラムは、そのコンテキスト・ハンドルで他の API 操作の実行を続行できます。

Essbase サーバーからのログアウトと API の終了

すべてのデータベース操作が完了すると、アプリケーションは **EsbLogout()**を呼び出してログアウトします。これによって、データベース内に予約されていた内部リソースが解放され、サーバー上のログイン・ポートも解放されて別のユーザーが使用できるようになります。

アプリケーション・プログラムが終了する際、**EsbTerm()**関数を呼び出し、**EsbInit()**への元の呼出しから戻されたインスタンス・ハンドルを渡します。これによって、Essbase API が使用するすべてのリソースが解放されます。この関数を呼び出した後、**EsbInit()**を再度呼び出して API を再初期化しないかぎり、これ以上 API 呼出しはできません。

Visual Basic のメイン API 共通の問題と解決策

EsbLogin()、**EsbAutoLogin()**、**EsbGetString()**および **EsbListDatabases()**などへの呼出しによって、呼出し中の API 関数が戻りパラメータ内に大量のデータを生成する可能性がある場合、データを受信するための十分なバッファ・スペースを予約したかどうかを確認する必要があります。

表 8 では、問題の認識と解決をサポートします。

表 8　Visual Basic のメイン API 共通の問題と解決策

問題	解決策
プログラムによって、保護エラーが生成されています。	Visual Basic プログラムを使用して起こった問題であれば、API に渡されているポインタの宣言済の間接レベルを確認します。

問題	解決策
API 関数を呼び出すと、ユーザーのプログラムは Essbase エラーを生成します。	ほとんどの Essbase エラー・メッセージは説明を必要としません。問題がどこにあるかはかなり明白なはずです。ただし、次に注意すべきよくあるエラーをいくつか示します(%n は、コンテキストに特有の文字列と置換されるメッセージ引数を示します): ● 「NULL の引数(%1)が ESSAPI 関数%2 に渡されました」。このメッセージは、API 関数%2 に渡された 1 つ以上の引数が NULL だったことを示します。%1 は、最初の NULL 引数の数(1 から始まる)を示します。 ● 「ESSAPI 関数%1 の呼出しシーケンスが無効です」。このメッセージは、別の関数呼出しが必要なときに、API 関数(%1)への呼出しを行ったことを示します。たとえば、**EsbReport()**など、レポート関数を実行している場合は、NULL 文字列が戻されるまで **EsbGetString()**を繰り返し呼び出します。または、**EsbCalc()**など計算関数を実行している場合、計算が完了したことを戻り値が示すまで、**EsbGetProcessState()**を呼び出すことにより、繰り返し計算状態を確認します。 ● 「ESSAPI 関数%s ではローカル演算はできません」。ローカルのコンテキスト・ハンドルを、それを許可しない関数へ渡しました。ログイン・コンテキスト・ハンドルをそのかわりに使用します。 ● 「メッセージ・データベース%s を開けません」。メッセージ・データベースが、プログラムが動作しているマシン上でアクセスできません。Essbase が予期する場所にメッセージ・データベースがあることを確認してください。Essbase は、最初に `EsbInit()` に渡された初期化構造体の **MessagePath** フィールドを調べます。次に、ARBORMSGPATH 環境変数が指定するディレクトリおよびファイル名を調べ、最後に、`$ESSBASEPATH\BIN` ディレクトリを調べます。ここで、$ESSBASEPATH は環境変数です。メッセージ・データベースがこれらのディレクトリのいずれでも利用できない場合、Essbase は実行時にエラー・メッセージを戻します。どの設定を Essbase が使用するか確認し、その後、指定された場所にメッセージ・データベースがあることを確認します。第 3 章「Essbase と使用中の製品との統合」を参照してください。
プログラムが確実に API 関数から Essbase エラーの戻りコードを受け取っていてもメッセージが表示されないか、または「メッセージ・データベース内のメッセージ#%1 に対するメッセージはありません」というメッセージが生成されます。	内部 API エラーにはメッセージを表示できないものがありますが、これは通常、メッセージが発生した際にユーザーのコンテキスト情報が使用できないためです。この場合には、関数から戻されたエラー・コードを書き留めて、ヘッダー・ファイル `ESSERROR.H` のエラー・メッセージのリストを参照して、対応するメッセージ・テキストを見つけます。

ドリルスルー Visual Basic API の例

```
   Attribute VB_Name = "Module3"
Dim sts As Long
  Dim hInst As Long
  Dim hDestInst As Long
  Dim hCtx As Long
  Dim hDestCtx As Long
  Dim Server As String * ESB_SVRNAMELEN
  Dim User As String * ESB_USERNAMELEN
  Dim Password As String * ESB_PASSWORDLEN
  Dim AppName As String * ESB_APPNAMELEN
  Dim DbName As String * ESB_DBNAMELEN
```

```
Sub ESB_GetVersion()
 Dim sts As Long
 Dim Release As Integer
 Dim Version As Integer
 Dim Revision As Integer
 sts = EsbGetVersion(hCtx, Release, Version, Revision)
 Debug.Print "EsbGetVersion: sts = " & sts
 Debug.Print "Release: " & Release
 Debug.Print "Version: " & Version
 Debug.Print "Revision: " & Revision
End Sub

Sub ESB_Init()
 Dim Init As ESB_INIT_T

 ESB_FALSE = 0
 ESB_TRUE = 1

 Init.Version = ESB_API_VERSION
 Init.MaxHandles = 10
 Init.LocalPath = "C:\install\zolahit\products\Essbase\EssbaseClient"
 ' Use default message file
 Init.MessageFile = ""
 ' Use EsbGetMessage to retrieve
 ' messages
 Init.ClientError = ESB_TRUE
 Init.ErrorStack = 100
 'Init.vbCallbackFuncAddress = GetProcAddress(AddressOf EsbErrorHandler)

 sts = EsbInit(Init, hInst)
 'MsgBox ("EsbInit = " & sts)
 Debug.Print "EsbInit: sts = " & sts

 'For copy objects between servers
 'sts = EsbInit(Init, hDestInst)
 'MsgBox ("EsbInit = " & sts)
 'Debug.Print "EsbInit: sts = " & sts
End Sub

Public Function GetProcAddress(ByVal lngAddressOf As Long) As Long
 GetProcAddress = lngAddressOf
End Function

Public Function EsbErrorHandler(ByVal MsgNum As Long, ByVal Level As Long, ByVal uLog
As String, ByVal uMsg As String) As Long
 If Level >= ESB_LEVEL_ERROR Then
  MsgBox "Error: " & MsgNum & " - " & uMsg
 End If

 'MsgBox " Info " & MsgNum & ": Level: " & Level & ": " & uLog & ": " & uMsg
End Function

Sub ESB_GetMessage()
 Dim DbName As String
 Dim FilterName As String
 Const szMessage = 256
```

```
 Dim Message As String * szMessage
 Dim Number As Long
 Dim Level As Integer
 Dim sts As Long
 Dim Object As ESB_OBJDEF_T
 Dim hOutline As Long
 Dim hMemberProfit As Long

 Object.hCtx = hCtx
 Object.Type = ESB_OBJTYPE_OUTLINE
 Object.AppName = "Temp"
 Object.DbName = "Basic"
 Object.FileName = "Basic"
 sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES, ESB_YES, hOutline)
 Debug.Print "EsbOtlOpenOutline: sts = " & sts

 sts = EsbOtlFindMember(hOutline, "100-10", hMember)
 Debug.Print "EsbOtlFindMember: sts = " & sts

 If sts > 0 Then
  sts = EsbGetMessage(hInst, Level, Number, Message, szMessage)
  Do While Mid$(Message, 1, 1) <> Chr$(0)
   Debug.Print Level
   Debug.Print Number
   Debug.Print Message
   sts = EsbGetMessage(hInst, Level, Number, Message, szMessage)
   Debug.Print "EsbGetMessage: sts = " & sts
  Loop
 End If
End Sub

Sub ESB_Login()
 Dim Items As Integer
 Dim AppDb As ESB_APPDB_T

 Server = "ppamu-pc1"
 User = "essexer"
 Password = "password"
 sts = EsbLogin(hInst, Server, User, Password, Items, hCtx)
 Debug.Print "EsbLogin: sts = " & sts
 'For n = 1 To Items
  ' sts = EsbGetNextItem(hCtx, ESB_LAPPDB_TYPE, AppDb)
  ' Debug.Print "EsbGetNextItem: sts = " & sts
  ' Debug.Print "App Name: "; AppDb.AppName
  ' Debug.Print "Db Name: "; AppDb.DbName
 ' Next

 'For copy objects between servers
 'sts = EsbLogin(hDestInst, "qtfsun1:1501", User, Password, Items, hDestCtx)
 'Debug.Print "EsbLogin: sts = " & sts
End Sub

Sub ESB_AutoLogin()
 Dim pOption As Integer
 Dim pAccess As Integer

 Server = "localhost"
```

```
  'User = "essexer"
  'Password = "Password"
  'AppName = "sample"
  'DbName = "basic"

  'pOption = ESB_AUTO_NODIALOG + ESB_AUTO_NOSELECT
  pOption = ESB_AUTO_DEFAULT
  sts = EsbAutoLogin(hInst, Server, User, Password, AppName, DbName, pOption,
pAccess, hCtx)
  'MsgBox ("EsbAutoLogin = " & sts)
  Debug.Print "EsbAutoLogin: sts = " & sts
 ' Call Esb_runreport
End Sub

Sub ESB_LoginSetPassword()
  'Dim hInst    As Long
  'Dim Server   As String * ESB_SVRNAMELEN
  'Dim User    As String * ESB_USERNAMELEN
  'Dim Password  As String * ESB_PASSWORDLEN
  Dim NewPassword  As String * ESB_PASSWORDLEN
  Dim Items    As Integer
  Dim AppDb    As ESB_APPDB_T

  Server = "stiahp1:1501"
  User = "essexer"
  Password = "password"
  NewPassword = "password2"
  sts = EsbLoginSetPassword(hInst, Server, User, Password, NewPassword, Items, hCtx)
  Debug.Print "EsbLoginSetPassword: sts = " & sts

  For N = 1 To Items
   sts = EsbGetNextItem(hCtx, ESB_LAPPDB_TYPE, AppDb)
   Debug.Print "EsbGetNextItem: sts = " & sts
   Debug.Print "App Name: "; AppDb.AppName
   Debug.Print "Db Name: "; AppDb.DbName
  Next

  'Reset password back to original
  NewPassword = "password"
  sts = EsbLoginSetPassword(hInst, Server, User, Password, NewPassword, Items, hCtx)
  Debug.Print "EsbLoginSetPassword: sts = " & sts
End Sub

Sub ESB_SetActive()
  Dim AppName As String
  Dim DbName As String
  Dim pAccess As Integer
  Dim sts As Long

  'AppName = "Bugs"
  'DbName = "09129823"

  AppName = "vb"
  DbName = "Basic"

  sts = EsbSetActive(hCtx, AppName, DbName, pAccess)
  Debug.Print "EsbSetActive: sts = " & sts
```

```
End Sub

Sub ESb_GetStoresInfo() '(Chnl As String)
 Dim Object As ESB_OBJDEF_T

 Object.hCtx = hCtx
 Object.Type = ESB_OBJTYPE_OUTLINE
 Object.AppName = AppName
 Object.DbName = DbName
 Object.FileName = DbName
 Dim hMember As Long
 Dim ihMember As Long
 Dim MbrInfo As ESB_MBRINFO_T
 Dim Counts As ESB_MBRCOUNTS_T

 sts = EsbSetActive(hCtx, AppName, DbName, Access)

 Dim hMemberJan As Long
 Dim MbrChldCnt As Long
 Dim x As Integer
 Dim Parent As String
 Dim found As Boolean
 Dim img As Integer
 Dim Member As String
 Dim szAlias As String * ESB_MBRNAMELEN
 Dim Alias As String
 Dim levelnum As String
 Dim ShareStat As Integer

 Dim tLevelName As String * ESB_MBRNAMELEN
 Const AltGroup As String = "ALT_GROUP"
 'Dim LevelName As String * ESB_MBRNAMELEN
 sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES, ESB_YES, hOutline)

 If sts = 0 Then
  sts = EsbOtlFindMember(hOutline, "JOHNSON, ROGER", hMemberJan)
  'sts = EsbOtlFindMember(hOutline, "GMM_A", hMemberJan)
  If hMemberJan = 0 Then
   sts = EsbOtlFindAlias(hOutline, "JOHNSON, ROGER", "default", hMemberJan)
  End If
 End If

 If sts = 0 And hMemberJan <> 0 Then
  sts = EsbOtlGetMemberInfo(hOutline, hMemberJan, MbrInfo)
  MsgBox ("Member Name = " & MbrInfo.szMember)
  Member = MbrInfo.szMember
  levelnum = MbrInfo.usLevel
  ShareStat = MbrInfo.usShare
  MsgBox ("Shared Member = " & ShareStat)
 End If

 MbrChldCnt = MbrInfo.ulChildCount
 ' If ShareStat <> ESB_SHARE_SHARE Then

 'Do While x <= MbrChldCnt
 For x = 1 To MbrChldCnt
  If x = 1 Then
```

```
    sts = EsbOtlGetChild(hOutline, hMemberJan, hMember)
    'sts = EsbOtlGetMemberInfo(hOutline, hMember, MbrInfo)
    'MsgBox ("Child Member Name = " & MbrInfo.szMember)
   Else
    sts = EsbOtlGetNextSibling(hOutline, hMemberJan, hMember)
    ' sts = EsbOtlGetMemberInfo(hOutline, hMember, MbrInfo)
    ' MsgBox ("Sibling Member Name = " & MbrInfo.szMember)
   End If

  'Next

   sts = EsbOtlGetMemberInfo(hOutline, hMember, MbrInfo)
    MsgBox ("Sibling Member Name = " & MbrInfo.szMember)
    ' szAlias = ""
    'sts = EsbOtlGetMemberAlias(hOutline, hMember, "", szAlias)
    'sts = EsbOtlGetLevelName(hOutline, sRoot, MbrInfo.usLevel, tLevelName)
    'If sts > 0 Then tLevelName = ""

   'Alias = sTrim(szAlias)
   'Member = sTrim(MbrInfo.szMember)

  Next
End Sub

Sub ESB_Logout()

  sts = EsbLogout(hCtx)
  'MsgBox ("EsbLogout = " & sts)
  Debug.Print "EsbLogout: sts = " & sts
End Sub

Sub ESB_Term()

  sts = EsbTerm(hInst)
  'MsgBox ("EsbTerm = " & sts)
  Debug.Print "EsbTerm: sts = " & sts
End Sub

Public Sub ESB_LROListObjects()
  Dim UserName As String * ESB_USERNAMELEN
  Dim listDate As Long
  Dim Items As Integer
  Dim Desc As ESB_LRODESC_API_T
  Dim i As Integer
  Dim j As Integer
  Dim CutOffDate As Date
  Dim MemberName As String * ESB_MBRNAMELEN

  Const ESB_REFERENCE_DATE = #1/1/1970#
  UserName = "essexer"
  CutOffDate = #9/21/2007#
  'CutOffDate = #1/2/1970#
  listDate = DateDiff("s", CutOffDate, ESB_REFERENCE_DATE)
  'listDate = DateDiff("s", ESB_REFERENCE_DATE, CutOffDate)
  'listDate = -1

  sts = EsbLROListObjects(hCtx, UserName, listDate, Items)
```

```
 Debug.Print "EsbLROListObjects: sts = " & sts

 Debug.Print "Number of LRO(s): " & Items

 If sts = 0 Then
  For i = 1 To Items

   Debug.Print "LRO # " & i; ":"

   sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE, Desc)
   Debug.Print "EsbGetNextItem: sts = " & sts
   Debug.Print "Object Type: " & Desc.ObjType
   Select Case (Desc.ObjType)
    Case 0
     Debug.Print "Cell notes: " & Desc.note
    Case 1
     Debug.Print "Object Name: " & Desc.lroInfo.ObjName
     Debug.Print "Object Description: " & Desc.lroInfo.objDesc
    Case 2
     Debug.Print "Object Name: " & Desc.lroInfo.ObjName
     Debug.Print "Object Description: " & Desc.lroInfo.objDesc
   End Select
   Debug.Print "Member Combination:"
   For j = 1 To Desc.memCount
    sts = EsbLROGetMemberCombo(hCtx, j, MemberName)
    Debug.Print "   " & MemberName
   Next j

  Next i
 End If
End Sub

Sub ESB_SetUser()
 Dim sts As Long
 Dim UserInfo As ESB_USERINFO_T

 UserInfo.Name = "Test"
 UserInfo.Type = ESB_TYPE_USER
 UserInfo.Access = ESB_ACCESS_SUPER
 UserInfo.MaxAccess = ESB_ACCESS_SUPER
 UserInfo.PwdChgNow = ESB_TRUE

 sts = EsbSetUser(hCtx, UserInfo)
 Debug.Print "EsbSetUser: sts = " & sts
End Sub

Sub ESB_GetUser()
 Dim sts As Long
 Dim User As String
 Dim UserInfo As ESB_USERINFO_T

 User = "Test"
 '************************
 ' Get User Info structure
 '************************
 sts = EsbGetUser(hCtx, User, UserInfo)
 Debug.Print "EsbGetUser: sts = " & sts
```

```
End Sub

Public Sub ESB_LROPurgeObjects()
 Dim UserName As String * ESB_USERNAMELEN
 Dim purgeDate As Long
 Dim Items As Integer
 Dim Desc As ESB_LRODESC_API_T
 Dim CutOffDate As Date
 Dim i As Integer
 Const ESB_REFERENCE_DATE = #1/1/1970#

 UserName = "essexer"
 CutOffDate = #9/21/2007#
 purgeDate = DateDiff("s", ESB_REFERENCE_DATE, CutOffDate) 'bug 8-651484045
 'purgeDate = DateDiff("s", CutOffDate, ESB_REFERENCE_DATE)
 'purgeDate = -1

 sts = EsbLROPurgeObjects(hCtx, UserName, purgeDate, Items)
 Debug.Print "EsbLROPurgeObjects: sts = " & sts

 If sts = 0 Then
  For i = 1 To Items
   '*******************************
   '* Get the next LRO description
   '* item from the list
   '*******************************
   sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE, Desc)
   Debug.Print "EsbGetNextItem: sts = " & sts
  Next i
 End If
End Sub

Sub ESB_CreateGroup()
 Dim sts As Long
 Dim GroupName As String

 GroupName = "PowerUsers"
 sts = EsbCreateGroup(hCtx, GroupName)
 Debug.Print "EsbCreateGroup: sts = " & sts
End Sub

Sub ESB_GetDatabaseInfo()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim Items As Integer
 Dim N As Integer
 Dim DbInfo As ESB_DBINFO_T
 Dim DbReqInfo As ESB_DBREQINFO_T

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbGetDatabaseInfo(hCtx, AppName, DbName, DbInfo, Items)
 Debug.Print "EsbGetDatabaseInfo: sts = " & sts
 Debug.Print "DbInfo.status: " & DbInfo.Status

 If sts = 0 Then
```

```
   For N = 1 To Items
    sts = EsbGetNextItem(hCtx, ESB_DBREQINFO_TYPE, DbReqInfo)
    Debug.Print "EsbGetNextItem: sts = " & sts
   Next
  End If
End Sub

Sub ESB_GetDatabaseAccess()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim User As String
 Dim UserDb As ESB_USERDB_T
 Dim sts As Long

 AppName = "Sample"
 DbName = "Basic"

 User = "user1"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next

 User = "user2"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next

 User = "user3"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next

 User = "user4"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next
```

```
 User = "user5"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next

 User = "user6"
 sts = EsbGetDatabaseAccess(hCtx, User, AppName, DbName, Items)
 Debug.Print "EsbGetDatabaseAccess: sts = " & sts
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERDB_TYPE, UserDb)
  Debug.Print "EsbGetNextItem: sts = " & sts
  Debug.Print "User: " & User
  Debug.Print "Access: " & UserDb.Access
 Next
End Sub

Sub ESB_GetDatabaseStats()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim DbStats As ESB_DBSTATS_T
 Dim DimStats As ESB_DIMSTATS_T
 Dim sts As Long
 AppName = "Sample"
 DbName = "Basic"
 sts = EsbGetDatabaseStats(hCtx, AppName, DbName, DbStats, Items)
 Debug.Print "EsbGetDatabaseStats: sts = " & sts
 'MsgBox ("cluster = " & DbStats.ClusterRatio)
 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_DBSTATS_TYPE, DbStats)
 Next
End Sub

Public Sub ESB_LROAddObject()
 Dim Desc As ESB_LRODESC_API_T
 Dim memCount As Long
 Dim memComb As String
 Dim opt As Integer
 Dim i As Integer

 memCount = 5
 memComb = "Year" & vbCrLf & "Product" & vbCrLf & _
  "Market" & vbCrLf & "Measures" & vbCrLf & "Scenario"
 Desc.UserName = "essexer"

 Desc.ObjType = ESB_LROTYPE_CELLNOTE_API
 Desc.note = "Cell note"
 opt = ESB_NOSTORE_OBJECT_API
 sts = EsbLROAddObject(hCtx, memCount, memComb, opt, Desc)
 Debug.Print "EsbLROAddObject: sts = " & sts
```

```
 Desc.ObjType = ESB_LROTYPE_WINAPP_API
 Desc.lroInfo.ObjName = "c:\hyperion\essbase95\bin\essbase.exe"
 Desc.lroInfo.objDesc = "Essbase executable."
 opt = ESB_STORE_OBJECT_API
 sts = EsbLROAddObject(hCtx, memCount, memComb, opt, Desc)
 Debug.Print "EsbLROAddObject: sts = " & sts

 Desc.ObjType = ESB_LROTYPE_URL_API
 Desc.lroInfo.ObjName = "www.oracle.com"
 Desc.lroInfo.objDesc = "Oracle homepage"
 opt = ESB_NOSTORE_OBJECT_API
 sts = EsbLROAddObject(hCtx, memCount, memComb, opt, Desc)
 Debug.Print "EsbLROAddObject: sts = " & sts

  Desc.ObjType = ESB_LROTYPE_CELLNOTE_API
  Desc.note = "Cell note 2"
  opt = ESB_NOSTORE_OBJECT_API
 sts = EsbLROAddObject(hCtx, memCount, memComb, opt, Desc)
  Debug.Print "EsbLROAddObject: sts = " & sts
End Sub

Public Sub ESB_LROGetCatalog()

 Dim Desc As ESB_LRODESC_API_T
 Dim Items As Integer
 Dim memCount As Long
 Dim memComb As String
 Dim i As Integer

 memCount = 5
 memComb = "Qtr1" & vbCrLf & "Profit" & vbCrLf & _
      "100" & vbCrLf & "East" & vbCrLf & "Scenario"
  'memComb = "Jan" & vbCrLf & "Sales" & _
  '      "Cola" & vbCrLf & "Utah" & _
  '      "Actual"

 sts = EsbLROGetCatalog(hCtx, memCount, memComb, Items)
 Debug.Print "EsbLROGetCatalog: sts = " & sts

 If sts = 0 Then
   For i = 1 To Items
    sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE, Desc)
    Debug.Print "Desc.ObjType = " & Desc.ObjType
    Debug.Print "Desc.note = " & Desc.note
    Debug.Print "Desc.lroInfo.objDesc = " & Desc.lroInfo.objDesc
    Debug.Print "Desc.lroInfo.objName = " & Desc.lroInfo.ObjName
   Next i
 End If
End Sub

Sub ESB_CopyObject()
 Dim sts As Long
 Dim SrcApp As String
 Dim SrcDb As String
 Dim SrcObj As String
 Dim DestApp As String
 Dim DestDb As String
```

```
 Dim DestObj As String

 SrcApp = "Sample"
 SrcDb = "Basic"
 SrcObj = "Basic"

 DestApp = "Sample"
 DestDb = "Basic"
 DestObj = "Basic1"
 ObjType = ESB_OBJTYPE_OUTLINE

 sts = EsbCopyObject(hCtx, hDestCtx, ObjType, SrcApp, DestApp, _
 SrcDb, DestDb, SrcObj, DestObj)
 Debug.Print "EsbCopyObject: sts = " & sts
End Sub

Sub ESB_GetAssociatedAttributesInfo()
 Dim sts As Long
 Dim MbrName As String
 Dim AttrDimName As String
 Dim Count As Long
 Dim Attribinfo As ESB_ATTRIBUTEINFO_T
 Dim index As Integer
 Dim tempstring As String

 'MbrName = InputBox("Base member name", "Base Member Name")
 'AttrDimName = InputBox("Attribute Dimension Name (Optional)", "Attribute Dimension
Name")

 MbrName = "em41666"
 AttrDimName = "Job Start Date"
 sts = EsbGetAssociatedAttributesInfo(hCtx, MbrName, AttrDimName, Count)
 Debug.Print "EsbGetAssociatedAttributesInfo: sts = " & sts

 Debug.Print "Associated Attr info for: " & MbrName

 For index = 1 To Count
  sts = EsbGetNextItem(hCtx, ESB_ATTRIBUTEINFO_TYPE, Attribinfo)
  'Debug.Print "Dim Name: " & Attribinfo.DimName
  Debug.Print "Attribute Dim Name: " & Attribinfo.DimName
  Debug.Print "Attribute Mbr Name: " & Attribinfo.MbrName

  ' NOTE: use of select case statement to discern (and act upon) type of attribute
returned
  Select Case VarType(Attribinfo.Attribute)
   Case vbDouble
    Debug.Print "Data Type   : Numeric(Double)"
    Debug.Print "Data Value  : " & Attribinfo.Attribute
    Debug.Print ""
   Case vbBoolean
    Debug.Print "Data Type   : Boolean"
    Debug.Print "Data Value  : " & Attribinfo.Attribute
    Debug.Print ""
   Case vbDate
    Debug.Print "Data Type   : Date"
    Debug.Print "Data Value  : " & Attribinfo.Attribute
    Debug.Print ""
```

```
    Case vbString
      Debug.Print "Data Type   : String"
      Debug.Print "Data Value  : " & Attribinfo.Attribute
      Debug.Print ""
    End Select
   Debug.Print ""
 Next index
End Sub

Sub ESB_ListConnections()
 Dim Items As Integer
 Dim UserInfo As ESB_USERINFO_T
 Dim sts As Long

 sts = EsbListConnections(hCtx, Items)
 Debug.Print "EsbListConnections: sts = " & sts

 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_USERINFO_TYPE, UserInfo)
  Debug.Print "EsbGetNextItem: sts = " & sts
 Next
End Sub

Sub ESB_ListRequests()
  Dim Items As Integer
  Dim ReqInfo As ESB_REQUESTINFO_T
  Dim sts As Long

  sts = EsbListRequests(hCtx, UserName, AppName, DbName, Items)
  Debug.Print "EsbListRequests: sts = " & sts

  For N = 1 To Items
   sts = EsbGetNextItem(hCtx, ESB_REQUESTINFO_TYPE, ReqInfo)
   Debug.Print "EsbGetNextItem: sts = " & sts
   Debug.Print "AppName: " & ReqInfo.AppName
   Debug.Print "DbName: " & ReqInfo.DbName
   Debug.Print "DbRequestCode: " & ReqInfo.DbRequestCode
   Debug.Print "LoginID: " & ReqInfo.LoginId
   Debug.Print "LoginSourceMachine: " & ReqInfo.LoginSourceMachine
   Debug.Print "RequestString: " & ReqInfo.RequestString
   Debug.Print "State: " & ReqInfo.State
   Debug.Print "TimeStarted: " & ReqInfo.TimeStarted
   Debug.Print "Username: " & ReqInfo.UserName
  Next
End Sub

Sub ESB_AddToGroup()
  Dim sts As Long
  Dim GroupName As String
  Dim User As String

  GroupName = "Group1"
  User = "user1"
  sts = EsbAddToGroup(hCtx, GroupName, User)
  Debug.Print "EsbAddToGroup sts: " & sts
End Sub
```

```
Sub ESB_GetGroupList()
 Dim Items As Integer
 Dim Group As String
 Dim GroupName As String * ESB_USERNAMELEN
 Dim sts As Long

 Group = "group1"
 sts = EsbGetGroupList(hCtx, Group, Items)
 Debug.Print "EsbGetGroupList: sts = " & sts

 For N = 1 To Items
  sts = EsbGetNextItem(hCtx, ESB_GROUPNAME_TYPE, ByVal GroupName)
   Debug.Print "EsbGetGroupList: sts = " & sts
   Debug.Print "User Name = " & GroupName
   MsgBox ("User Name = " & GroupName)
 Next
End Sub

Sub ESB_GetDatabaseState()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim DbState As ESB_DBSTATE_T
 AppName = "Sample"
 DbName = "Basic"

 sts = EsbGetDatabaseState(hCtx, AppName, DbName, DbState)
 Debug.Print "EsbGetDatabaseState: sts = " & sts
End Sub


Sub ESB_CreateLocalContext()
 Dim sts As Long
 Dim User As String
 Dim Password As String
 Dim hCtx As Long

 '*********************
 ' Create Local Context
 '*********************
 sts = EsbCreateLocalContext(hInst, User, Password, hCtx)
End Sub

Sub ESB_Import()
 Dim sts     As Long
 Dim Rules    As ESB_OBJDEF_T
 Dim Data     As ESB_OBJDEF_T
 Dim User     As ESB_MBRUSER_T
 Dim ErrorName  As String
 Dim AbortOnError As Integer
 Dim hLocalCtx As Long

 '************************************************************
 ' Need to create a local context, if files are not on the server
 '************************************************************
 sts = EsbCreateLocalContext(hInst, "", "", hLocalCtx)
 Debug.Print "EsbCreateLocalContext sts: " & sts
```

```
 Data.hCtx = hLocalCtx
 Data.Type = ESB_OBJTYPE_TEXT
 Data.AppName = ""
 Data.DbName = ""
 Data.FileName = "F:\\testArea\\VBAPI\\calcdat.txt"

 '*********************************
 ' Rules file resides at the server
 '*********************************
 'Rules.hCtx = hCtx
 'Rules.Type = ESB_OBJTYPE_RULES
 'Rules.AppName = "Demo"
 'Rules.DbName = "Basic"
 'Rules.FileName = "Test"

 '*********************************
 ' Data file resides at the server
 '*********************************
 'Data.hCtx = hCtx
 'Data.Type = ESB_OBJTYPE_TEXT
 'Data.AppName = "Demo"
 'Data.DbName = "Basic"
 'Data.FileName = "Data"

 '*********************************
 ' Specify file to redirect errors
 ' to if any
 '*********************************
 ErrorName = "IMPORT.ERR"

 '*************************
 ' Abort on the first error
 '*************************
 AbortOnError = ESB_YES

 '********
 ' Import
 '********
 sts = EsbImport(hCtx, Rules, Data, User, ErrorName, AbortOnError)
 Debug.Print "EsbImport sts: " & sts
End Sub

Sub ESB_VerifyFilter()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim Row As String

 AppName = "Sample"
 DbName = "Basic"

 sts = EsbVerifyFilter(hCtx, AppName, DbName)
 Debug.Print "EsbVerifyFilter sts: " & sts

 ' Initialize Filter Row
 Row = "@IDESCENDANTS(Scenario)"
 sts = EsbVerifyFilterRow(hCtx, Row) ' Initialize Filter Row
```

```
 Debug.Print "EsbVerifyFilterRow sts: " & sts

 Row = "@IDESCENDANTS(AAAA)"
sts = EsbVerifyFilterRow(hCtx, Row)
 Debug.Print "EsbVerifyFilterRow sts: " & sts

sts = EsbVerifyFilterRow(hCtx, ByVal 0&)
 Debug.Print "EsbVerifyFilterRow sts: " & sts
End Sub

Sub Test()
  strComputer = "."
  Const ForReading = 1
  Const ForWriting = 2
  Const ForAppending = 8
  '================================================================
  Const Data_Path = "F:\Testarea\temp\"
  Const FileName = "process.txt"

  Set fso = CreateObject("Scripting.FileSystemObject")
  If Not fso.FileExists(Data_Path & FileName) Then
   Set f = fso.OpenTextFile(Data_Path & FileName, 2, True)
  Else
   Set f = fso.OpenTextFile(Data_Path & FileName, 8)
  End If

  Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" &
strComputer & "\root\cimv2")
  Set colProcessList = objWMIService.ExecQuery("Select * from Win32_Process")
  For Each objProcess In colProcessList
    f.WriteLine "Process " & objProcess.Name
  Next
End Sub

Sub ESB_CreateGLDrillThru()
 Dim sts     As Long
 Dim url     As ESB_DURLINFO_T
 Dim cppDrillRegions(0 To 1) As String

 '************************************************************
 ' Need to create a local context, if files are not on the server
 '************************************************************
 url.bIsLevel0 = 0

 cppDrillRegions(0) = "sales"
 cppDrillRegions(1) = "cogs"
 url.cpURLXML = "<?xml version="1.0" encoding="UTF-8"?>
<foldercontents path="/">
 <resource name="Assets Drill through GL" description="" type="application/x-hyperion-
applicationbuilder-report">
  <name xml:lang="fr">Rapport de ventes</name>
  <name xml:lang="es">Informe de ventas</name>
  <action name="Display HTML" description="Launch HTML display of Content"
shortdesc="HTML">
   <url>/fusionapp/Assetsdrill.jsp?$SSO_TOKEN$&$CONTEXT$&$ATTR(ds,pos,gen,level.edge)
$
   </url>
```

```
  </action>
 </resource>
</foldercontents>"
 url.cpURLName = "VB URL7"
 url.iURLXMLSize = 512

 sts = EsbCreateDrillThruURL(hCtx, cppDrillRegions, url)

 Debug.Print "EsbCreateDrillThruURL sts: " & sts

End Sub

Sub ESB_UpdateGLDrillThru()
 Dim sts           As Long
 Dim url           As ESB_DURLINFO_T
 Dim cppDrillRegions(0 To 1)  As String
 Dim bMerge          As Integer

 '************************************************************
 ' Need to create a local context, if files are not on the server
 '************************************************************
 url.bIsLevel0 = 0
 bMerge = ESB_TRUE

 cppDrillRegions(0) = "qtr1"
 url.cpURLXML = "<?xml version="1.0" encoding="UTF-8"?>
<foldercontents path="/">
 <resource name="Assets Drill through GL" description="" type="application/x-hyperion-
applicationbuilder-report">
  <name xml:lang="fr">Rapport de ventes</name>
  <name xml:lang="es">Informe de ventas</name>
  <action name="Display HTML" description="Launch HTML display of Content"
shortdesc="HTML">
   <url>/fusionapp/Assetsdrill.jsp?$SSO_TOKEN$&$CONTEXT$&$ATTR(ds,pos,gen,level.edge)
$
   </url>
  </action>
 </resource>
</foldercontents>"
 url.cpURLName = "VB URL7"
 url.iURLXMLSize = 512

 sts = EsbUpdateDrillThruURL(hCtx, cppDrillRegions, url, bMerge)

 Debug.Print "EsbUpdateDrillThruURL sts: " & sts
End Sub

Sub ESB_DeleteGLDrillThru()
 Dim URLName    As String

 URLName = "VB URL7"
 sts = EsbDeleteDrillThruURL(hCtx, URLName)

 Debug.Print "EsbDeleteDrillThruURL sts: " & sts
End Sub

Sub ESB_GetGLDrillThru()
```

```
 Dim URLName   As String
 Dim url     As ESB_DURLINFO_T
 Dim intX     As Integer
 Dim cppDrillRegions  As Variant

 URLName = "VB URL2"
 sts = EsbGetDrillThruURL(hCtx, URLName, url, cppDrillRegions)

 Debug.Print "EsbGetDrillThruURL sts: " & sts

 If sts = 0 Then
  Debug.Print "URL Name: " & url.cpURLName
  Debug.Print "URL XML: " & url.cpURLXML

  For intX = LBound(cppDrillRegions) To UBound(cppDrillRegions)

  Debug.Print "URL Region: " & cppDrillRegions(intX)

  Next
 End If
End Sub

Sub ESB_ListGLDrillThru()
 Dim intX     As Integer
 Dim URLNames  As Variant

 sts = EsbListDrillThruURLs(hCtx, URLNames)

 If sts = 0 Then
   Debug.Print "EsbListDrillThruURL sts: " & sts

   For intX = LBound(URLNames) To UBound(URLNames)

   Debug.Print "URL Name: " & URLNames(intX)

   Next
 End If
End Sub

Sub ESB_GetCellDrillThruReports()
 Dim intX     As Integer
 Dim mbrs(0 To 4) As String
 Dim pURLXMLLens  As Variant
 Dim pURLXMLs  As Variant

 mbrs(0) = "sales"
 mbrs(1) = "jan"
 mbrs(2) = "New York"
 mbrs(3) = "actual"
 mbrs(4) = "100-10"

 sts = EsbGetCellDrillThruReports(hCtx, mbrs, pURLXMLLens, pURLXMLs)

 If sts = 0 Then

   Debug.Print "EsbGetCellDrillThruReports sts: " & sts
```

```
    For intX = LBound(pURLXMLLens) To UBound(pURLXMLLens)

    Debug.Print "URL XML: " & intX
    Debug.Print "URL XML Len: " & pURLXMLLens(intX)
    Debug.Print "URL XML String: " & pURLXMLs(intX)

    Next
  End If

  mbrs(0) = "profit"
  sts = EsbGetCellDrillThruReports(hCtx, mbrs, pURLXMLLens, pURLXMLs)

  If sts = 0 Then

    Debug.Print "EsbGetCellDrillThruReports sts: " & sts

    For intX = LBound(pURLXMLLens) To UBound(pURLXMLLens)

    Debug.Print "URL XML: " & intX
    Debug.Print "URL XML Len: " & pURLXMLLens(intX)
    Debug.Print "URL XML String: " & pURLXMLs(intX)

    Next
  End If
End Sub

Sub Main()
  'Test
  ESB_Init
  'ESB_CreateLocalContext
  'ESB_AutoLogin
  ESB_Login
  'ESB_LoginSetPassword
  ESB_SetActive
  ESB_CreateGLDrillThru
  ESB_UpdateGLDrillThru
  ESB_GetGLDrillThru
  ESB_ListGLDrillThru
  ESB_GetCellDrillThruReports
  ESB_DeleteGLDrillThru
  'ESB_GetGLDrillThru
  'ESB_ListGLDrillThru
  ESB_GetCellDrillThruReports
  'ESB_SetUser
  'ESB_GetUser
  'ESB_GetMessage
  'ESB_Import
  'ESB_GetVersion
  'ESB_GetDatabaseInfo
  'ESB_GetDatabaseState
  'ESB_GetDatabaseStats
  'ESB_GetDatabaseAccess
  'ESB_GetGroupList
  'ESB_ListConnections
  'ESB_ListRequests
  'ESB_GetAssociatedAttributesInfo
  'ESb_GetStoresInfo
```

```
  'ESB_OtlGetMemberAlias
  'ESB_AddAliasCombination
  'ESB_CreateGroup
  'ESB_LROAddObject
  'ESB_LROGetCatalog
  'ESB_LROListObjects
  'ESB_LROPurgeObjects

  'ESB_CopyObject
  'ESB_PartitionReadDefFile
  'ESB_PartitionWriteDefFile
  'ESB_PartitionReplaceDefFile
  'ESB_PartitionValidateDefinition
  'ESB_PartitionValidateLocal
  'ESB_PartitionReadOtlChangeFile

  'ESB_AddToGroup
  'ESB_GetGroupList
  'ESB_VerifyFilter
  ESB_Logout
  ESB_Term
End Sub
```

19 Visual BasicのメインAPIの宣言

この章の内容

定数の定義

次の定数は、Essbase Visual Basic グローバル・テキスト・ファイル ESB32.BAS と C 言語ヘッダー・ファイル ESBAPI.H で定義されます

- 1253 ページの「最大文字列長」
- 1254 ページの「情報フラグ定数」
- 1254 ページの「サイズ・フラグ定数」
- 1254 ページの「次元タグ定数」

最大文字列長

次の定数は、Essbase VB API での様々な文字列型の最大長を定義します。これらの定数は、VB アプリケーションでの変数の宣言に使用されます:

定数	定義
ESB_USERNAMELEN	ユーザーまたはグループ名の最大長
ESB_PASSWORDLEN	ユーザー・パスワードの最大長
ESB_SVRNAMELEN	サーバー名の最大長
ESB_APPNAMELEN	アプリケーション名の最大長
ESB_DBNAMELEN	データベース名の最大長
ESB_OBJNAMELEN	オブジェクト名の最大長
ESB_MBRNAMELEN	メンバー名の最大長

定数	定義
ESB_FTRNAMELEN	フィルタ名の最大長
ESB_ALIASNAMELEN	別名テーブル名の最大長
ESB_PATHLEN	ファイル・パス名の最大長
ESB_LINELEN	レポートの 1 行の最大長
ESB_DESCLEN	アプリケーションまたはデータベースの説明の最大長

情報フラグ定数

次の定数は、1273 ページの「ESB_DBREQINFO_T」構造体の DbReqFlags(データのロード)フィールドで使用する利用可能な情報フラグを定義します。

定数	定義
ESB_DBREQFLAG_CALCDEF	DbReqFlags フィールドのデフォルト・フラグ。デフォルトの計算スクリプトで使用されます。値: 1。
ESB_DBREQFLAG_CALCDSCR	DbReqFlags フィールドのカスタム計算スクリプト・フラグ。カスタム計算スクリプトで使用されます。値: 2。

サイズ・フラグ定数

次の定数は、ESB_DBSTATE_T 構造体における MaxMemIndex および1274 ページの「ESB_DBSTATE_T」フィールドの最大サイズと最小サイズを定義します。

定数	定義
ESB_INDEXCACHEMIN_SIZE	ESB_DBSTATE_T 構造体の MaxMemIndex フィールドの最小インデックス・キャッシュ・サイズ。値: 1048576。最大値は定義されていません。
ESB_INDEXPAGEMAX_SIZE	ESB_DBSTATE_T 構造体の IndexPageSize フィールドの最大インデックス・ページ・サイズ。値: 8192
ESB_INDEXPAGEMIN_SIZE	ESB_DBSTATE_T 構造体の IndexPageSizeMin フィールドの最小インデックス・ページ・サイズ。値: 1024

次元タグ定数

次の定数は、1279 ページの「ESB_DIMENSIONINFO_T」構造体の DimTag フィールドで使用する利用可能な情報フラグを定義します。

定数	定義
ESB_TTYPE_NONE	次元タイプなし。ESB_DIMENSIONINFO_T の DimTag フィールドの値。
ESB_TTYPE_CATEGORY	勘定科目: 通貨 ACCOUNTS タグ。ESB_DIMENSIONINFO_T の DimTag タイプの値。

定数	定義
ESB_TTYPE_CNAME	国: 通貨 COUNTRY タグ。ESB_DIMENSIONINFO_T の DimTag フィールドの値。
ESB_TTYPE_CTIME	時刻: 通貨 TIME タグ。ESB_DIMENSIONINFO_T の DimTag フィールドの値。
ESB_TTYPE_TYPE	タイプ: 通貨 TYPE タグ。ESB_DIMENSIONINFO_T の DimTag フィールドの値。
ESB_TTYPE_PARTITION	通貨 PARTITION タグ。ESB_DIMENSIONINFO_T の DimTag フィールドの値。

リンク・オブジェクトに対する定数および構造体の定義

次の定数および構造体は、特に LRO で使用するために定義されています:

LRO の定数

次の定数は、LRO 関数と Essbase Visual Basic API の構造体で使用される各種の値を定義します。

定数	値	定義
ESB_LRODESCLEN_API	79	オブジェクトの記述の最大長
ESB_LRONOTELEN_API	599	セル・ノートの最大長
ESB_ONAMELEN_API	511	ファイル名とパスからなるオブジェクト名の長さ
ESB_STORE_OBJECT_API	&H0010	サーバー上にリンク・オブジェクトを保管するように指定する値
ESB_NOSTORE_OBJECT_API	&H0001	サーバー上にリンク・オブジェクトを保管しないように指定する値
ESB_LRO_OBJ_API	1	リンク・オブジェクト・ファイルのみを更新するように指定する値
ESB_LRO_CATALOG_API	2	オブジェクトのカタログ・エントリのみを更新するように指定する値
ESB_LRO_BOTH_API	3	オブジェクト・ファイルとカタログ・エントリの両方を更新するように指定する値
ESB_LROTYPE_CELLNOTE_API	0	リンク・オブジェクトがセル・ノートであることを指定する値
ESB_LROTYPE_WINAPP_API	1	リンク・オブジェクトが Windows アプリケーションであることを指定する値

定数	値	定義
ESB_LROTYPE_URL_API	2	リンク・オブジェクトが URL であることを指定する値

ESB_CELLADDR_API_T

この構造体はデータ・セルのアドレスに関する情報を含んでいます。Essbase はメンバーの組合せからセル・アドレスを派生させ、アドレスを使用してデータ・セルにリンクされたオブジェクトを追跡します。`EsbLROAddObject` 関数はオブジェクトの説明構造体でセル・アドレスを戻します。この情報を後続の API 呼出しで使用できます。この構造体のフィールドは API では変更できません。次にフィールドについて説明します:

データ型	フィールド	説明
Long	cellOffset	データ・ブロックにおけるセルのオフセット
DOUBLE	blkOffset	ブロック・オフセット
DOUBLE	segment	セグメント番号

ESB_LRODESC_API_T

この構造体は、Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを説明する情報を含んでいます。次にフィールドについて説明します:

データ型	フィールド	説明
Integer	ObjType	オブジェクト・タイプ
Integer	status	カタログ・エントリのステータス
Long	memCount	データ・セルを識別する、メンバーの組合せにおけるメンバー名の数
ESB_LROHANDLE_API_T	LinkID	オブジェクトの識別構造体へのリンク
Long	updateDate	オブジェクトが最後に変更された日付
Integer	accessLevel	リンク・オブジェクトに関連付けられたデータ・セルに対するアクセス・レベル
String * ESB_ USERNAMELEN	userName	オブジェクトを最後に変更したユーザー名
String	memComb	リンク・オブジェクトと関連付けられたデータ・セルを識別するメンバーの組合せ
String * ESB_ LRONOTELEN_API	note	ユニオンによって関連付けられたセル・ノート
ESB_LROINFO_API_T	lroInfo	ユニオンによって関連付けられた LRO 情報構造体

ESB_LROHANDLE_API_T

この構造体は、リンク・オブジェクトの識別子を提供します。識別子はセル・アドレスと内部オブジェクト・ハンドルから構成されます。この構造体のフィールドは変更してはいけません。次にフィールドについて説明します:

データ型	フィールド	説明
ESB_CELLADDR_API_T	cellKey	セル・アドレス
Long	hObject	内部オブジェクト・ハンドル

ESB_LROINFO_API_T

この構造体は、Essbase データベースのデータ・セルにリンクされた特定のオブジェクトに関する説明的な情報を含みます。この構造体を、オブジェクトのソース・ファイル名またはオブジェクト記述の更新時に変更できます。これを行うには、EsbLROGetCatalog を使用してオブジェクトのカタログ・エントリを取得し、必要に応じて objName および objDesc フィールドを変更し、EsbLROUpdateObject を使用して変更をサーバーに保存します。次にフィールドについて説明します:

データ型	フィールド	説明
String * ESB_ONAMELEN_API	objName	データ・セルにリンクされたオブジェクトのソース・ファイル名。ESB_ONAMELEN_API はオブジェクト名の最大長を指定します。デフォルト値は 511 です。
String * ESB_LRODESCLEN_API	objDesc	データ・セルにリンクされたオブジェクトの説明。ESB_LRODESCLEN_API は説明の最大長を指定します。デフォルト値は 79 です。

パーティションの定数および構造体の定義

1288 ページの「ESB_PART_CONNECT_INFO_T」

1288 ページの「ESB_PART_DEFINED_T」

1289 ページの「ESB_PART_INFO_T」

1290 ページの「ESB_PART_REPL_T」

1291 ページの「ESB_PARTOTL_QRY_FILTER_T」 1291 ページの「ESB_PARTOTL_QUERY_T」

1292 ページの「ESB_PARTSLCT_T」

標準 C 言語の型

次の表は、C 言語プログラムで使用される ESBAPI.H に定義されたデータ型を記載しています:

- 表 9
- 表 10
- 表 11
- 表 12
- 表 13

表 9　単純なデータ型

データ型	Essbase 型
typedef char	ESB_CHAR_T
typedef short	ESB_SHORT_T
typedef long	ESB_LONG_T
typedef unsigned char	ESB_UCHAR_T
typedef unsigned short	ESB_USHORT_T
typedef unsigned long	ESB_ULONG_T
typedef float	ESB_FLOAT_T
typedef double	ESB_DOUBLE_T

表 10　その他のデータ型

データ型	Essbase 型	説明
typedef unsigned char	ESB_BOOL_T	boolean
typedef char	*ESB_STR_T	文字列(文字の配列)
typedef void	*ESB_HINST_T	API インスタンス・ハンドル
typedef void	*ESB_HCTX_T	API コンテキスト・ハンドル
typedef void	ESB_VOID_T	void
typedef size_t	ESB_SIZE_T	メモリー・ブロックのサイズ
typedef unsigned short	ESB_ACCESS_T	Essbase のアクセス・レベル
typedef unsigned long	ESB_LOGINID_T	Essbase ログイン ID

表 11　ポインタ型

データ型	Essbase 型	説明
typedef char	*ESB_PCHAR_T	char 型に対するポインタ
typedef short	*ESB_PSHORT_T	short 型に対するポインタ
typedef long	*ESB_PLONG_T	long 型に対するポインタ

データ型	Essbase 型	説明
typedef unsigned char	*ESB_PUCHAR_T	符合なし char 型に対するポインタ
typedef unsigned short	*ESB_PUSHORT_T	符号なし short 型に対するポインタ
typedef unsigned long	*ESB_PULONG_T	符号なし long 型に対するポインタ
typedef float	*ESB_PFLOAT_T	float 型に対するポインタ
typedef double	*ESB_PDOUBLE_T	double 型に対するポインタ
typedef ESB_BOOL_T	*ESB_PBOOL_T	boolean 型に対するポインタ
typedef ESB_STR_T	*ESB_PSTR_T	文字列へのポインタ
typedef ESB_VOID_T	*ESB_PVOID_T	void 型に対するポインタ
typedef ESB_SIZE_T	*ESB_PSIZE_T	メモリー・ブロックのサイズへのポインタ
typedef ESB_HINST_T	*ESB_PHINST_T	VB API インスタンス・ハンドルへのポインタ
typedef ESB_HCTX_T	*ESB_PHCTX_T	VB API コンテキスト・ハンドルへのポインタ
typedef ESB_ACCESS_T	*ESB_PACCESS_T	セキュリティ・アクセス・レベルへのポインタ
typedef ESB_LOGINID_T	*ESB_PLOGINID_T	ログイン ID へのポインタ

表 12　その他の型

データ型	Essbase 型	説明
typedef long	ESB_STS_T	API 関数からの戻り値
typedef ESB_STS_T	(*ESB_FUNC_T)()	関数へのポインタ

表 13　API 定義

定数	値
#define ESB_TRUE	1
#define ESB_FALSE	0
#define ESB_NULL	NULL

標準的な Visual Basic 言語の型

次の表は、VB API 関数が **ESB_xx...x_T** を参照する場合は常に(ユーザー定義の型を除く)Visual Basic アプリケーションで使用される C データ型について説明しています。Visual Basic では、これらのデータ型に基づいた新規のデータ型の定義を許可しません。

- 表 16
- 表 17
- 表 18
- 表 19

表 14　単純なデータ型

データ型	Essbase 型
As String * 1	ESB_CHAR_T
As Integer	ESB_SHORT_T
As Long	ESB_LONG_T
As String * 1	ESB_UCHAR_T
As Integer	ESB_USHORT_T
As Long	ESB_ULONG_T
As Long	ESB_FLOAT_T
As Long	ESB_DOUBLE_T
As Long	ESB_TIME_T
As Long	ESB_DATE_T

表 15 ビットマスク・データ型

データ型	Essbase 型	説明
As Integer	ESB_ ACCESS_T	セキュリティ・アクセス・レベル。可能なビット値は次のとおりです: ● ESB_PRIV_NONE - 0x0000 - 権限なし ● ESB_PRIV_READ - 0x0001 - データの読取り ● ESB_PRIV_WRITE - 0x0002 - データの書込み ● ESB_PRIV_CALC - 0x0004 - データの計算 ● ESB_PRIV_DBLOAD - 0x0010 - データベースのロードおよびアンロード ● ESB_PRIV_DBDESIGN - 0x0020 - データベースのデザイン ● ESB_PRIV_DBCREATE - 0x0040 - データベースの作成、削除および編集 ● ESB_PRIV_APPLOAD - 0x0100 - アプリケーションのロードおよびアンロード ● ESB_PRIV_APPDESIGN - 0x0200 - アプリケーションのデザイン ● ESB_PRIV_APPCREATE - 0x0400 - アプリケーションの作成、削除および編集 ● ESB_PRIV_USERCREATE - 0x1000 - ユーザーの作成、削除および編集 アクセス・タイプは権限の組合せです。有効な値は次のとおりです: ● ESB_ACCESS_NONE - 0x0000 ● ESB_ACCESS_READ - 0x0111 ● ESB_ACCESS_WRITE - 0x0113 ● ESB_ACCESS_CALC - 0x0117 ● ESB_ACCESS_DBDESIGN - 0x0137 ● ESB_ACCESS_DBCREATE - 0x0177 ● ESB_ACCESS_APPDESIGN - 0x0377 ● ESB_ACCESS_APPCREATE - 0x0777 ● ESB_ACCESS_FILTER - 0x0110 ● ESB_ACCESS_DBALL - 0x00ff - データベースへのフル・アクセス ● ESB_ACCESS_APPALL - 0x0fff - アプリケーションおよびデータベースへのフル・アクセス ● ESB_ACCESS_SUPER - 0xffff - 管理者(無制限アクセス)

データ型	Essbase 型	説明
As Integer	ESB_ OBJTYPE_T	ファイル・オブジェクト・タイプ。単一オブジェクト・タイプは次のとおりです: ESB_OBJTYPE_NONE ESB_OBJTYPE_OUTLINE ESB_OBJTYPE_CALCSCRIPT ESB_OBJTYPE_REPORT ESB_OBJTYPE_RULES ESB_OBJTYPE_ALIAS ESB_OBJTYPE_STRUCTURE ESB_OBJTYPE_ASCBACKUP ESB_OBJTYPE_BINBACKUP ESB_OBJTYPE_EXCEL ESB_OBJTYPE_LOTUS2 (No longer supported) ESB_OBJTYPE_LOTUS3 (No longer supported) ESB_OBJTYPE_TEXT ESB_OBJTYPE_PARTITION ESB_OBJTYPE_LOTUS4 (No longer supported) ESB_OBJTYPE_WIZARD ESB_OBJTYPE_SELECTION ESB_OBJTYPE_LRO オブジェクト・タイプの組合せは次のとおりです: ESB_OBJTYPE_BACKUP ESB_OBJTYPE_WORKSHEET ESB_OBJTYPE_DATA ESB_OBJTYPE_ALL

注： ビットマップ・データ型に対する値は、適切な場合、追加の値を提供するために組み合せられるビット値から構成されます。たとえば、データベースに対する WRITE アクセスを必要とする呼出し元は、READ および WRITE の特権を持っている必要があります。したがって、**ESB_ACCESS_WRITE** は、**ESB_PRIV_READ** および **ESB__PRIV_WRITE** のビット値と等しくなります。同様に、**ESB_OBJTYPE_BACKUP** は、**ESB_OBJTYPE_ASCBACKUP** と **ESB_OBJTYPE_BINBACKUP** との組合せです。

表 16　追加データ型

データ型	Essbase 型	説明
ByVal As String * 1	ESB_BOOL_T	boolean
ByVal As String	*ESB_STR_T	文字列(文字の配列)
ByVal As Long	*ESB_HINST_T	API インスタンス・ハンドル
ByVal As Long	*ESB_HCTX_T	API コンテキスト・ハンドル

データ型	Essbase 型	説明
As Any	ESB_VOID_T	void
ByVal As Long	ESB_SIZE_T	メモリー・ブロックのサイズ
ByVal As Integer	ESB_ACCESS_T	Essbase のアクセス・レベル
ByVal As Long	ESB_LOGINID_T	Essbase ログイン ID

表 17　ポインタ型

データ型	Essbase 型	説明
As Integer	*ESB_PSHORT_T	short 型に対するポインタ
As Long	*ESB_PLONG_T	long 型に対するポインタ
As Integer	*ESB_PUSHORT_T	符号なし short 型に対するポインタ
As Long	*ESB_PULONG_T	符号なし long 型に対するポインタ
As Long	*ESB_PFLOAT_T	float 型に対するポインタ
As Long	*ESB_PDOUBLE_T	double 型に対するポインタ
As Any	*ESB_PVOID_T	void 型に対するポインタ
As Long	*ESB_PSIZE_T	メモリー・ブロックのサイズへのポインタ
As Long	*ESB_PHINST_T	VB API インスタンス・ハンドルへのポインタ
As Long	*ESB_PHCTX_T	VB API コンテキスト・ハンドルへのポインタ
As Integer	*ESB_PACCESS_T	セキュリティ・アクセス・レベルへのポインタ
As Long	*ESB_PLOGINID_T	Essbase ログイン ID へのポインタ

表 18　その他の型

データ型	Essbase 型	説明
ByVal As Long	ESB_STS_T	API 関数からの戻り値
As Long	ESB_FUNC_T	関数へのポインタ

表 19　ブール・フラグ

データ型	Essbase 型	説明
chr$(1)	ESB_TRUE	データ構造で使用されるブール TRUE
chr$(0)	ESB_FALSE	データ構造で使用されるブール FALSE
1	ESB_YES	VB API 関数パラメータのリストで使用される YES フラグ
0	ESB_NO	VB API 関数パラメータのリストで使用される NO フラグ

データ型	Essbase 型	説明
ByVal 0&	NULL	Null

Visual Basic API 属性の用語

表 20　VB API 属性の用語

用語	定義
バケットのタイプ	次元を作成する場合、ESB_ATTRMBRDT_DOUBLE 型のゼロレベルの属性メンバーをリレーショナル・ソースのデータ範囲に関連付けることができます。 バケットのタイプは、データ範囲の上限と下限を指定します。 1269 ページの「ESB_ATTRSPECS_T」を参照してください。
ESB_ATTRIBUTE_DIMENSION および ESB_ATTRIBUTE_MEMBER	ESB_ATTRIBUTE_DIMENSION は属性次元です。 ESB_ATTRIBUTE_MEMBER は属性次元メンバーです。 1610 ページの「ESB_ATTRIBUTEQUERY_T」を参照してください。 EsbCheckAttributes も参照してください。
ESB_ATTRIBUTED_MEMBER	ESB_ATTRIBUTED_MEMBER は、(基本次元の)メンバーの 1 つであり、そのメンバーに関連した属性メンバーが保管されています。 1610 ページの「ESB_ATTRIBUTEQUERY_T」を参照してください。 EsbCheckAttributes も参照してください。
ESB_BASE_DIMENSION および ESB_BASE_MEMBER	ESB_BASE_DIMENSION は、標準次元であり、標準次元に関連する属性次元が保管されています。 ESB_BASE_MEMBER は、基本次元メンバーです。 1610 ページの「ESB_ATTRIBUTEQUERY_T」を参照してください。 EsbCheckAttributes も参照してください。
ESB_STANDARD_DIMENSION および ESB_STANDARD_MEMBER	ESB_STANDARD_DIMENSION は、属性次元以外のすべての次元です。 ESB_STANDARD_MEMBER は、標準次元メンバーです。 1610 ページの「ESB_ATTRIBUTEQUERY_T」を参照してください。 EsbCheckAttributes も参照してください。

用語	定義
ロング名	ESB_ATTRMBRDT_STRING 型以外のゼロレベルの属性メンバーは、ロング名で一意に識別できます。 ESB_ATTRMBRDT_STRING 型のゼロレベルの属性メンバーは、それ自体が一意である必要があります。 次の構造体を参照してください: ● 1269 ページの「ESB_ATTRSPECS_T」 ● 1269 ページの「ESB_ATTRIBUTEINFO_T」 また、次の関数も参照してください: ● EsbGetAttributeSpecifications ● EsbOtlGetAttributeSpecifications ● EsbOtlSetAttributeSpecifications さらに、「属性メンバーの追加に関する注意」も参照してください。
ショート名	ESB_ATTRMBRDT_STRING 型以外のゼロレベルの属性メンバーを、ショート名と呼びます。 ショート名は、ESB_STR_T 型のパラメータとして関数に渡されます。 EsbOtlFindAttributeMembers を参照してください。

Visual Basic のメイン API 構造体

ESB_APPDB_T

一致するアプリケーション名とデータベース名を戻すために使用される、アプリケーション名とデータベース名の構造体。フィールドは次のとおりです:

```
   Type ESB_APPDB_T

  AppName    As String * ESB_APPNAMELEN
  DbName     As String * ESB_DBNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_APPNAMELEN	AppName	アプリケーション名
As String * ESB_DBNAMELEN	DbName	データベース名

ESB_APPINFO_T

これは特定のアプリケーションに関する情報を入手するために使用するアプリケーション情報構造体です。この構造体のフィールドは、VB API を使用して変更できません。変更可能な追加のアプリケーション状態パラメータを含んでいる1268

ページの「ESB_APPSTATE_T」構造体も参照してください。フィールドは次のとおりです:

```
  Type ESB_APPINFO_T

 Name       As String * ESB_APPNAMELEN
 Server      As String * ESB_SVRNAMELEN
 status      As Integer
 AppType     As Integer
 nConnects    As Integer
 nDbs       As Integer
 ElapsedAppTime  As Long
 storageType   As Integer
 AppLocale    As String * ESB_LOCALESTRING_LENGTH

End Type
```

VB データ型	フィールド	説明
As String * ESB_APPNAMELEN	Name	アプリケーション名
As String * ESB_SVRNAMELEN	Server	サーバー名
As Integer	Status	アプリケーション・ロード・ステータス。値は次のとおりです: ● ESB_STATUS_NOTLOADED ● ESB_STATUS_LOADING ● ESB_STATUS_LOADED ● ESB_STATUS_UNLOADING
As Integer	AppType	アプリケーションのタイプ。有効な値は次のとおりです: ● ESB_APP_UNICODE - 0x0003 - プログラムは Unicode クライアント・プログラムです。サーバーが Unicode モードでない場合、関数は失敗します。これはデフォルト値です。 ● ESB_APP_NONUNICODE - 0x0002 - プログラムは非 Unicode モード・クライアント・プログラムです。
As Integer	nConnects	アプリケーションに現在接続しているユーザー数
As Integer	nDbs	このアプリケーションにおけるデータベースの数
As Long	ElapsedAppTime	アプリケーションをロードしてから経過した秒数
As Integer	StorageType	ストレージ・タイプ。有効な値は次のとおりです: ● 0 - デフォルト ● 1 - 多次元 ● 2 - DB2 リレーショナル ● 3 - Oracle リレーショナル ● 4 - 集約ストレージ(ASO) ● 1000 - 未定義

VB データ型	フィールド	説明
As String	AppLocale	アプリケーションのロケール記述。データ型は ESB_LOCALESTRING_LENGTH です。

ESB_APPINFOEX_T

この構造体は **EsbGetApplicationInfoEx()**で使用されます。フィールドは次のとおりです:

```
   Type ESB_APPINFOEX_T

  Name       As String * ESB_APPNAMELEN
  Server      As String * ESB_SVRNAMELEN
  AppType     As Integer
  AppLocale    As String * ESB_LOCALESTRING_LENGTH
  storageType   As Integer
  status      As Integer
  nConnects    As Integer
  ElapsedAppTime  As Long
End Type
```

VB データ型	フィールド	説明
As String * ESB_APPNAMELEN	Name	アプリケーション名
As String * ESB_SVRNAMELEN	Server	ネットワーク・サーバー名
As Integer	AppType	アプリケーションのタイプ。有効な値は次のとおりです: ● ESB_APP_UNICODE - 0x0003 - プログラムは Unicode クライアント・プログラムです。サーバーが Unicode モードでない場合、関数は失敗します。これはデフォルト値です。 ● ESB_APP_NONUNICODE - 0x0002 - プログラムは非 Unicode モード・クライアント・プログラムです。
As String * ESB_LOCALESTRING_LENGTH	AppLocale	アプリケーションのロケール記述。データ型は ESB_LOCALESTRING_LENGTH です。
As Integer	StorageType	ストレージ・タイプ。有効な値は次のとおりです: ● 0 - デフォルト ● 1 - 多次元 ● 2 - DB2 リレーショナル ● 3 - Oracle リレーショナル ● 4 - 集約ストレージ(ASO) ● 1000 - 未定義
As Integer	Status	アプリケーションのロード・ステータス
As Integer	nConnects	接続しているユーザーの数

VB データ型	フィールド	説明
As Long	ElapsedApp Time	アプリケーションの経過時間: アプリケーションがロードされてからの秒数

ESB_APPSTATE_T

このアプリケーション状態構造体は、特定のアプリケーションの状態パラメータを取得および設定します。この構造体のすべてのフィールドは、VB API を使用して変更できます。ただし、フィールドによっては、集約ストレージ・データベースに適用されないものもあります。1265 ページの「ESB_APPINFO_T」を参照すると、変更できない追加のアプリケーション情報をご覧いただけます。フィールドは次のとおりです:

```
   Type ESB_APPSTATE_T

  Description  As String * ESB_DESCLEN
  Loadable    As Integer
  Autoload    As Integer
  Access     As Integer
  Connects    As Integer
  Commands    As Integer
  Updates     As Integer
  LockTimeout  As Long
  lroSizeLimit As Long
  Security    As Integer
End Type
```

VB データ型	フィールド	説明
As String * ESB_DESCLEN	Description	アプリケーションの説明(80 文字まで)
As String * 1	Loadable	アプリケーションがロード可能かどうかを示すフラグ(アプリケーションがロード可能な場合は ESB_TRUE)。
As String * 1	Autoload	Essbase サーバーの起動時に、アプリケーションが自動的にロードされるかどうかを示すフラグ(アプリケーションが自動的にロードされる場合は ESB_TRUE)。
As Integer	Access	アプリケーションにおけるデータベースへのデフォルトのアクセス(すべてのユーザーに対するアクセスの可能な最下位レベル)。値: ● ESB_PRIV_NONE ● ESB_PRIV_DBDESIGN ● ESB_PRIV_CALC ● ESB_PRIV_WRITE ● ESB_PRIV_READ
As String * 1	Connects	ユーザーがアプリケーションに接続できるかどうかを示すフラグ(ユーザーが接続できる場合は ESB_TRUE)。

VB データ型	フィールド	説明
As String * 1	Commands	ユーザーがアプリケーションにコマンドを発行できるかどうかを示すフラグ(アプリケーションがユーザー・コマンドを受け入れる場合は ESB_TRUE)。
As String * 1	Updates	ユーザーがアプリケーションのデータを更新できるかどうかを示すフラグ(アプリケーションがユーザーの更新コマンドを受け入れる場合は ESB_TRUE)。
As Long	LockTimeout	ブロックレベルのロックが自動的に解除されるまでのタイムアウト期間(秒)。このフィールドは集約ストレージ・データベースには適用されません。
As Long	lroSizeLimit	LRO ファイルのサイズに対する制限値。この制限値は、各アプリケーションに設定され、管理者またはプログラムは、大きすぎるリンク・ファイルからサーバーを保護できます。Essbase 自体は、サイズを制限せず、デフォルト値もありません。この制限値は、LRO URL (512 文字までに制限)または LRO セル・ノート(599 文字までに制限)に適用されません。このフィールドは集約ストレージ・データベースには適用されません。
As String * 1	Security	アプリケーション・セキュリティが使用可能であるかどうかを示すフラグ(セキュリティが使用可能な場合は ESB_TRUE)。

ESB_ATTRIBUTEINFO_T

この構造体は、属性に関する情報を含んでいます。

```
   Type ESB_ATTRIBUTEINFO_T
  MbrName  As String * ESB_MBRNAMELEN
  DimName  As String * ESB_MBRNAMELEN
  Attribute As Variant
End Type
```

VB データ型	フィールド	説明
As String * ESB_MBRNAMELEN	MbrName	ロング名も含めて、1284 ページの「ESB_MEMBERINFO_T」または1612 ページの「ESB_MBRINFO_T」で指定されている属性メンバー名
As String * ESB_MBRNAMELEN	DimName	属性次元名
As Variant	Attribute	属性値

ESB_ATTRSPECS_T

この構造体は、アウトラインの属性仕様の設定のために **EsbOtlSetAttributeSpecifications()**によって使用されます。また、アウトラインの属性仕様を取得するために **EsbOtlGetAttributeSpecifications()**と **EsbGetAttributeSpecifications()**でも使用されます。

```
   Type ESB_ATTRSPECS_T
 DefaultTrueString    As String * ESB_MBRNAMELEN
 DefaultFalseString   As String * ESB_MBRNAMELEN
 DefaultAttrCalcDimName As String * ESB_MBRNAMELEN
 DefaultSumMbrName    As String * ESB_MBRNAMELEN
```

```
 DefaultCountMbrName   As String * ESB_MBRNAMELEN
 DefaultAverageMbrName  As String * ESB_MBRNAMELEN
 DefaultMinMbrName    As String * ESB_MBRNAMELEN
 DefaultMaxMbrName    As String * ESB_MBRNAMELEN
 GenNameBy       As Integer
 UseNameOf       As Integer
 Delimiter       As Integer
 DateFormat       As Integer
 BucketingType     As Integer
End Type
```

VB データ型	フィールド	説明
As String * ESB_MBRNAMELEN	DefaultTrueString	TRUE を示すためブール属性と使用される文字列。デフォルト値は ESB_DEFAULT_TRUESTRING ("True")です。
As String * ESB_MBRNAMELEN	DefaultFalseString	FALSE を示すためブール属性と使用される文字列。デフォルト値は ESB_DEFAULT_FALSESTRING ("False")です。
As String * ESB_MBRNAMELEN	DefaultAttrCalcDimName	属性計算(集約)次元の名前。デフォルト値は ESB_DEFAULT_ATTRIBUTECALCULATIONS ("Attribute Calculations")です。
As String * ESB_MBRNAMELEN	DefaultSumMbrName	SUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESB_DEFAULT_SUM ("Sum")です。
As String * ESB_MBRNAMELEN	DefaultCountMbrName	COUNT を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESB_DEFAULT_COUNT ("Count")です。
As String * ESB_MBRNAMELEN	DefaultAverageMbrName	AVERAGE を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESB_DEFAULT_AVERAGE ("Average")です。
As String * ESB_MBRNAMELEN	DefaultMinMbrName	MINIMUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESB_DEFAULT_MIN ("Min")です。
As String * ESB_MBRNAMELEN	DefaultMaxMbrName	MAXIMUM を示すため属性計算(集約)次元と使用される名前。デフォルト値は ESB_DEFAULT_MAX ("Max")です。
As Integer	GenNameBy	ロング名を生成するときに接頭辞または接尾辞としてゼロレベルメンバーの世代を使用するかどうかを示す定数識別子: ● ESB_GENNAMEBY_PREFIX (デフォルト値) ● ESB_GENNAMEBY_SUFFIX
As Integer	UseNameOf	ロング名を生成するときに使用するゼロレベルメンバーの世代を示す定数識別子: ● ESB_USENAMEOF_NONE (デフォルト値) ● ESB_USENAMEOF_PARENT ● ESB_USENAMEOF_GRANDPARENTANDPARENT ● ESB_USENAMEOF_ALLANCESTORS ● ESB_USENAMEOF_DIMENSION

VB データ型	フィールド	説明
As Integer	Delimiter	ロング名を生成するときに使用する区切り記号を示す定数識別子: ● ESB_DELIMITER_UNDERSCORE (デフォルト値) ● ESB_DELIMITER_PIPE ● ESB_DELIMITER_CARET
As Integer	DateFormat	日時属性のフォーマットを示す定数識別子: ● ESB_DATEFORMAT_MMDDYYYY (デフォルト値) ● ESB_DATEFORMAT_DDMMYYYY
As Integer	BucketingType	数値属性のバケットのタイプを示す定数識別子: ● ESB_UPPERBOUNDINCLUSIVE (デフォルト値) ● ESB_UPPERBOUNDNONINCLUSIVE ● ESB_LOWERBOUNDINCLUSIVE ● ESB_LOWERBOUNDNONINCLUSIVE

ESB_DBFILEINFO_T

この構造体は、EsbListDbFiles によって取得されるインデックスまたはデータ・ファイルに関する情報を含んでいます。

```
  Type ESB_DBFILEINFO_T
 AppName     As String * ESB_APPNAMELEN
 DbName      As String * ESB_DBNAMELEN
 FilePath    As String * ESB_FILENAMELEN
 FileSize    As Long
 FileSequenceNum As Long
 FileCount    As Long
 FileType    As Integer
 FileOpen    As Integer
End Type
```

VB データ型	フィールド	説明
As String * ESB_APPNAMELEN	AppName	アプリケーション名
As String * ESB_DBNAMELEN	DbName	データベース名
As String * ESB_FILENAMELEN	FilePath	ファイル・パス
As Long	FileSize	ファイルのサイズ(単位はバイト)
As Long	FileSequenceNum	指定されたデータベースの FileType のファイルのセット内での 1 から始まるシーケンス番号
As Long	FileCount	戻された FileType のファイル数

VB データ型	フィールド	説明
As Integer	FileType	次のファイルのタイプのいずれかになります: ● ESB_FILETYPE_INDEX ● ESB_FILETYPE_DATA
As Integer	FileOpen	ファイルが開いているかどうかを示すフラグ: ファイルが閉じているときは 0、ファイルが開いているときはゼロ以外です

ESB_DBINFO_T

このデータベース情報構造体は、特定のデータベースに関する情報を取得します。この構造体のフィールドは、VB API を使用して変更できません。1274 ページの「ESB_DBSTATE_T」も参照してください。変更できる追加のデータベース状態パラメータを含んでいます。また、1277 ページの「ESB_DBSTATS_T」も参照してください。フィールドは次のとおりです:

```
   Type ESB_DBINFO_T

  ElapsedDbTime  As Long
  DataCacheSize  As Long
  IndexCacheSize As Long
  IndexPageSize  As Long
  nDims      As Long
  DbType     As Integer
  status     As Integer
  nConnects    As Integer
  nLocks     As Integer
  Data       As Integer
  AppName     As String * ESB_APPNAMELEN
  Name       As String * ESB_DBNAMELEN
  Country     As String * ESB_MBRNAMELEN
  Time       As String * ESB_MBRNAMELEN
  Category    As String * ESB_MBRNAMELEN
  Type       As String * ESB_MBRNAMELEN
  CrPartition   As String * ESB_MBRNAMELEN
End Type
```

VB データ型	フィールド	説明
As long	ElapsedDbTime	データベースがロードされている秒数
As long	DataCacheSize	現在データベースが使用しているランタイム・データ・キャッシュ・サイズ(KB 単位)。データ・キャッシュ・サイズを変更した後、新しいデータ・ファイル・キャッシュ・サイズを有効にするには、データベースを停止して再起動する必要があることに注意してください。
As long	IndexCacheSize	現在データベースで使用中のランタイム・インデックス・キャッシュ・サイズ(KB 単位)
As long	IndexPageSize	現在データベースで使用中のランタイム・インデックス・ページ・サイズ(KB 単位)

VB データ型	フィールド	説明
As Long	nDims	データベース中の次元数
As Integer	DbType	データベースのタイプ(標準または通貨)。このフィールドは次の値を含むことができます: ● ESB_DBTYPE_NORMAL ● ESB_DBTYTPE_CURRENCY
As Integer	Status	データベースのロード・ステータス(ロードされたかどうか) - 次のいずれかの値になります: ● ESB_STATUS_NOTLOADED ● ESB_STATUS_LOADING ● ESB_STATUS_LOADED ● ESB_STATUS_UNLOADING
As Integer	nConnects	現在データベースに接続しているユーザーの数
As Integer	nLocks	現在排他的にロックされているデータ・ブロックの数
As Integer	Data	データベースへのデータのロード状態を示すフラグ(データはロードされていない、データはロードされたが未計算、データはロードされて計算済のいずれか)。このフィールドは次の値の 1 つを含むことができます: ● ESB_DBDATA_NONE /* no data */ ● ESB_DBDATA_LOADNOCALC /*data loaded without calc */ ● ESB_DBDATA_CLEAN /* data has been calculated */
As String * ESB_APPNAMELEN	AppName	関連付けられたアプリケーション名
As String * ESB_DBNAMELEN	Name	データベース名
As String * ESB_MBRNAMELEN	Country	通貨国次元メンバー(ある場合)。ない場合、このフィールドは空の文字列です。
As String * ESB_MBRNAMELEN	Time	通貨時間次元のメンバー(ある場合)。ない場合、このフィールドは空の文字列です。
As String * ESB_MBRNAMELEN	Category	通貨カテゴリ次元メンバー(ある場合)。ない場合、このフィールドは空の文字列です。
As String * ESB_MBRNAMELEN	Type	通貨タイプ次元のメンバー(通貨データベースのみ)。ない場合は、このフィールドは空の文字列です。
As String * ESB_MBRNAMELEN	CrPartition	通貨パーティション・メンバー(非通貨データベースのみ)

ESB_DBREQINFO_T

この構造体は EssGetDatabaseInfo で使用されます。Essbase には情報が存在する要求のタイプとして、データのロード、計算、アウトラインの更新の 3 つがあります。次の Essbase API 定数は各タイプの要求を特定します:

- ESB_DBREQTYPE_DATLOAD 0 データのロード
- ESB_DBREQTYPE_CALC 1 計算
- ESB_DBREQTYPE_OTLUPD 2 アウトラインの更新

フィールドは次のとおりです:

```
   Type ESB_DBREQINFO_T

  DbReqType    As Long
  DbReqFlags   As Long
  StartTimeRec  As ESB_TIMERECORD_T
  EndTimeRec   As ESB_TIMERECORD_T
  User      As String * ESB_USERNAMELEN
End Type
```

データ型	フィールド	説明
As Long	DbReqType	データベース要求のタイプ
As Long	DbReqFlags	情報フラグのビット・マップ
1294 ページの「ESB_TIMERECORD_T」として	StartTimeRec	時間記録の開始を求める要求
1294 ページの「ESB_TIMERECORD_T」として	EndTimeRec	時間記録の終了を求める要求
As String * ESB_USERNAMELEN	User	ユーザー名

ESB_DBSTATE_T

このデータベースの状態構造体は、特定のデータベース用に状態パラメータを入手し設定します。この構造体内のフィールドはすべて、VB API を使用して変更できます。1272 ページの「ESB_DBINFO_T」および1277 ページの「ESB_DBSTATS_T」構造体も参照してください。これらは、変更できない追加のデータベース情報を含んでいます。

```
   Type ESB_DBSTATE_T

  Description     As String * ESB_DESCLEN
  Loadable       As Integer
  Autoload       As Integer
  Access        As Integer
  IndexType       As Integer
  MaxMem        As Long
  MaxCompMem      As Long
  MaxMemIndex      As Long
  IndexPageSize     As Long
  CalcNoAggMissing   As Integer
  CalcNoAvgMissing   As Integer
  CalcTwoPass      As Integer
  CalcCreateBlock    As Integer
  CrDbName       As String * ESB_DBNAMELEN
```

```
  CrTypeMember     As String * ESB_MBRNAMELEN
  CrConvType      As Integer
  DataCompress     As Integer
  RetrievalBuffer   As Long
  RetrievalSortBuffer As Long
  TimeOut       As Long
  CommitBlocks     As Long
  CommitRows      As Long
  nVolumes       As Long
  DataCompressType   As Integer
  IsolationLevel    As Integer
  PreImage       As Integer
End Type
```

フィールドは次のとおりです:

VB データ型	フィールド	説明
As String * ESB_DESCLEN	Description	データベースの記述(80 文字まで)
As Integer	Loadable	データベースがロード可能かどうかを示すフラグ(データベースがロード可能な場合は、ESB_TRUE)
As Integer	Autoload	アプリケーションの開始時にデータベースが自動的にロードされるかどうかを示すフラグ(データベースが自動的にロードされる場合は、ESB_TRUE)
As Integer	Access	データベースへのデフォルトのアクセス・レベル。このフィールドが含むことができる値のリストは、表 15 についての説明を参照してください。
As Integer	IndexType	データベースのインデックス・タイプ(配列またはツリー)。値: ● ESB_INDEXTYPE_ARRAY ● ESB_INDEXTYPE_AVL **注：** API リリース 4 以降については、IndexType フィールドは廃止されています。
As Long	MaxMem	データベースの圧縮されていないデータ・ブロック用の最大メモリー(バイト単位)
As Long	MaxCompMem	データベースの圧縮データ・ブロック用の最大メモリー(バイト単位)
As Long	MaxMemIndex	最小インデックス・キャッシュ・サイズ。値: 1048576。定数 ESB_INDEXCACHEMIN_SIZE を使用して設定します
As Long	IndexPageSize	バッファ・プールが作成されるインデックス・ページのサイズ(バイト単位)。 最小インデックス・ページ・サイズ。値: 1024。定数 ESB_INDEXPAGEMIN_SIZE を使用して設定します IndexPageSize フィールドの最大ページ・サイズ。値: 8192。定数 ESB_INDEXPAGEMAX_SIZE を使用して設定します
As Integer	CalcNoAgg Missing	メンバーの子がすべて欠落している場合に、メンバーを集約しないためのフラグ(欠落した値を集約しない場合は ESB_TRUE)

VB データ型	フィールド	説明
As Integer	CalcNoAvg Missing	平均値を計算する際に欠落しているメンバーを含めないようにするフラグ(欠落した値を含めない場合は ESB_TRUE)
As Integer	CalcTwoPass	データベースのフル計算を実行する際、2 パス計算を強制するためのフラグ(2 パス計算を使用可能にする場合は ESB_TRUE)
As Integer	CalcCreate Block	定数割当て計算式でデータ・ブロックの作成を強制するフラグ(疎次元にのみ有効)。ブロックを強制的に作成する場合、ESB_TRUE に設定します
As String * ESB_DBNAMELEN	CrDbName	関連付けられた通貨データベースの名前(非通貨データベースで有効)
As String * ESB_MBRNAMELEN	CrTypeMember	通貨換算タイプ・メンバーの名前(非通貨データベースで有効)
As Integer	CrConvType	通貨換算タイプ(通貨換算を乗算で計算するか、または除算で計算するか)。値: ● ESB_CRCTYPE_DIV ● ESB_CRCTYPE_MULT
As Integer	DataCompress	このデータベースのブロックを圧縮するかどうか指定するオプションのフラグ。
As Long	RetrievalBuffer	抽出された行のデータ・セルが RESTRICT、TOP または BOTTOM コマンドによって評価される前に、これらのセルを保管するサーバーのバッファのサイズを KB 単位で指定します。デフォルトは 2048 バイト。
As Long	RetrievalSortBuffer	取得の際にソートするデータを保管するサーバーのバッファのサイズを、KB 単位で指定します。デフォルトは 10240 バイトです。
As Long	TimeOut	タイムアウト間隔(秒単位)。これは、コミット・アクセスにのみ設定できます。 ● -1 は、待ち時間に制限がありません。 ● 0 は即時アクセスで待機なし(デフォルト)。 ● n は、秒単位で指定された間隔です。
As Long	CommitBlocks	明示コミットを実行する前に変更したデータ・ブロックの数(コミット設定が UNCOMMITTED の場合にのみ使用)。
As Long	CommitRows	明示コミットを実行する前にデータ・ロードする入力ファイルの行数(コミット設定が UNCOMMITTED の場合についてのみ使用)。
As Long	nVolumes	このデータベース用に設定されたディスク・ボリュームの数。
As Integer	DataCompressType	オプションの圧縮フラグが設定されている場合、書込み操作で使用されるデータ圧縮のタイプ。 ● Bitmap - データ・セルを示すのにビットマップを使用します(デフォルト値)。 ● Run-Length Encoding - 連続して繰り戻される値をすべて圧縮します。 ● No Compression - データを圧縮しません。

VB データ型	フィールド	説明
As Integer	IsolationLevel	コミット設定は次のとおりです: ● COMMITTED - トランザクションがコミットされるまで、影響を受けるすべてのデータ・ブロックに対して書込みロックをしてアクセスを制限します。 ● UNCOMMITTED - (デフォルト)トランザクションの実行中、必要に応じて書込みロックの取得や解放を行います。
As Integer	PreImage	読取りのみ要求の間に以前にコミットしたデータを読み取るためのフラグ。このフラグは、コミット・アクセスにのみ設定できます。デフォルトは YES です。

ESB_DBSTATS_T

このデータベース統計構造体は、特定のデータベースに関する実行時統計情報を取得します。この構造体のフィールドは、VB API を使用して変更できません。1274 ページの「ESB_DBSTATE_T」構造体も参照してください。変更できる追加のデータベース状態パラメータを含んでいます。また、1272 ページの「ESB_DBINFO_T」構造体も参照してください。フィールドは次のとおりです:

```
  Type ESB_DBSTATS_T

 nDims          As Long
 DeclaredBlockSize    As Long
 ActualBlockSize     As Long
 DeclaredMaxBlocks    As Double
 ActualMaxBlocks     As Double
 NonMissingLeafBlocks   As Double
 NonMissingNonLeafBlocks As Double
 NonMissingBlocks     As Double
 PagedOutBlocks      As Double
 PagedInBlocks       As Double
 InMemCompBlocks     As Double
 TotalBlocks        As Double
 NonExclusiveLockCount  As Double
 ExclusiveLockCount    As Double
 TotMemPagedInBlocks   As Double
 TotMemBlocks       As Double
 TotMemIndex        As Double
 TotMemInMemCompBlocks  As Double
 BlockDensity       As Double
 SparseDensity      As Double
 CompressionRatio     As Double
 IndexType         As Integer
 ClusterRatio;      As Double
End Type
```

VB データ型	フィールド	説明
As Integer	IndexType	データベース・インデックス・タイプ(配列またはツリー)。このフィールドには次の値が含められます: ● ESB_INDEXTYPE_ARRAY ● ESB_INDEXTYPE_AVL
As Long	nDims	次元数
As Long	DeclaredBlockSize	宣言されたデータ・ブロックのサイズ
As Long	ActualBlockSize	実際のデータ・ブロックのサイズ
As Double	DeclaredMax Blocks	宣言された、データベース内の最大ブロック数
As Double	ActualMaxBlocks	実際の、データベース内の最大ブロック数
As Double	NonMissingLeaf Blocks	データベース内の欠落していないリーフ(最下位レベル)ブロックの数
As Double	NonMissingNon LeafBlocks	データベース内の欠落していない非リーフ(上位レベル)ブロックの数
As Double	NonMissing Blocks	データベース内の欠落していないブロックの総数
As Double	PagedOutBlocks	現在ディスクにページ・アウトされているデータベース・ブロックの数
As Double	PagedInBlocks	現在メモリーにページ・インされているデータベース・ブロックの総数
As Double	TotalBlocks	既存のデータ・ブロックの総数(最大数でない)
As Double	NonExclusive LockCount	現在、非排他的にロックされているデータベース・ブロックの数
As Double	ExclusiveLock Count	現在、排他的にロックされているデータベース・ブロックの数
As Double	TotMemBlocks	すべてのデータベース・ブロックに使用されているメモリーの合計
As Double	TotMemIndex	データベース・インデックスに使用されているメモリーの合計
As Double	TotMemPagedIn Blocks	ページイン(非圧縮)されているすべてのデータベース・ブロックに使用されているメモリーの合計
As Double	BlockDensity	データベース・ブロックの平均密度(現在ロードされているすべてのブロックを使用して計算)
As Double	SparseDensity	データベース内の疎次元の平均密度
As Double	CompressionRatio	ディスク上のデータ・ブロックの平均圧縮率
As Double	InMemCompBlocks	現在、圧縮メモリーにページ・インされているデータベース・ブロックの数
As Double	TotMemInMemCompBlocks	現在、圧縮メモリーにページ・インされているデータベース・ブロックに使用されているメモリーの合計
As Double	ClusterRatio	ページ・ファイルの断片化のメジャー。1 に近い値は、断片化の程度が低いことを示します。0 に近い値は、計算およびクエリーのパフォーマンスに影響する可能性がある、高度の断片化を示します。

ESB_DIMENSIONINFO_T

この構造体は **EsbGetDimensionInfo()**で使用されます。フィールドは次のとおりです:

```
   Type ESB_DIMENSIONINFO_T
  DimName     As String * ESB_MBRNAMELEN
  DimNumber    As Long
  DimType     As Integer
  DimTag     As Integer
  DeclaredDimSize As Long
  ActualDimSize  As Long
  Description   As String * ESB_DESCLEN
  DimDataType   As Integer
End Type
```

データ型	フィールド	説明
As String * ESB_MBRNAMELEN	DimName	次元名
As Long	DimNumber	メンバーの次元数
As Integer	DimType	次元タイプ。値: ● ESB_DIMTYPE_DENSE ● ESB_DIMTYPE_SPARSE
As Integer	DimTag	次元タグ・タイプ。値: ● ESB_TTYPE_ATTRCALC ● ESB_TTYPE_ATTRIBUTE ● ESB_TTYPE_CATEGORY ● ESB_TTYPE_CNAME ● ESB_TTYPE_CTIME ● ESB_TTYPE_NONE ● ESB_TTYPE_PARTITION ● ESB_TTYPE_TYPE
As Long	DeclaredDimSize	宣言された次元のサイズ
As Long	ActualDim Size	実際の次元のサイズ
As String * ESB_DESCLEN	Description	予約済: 現在サポートされていません
As Integer	DimDataType	属性次元のデータ型。値: ● ESB_ATTRMBRDT_BOOL ● ESB_ATTRMBRDT_DATETIME ● ESB_ATTRMBRDT_DOUBLE ● ESB_ATTRMBRDT_STRING

ESB_DIMSTATS_T

この次元統計構造体は、特定のデータベース次元に関する情報を取得します。この構造体のフィールドは、VB API を使用して変更できません。これらの構造体の配列はデータベース統計構造体(EsbGetDbStats)でデータベース内の各次元の情報を提供するときに生成されます。フィールドは次のとおりです:

```
   Type ESB_DIMSTATS_T

 DeclaredDimSize  As Long
 ActualDimSize   As Long
 DimType      As Integer
 DimName      As String * ESB_MBRNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_ MBRNAMELEN	DimName	次元メンバー名
As Integer	DimType	次元タイプ(疎または密)。このフィールドには次の値が含められます: ● ESB_DIMTYPE_SPARSE ● ESB_DIMTYPE_DENSE
As Long	DeclaredDimSize	宣言された次元サイズ(指定した次元のラベルのみまたは共有メンバーも含めた、その次元で宣言されたメンバーの数)
As Long	ActualDim Size	実際の次元サイズ(指定した次元のラベルのみまたは共有メンバーを除いた、その次元で宣言されたメンバーの数)

ESB_DURLINFO_T

URL 情報の取得に使用されるデータ構造です。フィールドは次のとおりです:

```
   Type ESB_DURLINFO_T

 bIsLevel0      As Integer         'consider level-0 members along symmetric
regions
 iURLXMLSize      As Integer         'URL XML size
 cpURLName      As String * 1024       'URL identifier
 cpURLXML(0 To 8191)  As Byte          'URL XML
End Type
```

Visual Basic のデータ型	フィールド	説明
As Integer	bIsLevel0	1 の場合、URL 定義はレベル 0 のデータに制限されます。0 の場合、制限はありません
As Integer	iURLXMLSize	URL XML のサイズ

Visual Basic のデータ型	フィールド	説明
As String * 1024	cpURLName	URL 定義の名前
As Byte 0 to 8191	cpURLXML	URL XML のコンテンツ

注： 領域リストは、各 Visual Basic ドリルスルー関数内で個別の引数 symRegions() として渡されます。

ESB_GLOBAL_T

この構造体には管理目的で使用されるグローバル・サーバー・システム・パラメータが含まれています。この構造体のすべてのフィールドは、VB API を使用して変更できます。フィールドは次のとおりです:

```
   Type ESB_GLOBAL_T

  Security     As Integer
  Logins      As Integer
  Access      As Integer
  Validity     As Integer
  Currency     As Integer
  PwMin      As Integer
  InactivityTime  As Long
  InactivityCheck As Long
End Type
```

VB データ型	フィールド	説明
As String * 1	Security	グローバル・セキュリティが使用可能かどうかを示すフラグ(デフォルト値は ESB_TRUE で、セキュリティが有効です)
As String * 1	Logins	ユーザー・ログインが使用可能かどうかを示すフラグ(デフォルト値は ESB_TRUE で、ログインは使用可能)
As Integer	Access	新規作成されたアプリケーションのデフォルト・アクセス・レベル(デフォルトは ESB_ACCESS_NONE)。使用可能な値のリストについては、表 15 についての説明を参照してください。
As Integer	Validity	デフォルトのパスワード有効期間(デフォルト値は 365 日間)
As String * 1	Currency	通貨オプションがサポートされているかどうかを示すフラグ(このフラグは読取り専用です)。通貨オプションが使用可能な場合は ESB_TRUE に設定されます。
As Integer	PwMin	パスワードの最小の長さ(デフォルトは 6 文字)
As Long	InactivityTime	すべてのアプリケーションおよびエージェントから非アクティブなユーザーが自動的にログアウトされるまでの最大時間を秒数で表します。デフォルト値: 3600 秒。最小値: 300 秒。自動ログアウトを使用不可にするには、InactivityTime を 0 に設定します。

VB データ型	フィールド	説明
As Long	Inactivity Check	自動ログアウトの確認頻度を秒数で表します。デフォルト値: 300 秒。最小値: 30 秒。InactivityTime の設定より低くなければ、InactivityCheck は InactivityTime に設定され、警告メッセージが表示されます。自動ログアウトを使用不可にするには、InactivityCheck を 0 に設定します。

ESB_INIT_T

この構造体は VB API 初期化関数 **EsbInit()**に渡されます。この構造体には、API 開発者が API の使用方法をカスタマイズできるようにするフィールドが含まれています。構造体のいずれかのフィールドがゼロに設定されている場合、API デフォルトが使用されます。フィールドは次のとおりです:

```
   Type ESB_INIT_T

  Version       As Long
  MaxHandles      As Integer
  LocalPath      As String * ESB_PATHLEN
  MessageFile     As String * ESB_PATHLEN
  HelpFile       As String * ESB_PATHLEN
  ClientError     As Integer
  ErrorStack      As Integer
  vbCallbackFuncAddress  As Long
End Type
```

VB データ型	フィールド	説明
As Long	Version	アプリケーションのコンパイルに使用する Essbase API のバージョン。下位互換性のために使用されます。
As Integer	MaxHandles	アプリケーションで必要な同時コンテキスト・ハンドルの最大数(1-10 の範囲)
As String * ESB_PATHLEN	LocalPath	クライアント側でファイルとオブジェクトの操作に使用する、デフォルトのローカル・パス名
As String * ESB_PATHLEN	MessageFile	メッセージ・データベース・ファイル ESSBASE.MDB の絶対パス名
As String * ESB_PATHLEN	HelpFile	ユーザー定義のアプリケーション・ヘルプ・ファイルの完全修飾パス名で、「自動ログイン」ダイアログ・ボックスのヘルプに使用されます。ログイン・ヘルプ・コンテキストはヘルプ・ファイルで定義する必要があります。第 3 章「Essbase と使用中の製品との統合」を参照してください。 ESSBASEPATH が定義されていない場合、ヘルプ・ファイル名は NULL に設定されます。
As String * 1	ClientError	デフォルトのエラー・ハンドラを使用するには ESB_FALSE を、メッセージを取得するために EsbGetMessage を使用するには ESB_TRUE を指定します

VB データ型	フィールド	説明
As Integer	ErrorStack	EsbGetMessage で使用されるメッセージ・スタックのサイズ。デフォルト値は 100 です
As Long	vbCallbackFuncAddress	カスタム Visual Basic コールバック関数である AddressOf。詳細は、1221 ページの「Visual Basic の API メッセージの処理」についての説明を参照してください。

ESB_LOCKINFO_T

この構造体は、**EsbListLock()**関数から戻された排他的にロックされているデータ・ブロックについての情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_LOCKINFO_T

  LoginId  As Long
  Time    As Long
  nLocks   As Integer
  userName  As String * ESB_USERNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_USERNAMELEN	UserName	ユーザー名
As Integer	nLocks	このユーザーによって排他的にロックされているブロックの数
As Integer	Time	ブロックが排他的にロックされた最長時間(秒単位)
As Long	LoginId	ユーザー・ログイン識別タグ

ESB_MBRALT_T

この構造体は、代替メンバー名についての情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_MBRALT_T

  MbrName  As String * ESB_MBRNAMELEN
  AltName  As String * ESB_MBRNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_MBRNAMELEN	MbrName	メンバー名

VB データ型	フィールド	説明
As String * ESB_MBRNAMELEN	AltName	関連付けられた別名(ESB_MBRNAME_T)

ESB_MBRUSER_T

この構造体は、SQL ユーザー名およびパスワードについての情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_MBRUSER_T

 User     As String * ESB_USERNAMELEN
 Password   As String * ESB_PASSWORDLEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_USERNAMELEN	User	SQL データベース・ユーザー名
As String * ESB_PASSWORDLEN	Password	SQL データベース・ユーザー・パスワード

ESB_MEMBERINFO_T

この構造体は、特定のメンバーに関する情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_MEMBERINFO_T
 CrMbrName   As String * ESB_MBRNAMELEN
 MbrName    As String * ESB_MBRNAMELEN
 DimName    As String * ESB_MBRNAMELEN
 ParentMbrName As String * ESB_MBRNAMELEN
 ChildMbrName  As String * ESB_MBRNAMELEN
 PrevMbrName  As String * ESB_MBRNAMELEN
 NextMbrName  As String * ESB_MBRNAMELEN
 Description  As String * ESB_DESCLEN
 MbrNumber   As Long
 DimNumber   As Long
 status     As Integer
 Level     As Integer
 Generation   As Integer
 UnaryCalc   As Integer
 MbrTagType   As Integer
 CurrConvert  As Integer
 Attribute   As Variant
 IsAttributed As Integer
End Type
```

VB データ型	フィールド	説明
As String * ESB_ MBRNAMELEN	CrMbrName	タグ付き通貨データベース・メンバーの名前。Time 次元については、タグ付き時間メンバーの名前を与え、Country 次元については、タグ付き通貨メンバーの名前を与えます。Accounts 次元については、タグ付きカテゴリ・メンバーの名前を与えます
As String * ESB_ MBRNAMELEN	MbrName	メンバー名
As String * ESB_ MBRNAMELEN	DimName	メンバーの次元名(ESB_MBRNAMELEN)
As String * ESB_ MBRNAMELEN	ParentMbr Name	指定したメンバーの親の名前またはメンバーに親がない場合は空の文字列
As String * ESB_ MBRNAMELEN	ChildMbrName	指定したメンバーの最初の子のメンバー名
As String * ESB_ MBRNAMELEN	PrevMbrName	指定したメンバーの前の兄弟のメンバーの名前
As String * ESB_ MBRNAMELEN	NextMbrName	指定したメンバーの次の兄弟のメンバー名
As String * ESB_ DESCLEN	Description	メンバーについての説明
As Long	MbrNumber	メンバー数
As Long	DimNumber	メンバーの次元数
As Integer	Status	メンバーの共有ステータスを得るには、このフィールドのコンテンツとフォーム ESB_MBRSTS_xxx の各定数値との間の論理 AND を実行します: ● ESB_MBRSTS_NOTSET ● ESB_MBRSTS_NEVER ● ESB_MBRSTS_LABEL ● ESB_MBRSTS_REFER ● ESB_MBRSTS_REFNME ● ESB_MBRSTS_SHARE ● ESB_MBRSTS_VIRTSTORE ● ESB_MBRSTS_VIRTNOSTORE
As Integer	Level	指定したメンバーの一番下の子孫から数え上げた、メンバーのレベル番号(ゼロから始まる)
As Integer	Generation	指定したメンバーの次元メンバーから下に数えた、メンバーの世代番号(1 から始まる)
As Integer	UnaryCalc	このメンバーに対するデフォルトの単項ロールアップ(ESB_UCAL_xxx という形式の定数値の 1 つ)。add、subtract、multiply、divide、percent または none のいずれかです
As Integer	MbrTagType	メンバーのタグ付きタイプの 16 ビット・マスク(マスクは ESB_ATYPE_xxx という形式になります)

VB データ型	フィールド	説明
As String * 1	CurrConvert	通貨換算。このフィールドには ESB_TRUE および ESB_FALSE を指定できます
As Variant	Attribute	属性値。属性次元またはゼロレベル(リーフ・ノード)の属性メンバーは、次のいずれかのデータ型になります: ● ブール(True|False) ● 日付(「09/19/2006」) ● Double (3.14) ● 文字列(「Hello」) 属性次元ではなく、属性メンバーの場合: ESB_ATTRMBRDT_NONE = 空を含むそれ以外すべて。
As Integer	IsAttributed	メンバーに属性が関連付けられているかどうかを示します。(属性が関連付けられている場合は ESB_TRUE。)

ESB_OBJDEF_T

この構造体は、特定のファイル・オブジェクトについての情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_OBJDEF_T

 hCtx      As Long
 Type      As Long
 AppName    As String * ESB_APPNAMELEN
 DbName    As String * ESB_DBNAMELEN
 FileName    As String * ESB_PATHLEN
End Type
```

VB データ型	フィールド	説明
As Long	hCtx	VB API コンテキスト・ハンドル
As Long	Type	オブジェクト・タイプ。オブジェクト・タイプのリストは、表 15 を参照してください。
As String * ESB_APPNAMELEN	AppName	アプリケーション名
As String * ESB_DBNAMELEN	DbName	データベース名

VB データ型	フィールド	説明
As String * ESB_PATHLEN	FileName	オブジェクトのファイル名。次の場合は、ローカル・ファイル名になります: ● hCtx がローカル・コンテンツ・ハンドル ● AppName および DbName が空の文字列 ● FileName がローカル・ファイルのフルパス名を指す。

ESB_OBJINFO_T

この構造体は、特定のファイル・オブジェクトについての情報を含んでいます。この構造体のフィールドは、API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_OBJINFO_T

  AppName     As String * ESB_APPNAMELEN
  DbName      As String * ESB_DBNAMELEN
  Name       As String * ESB_OBJNAMELEN
  Type       As Long
  FileSize     As Long
  TimeStamp    As Long
  TimeModified  As ESB_TIMERECORD_T
  User       As String * ESB_USERNAMELEN
  Locked      As Integer
End Type
```

VB データ型	フィールド	説明
As String * ESB_ OBJNAMELEN	Name	オブジェクト名
As Long	Type	オブジェクト・タイプ。オブジェクト・タイプのリストは、表 15 を参照してください。
As String * ESB_ APPNAMELEN	AppName	アプリケーション名
As String * ESB_DBNAMELEN	DbName	データベース名
As Long	FileSize	オブジェクトに割り当てられたファイル・サイズ(バイト単位)
As String * 1	Locked	オブジェクトがロックされているかどうかを示すフラグ(ESB_TRUE はオブジェクトがロックされていることを示します)
As String * ESB_ USERNAMELEN	User	オブジェクトをロックしたユーザー名(ロックされている場合)、それ以外の場合は未定義です
As Long	TimeStamp	オブジェクトがロックされた日付および時刻(ロックされている場合)、それ以外の場合は未定義です
As ESB_TIMERECORD_T	Time Modified	最後に行われた変更の日付と時刻

ESB_PART_CONNECT_INFO_T

この構造体は、データベースを指定します。

```
   Type ESB_PART_CONNECT_INFO_T

  HostName    As String * ESB_SVRNAMELEN
  AppName     As String * ESB_APPNAMELEN
  DbName      As String * ESB_DBNAMELEN

End Type
```

データ型	フィールド	説明
String	HostName	ホスト名。
String	AppName	アプリケーション名。
String	DbName	データベース名。

ESB_PART_DEFINED_T

この構造体は、共有パーティションを指定します。

```
   Type ESB_PART_DEFINED_T

  usType     As Integer  ' ESB_PARTITION_OP_REPLICATED, _LINKED, or _TRANSPARENT
  Direction   As Integer  ' ESB_PARTITION_DATA_SOURCE or _TARGET
  HostDatabase  As ESB_PART_CONNECT_INFO_T

End Type
```

データ型	フィールド	説明
Integer	usType	下に示した操作タイプ定数のいずれかになります。
Integer	usDirection	下に示した方向定数のいずれかになります。
1288 ページの「ESB_PART_CONNECT_INFO_T」	HostDatabase	ホスト・サーバー。

操作タイプ定数

```
   define ESB_PARTITION_OP_REPLICATED    0x0001
define ESB_PARTITION_OP_LINKED      0x0002
define ESB_PARTITION_OP_TRANSPARENT   0x0004
define ESB_PARTITION_OP_ALLTYPES     (ESB_PARTITION_OP_REPLICATED |
                                  ESB_PARTITION_OP_LINKED |
                                  ESB_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   define ESB_PARTITION_DATA_SOURCE    0x0001
define ESB_PARTITION_DATA_TARGET    0x0002
define ESB_PARTITION_DATA_BOTH     (ESB_PARTITION_DATA_SOURCE |
                                ESB_PARTITION_DATA_TARGET)
```

ESB_PART_INFO_T

この構造体は、共有パーティション情報を保持します。

```
   Type ESB_PART_INFO_T

  OperationType    As Integer
  DataDirection    As Integer
  MetaDirection    As Integer
  usReserved     As Integer
  LastMetaUpdateTime As Long
  LastRefreshTime   As Long
  AreaUpdatable    As Integer
  IncrRefreshAllowed As Integer
  LastUpdateTime   As Long
  SvrName       As String * ESB_SVRNAMELEN
  AppName       As String * ESB_APPNAMELEN
  DbName       As String * ESB_DBNAMELEN
End Type
```

データ型	フィールド	説明
Integer	OperationTypes	下に示した操作タイプ定数のいずれかになります。
Integer	DirectionTypes	下に示した方向定数のいずれかになります。
Integer	MetaDirectionTypes	下に示した MetaDirection 定数のいずれかになります。
Integer	usReserved	将来の使用に備えて予約済 - 0 に設定されています。
Long	LastMetaUpdateTime	メタデータの前回更新時刻。
次のフィールドは、複製データ・ターゲットにのみ適用されます		
Long	LastRefreshTime	ターゲットのデータの前回リフレッシュ時刻。
Integer	AreaUpdatable	複製されたデータの変更は許可されていますか?
次のフィールドは、複製データ・ソースにのみ適用されます		
Integer	IncrRefreshAllowed	変更されたデータのみをリフレッシュできますか?
Long	LastUpdateTime	パーティション内のデータの前回変更時刻。
次のフィールドは、リモート接続にのみ適用されます。		
String	SvrName	パーティション定義の他の側のホスト。

データ型	フィールド	説明
String	AppName	パーティション定義の他の側のアプリケーション。
String	DbName	パーティション定義の他の側のデータベース。メタデータ変更情報。

操作タイプ定数

```
   #define ESB_PARTITION_OP_REPLICATED   0x0001
#define ESB_PARTITION_OP_LINKED     0x0002
#define ESB_PARTITION_OP_TRANSPARENT   0x0004
#define ESB_PARTITION_OP_ALLTYPES = ESB_PARTITION_OP_REPLICATED
               + ESB_PARTITION_OP_LINKED +
               + ESB_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   #define ESB_PARTITION_DATA_SOURCE    0x0001
#define ESB_PARTITION_DATA_TARGET    0x0002
#define ESB_PARTITION_DATA_BOTH = ESB_PARTITION_DATA_SOURCE
              + ESB_PARTITION_DATA_TARGET)
```

MetaDirection 定数

```
   Global Const ESB_PARTITION_META_SOURCE = 0x0001   'Source metadata partitions
Global Const ESB_PARTITION_META_TARGET = 0x0002   'Target metadata partitions
Global Const ESB_PARTITION_META_BOTH = ESB_PARTITION_META_SOURCE
                + ESB_PARTITION_META_TARGET
```

ESB_PART_REPL_T

この構造体は、共有パーティションにクエリーを行います。

```
   Type ESB_PART_REPL_T

 AreaCount   As Long
 UpdatedOnly  As Integer
End Type
```

データ型	フィールド	説明
Long	AreaCount	リフレッシュするパーティションの数(-1 == ALL)
Integer (Boolean)	UpdatedOnly	前回のリフレッシュ操作以降、ソースで変更されたセルのみをリフレッシュします。
1288 ページの「ESB_PART_CONNECT_INFO_T」	pHostDatabase	パーティション指定の配列。

ESB_PARTOTL_QRY_FILTER_T

この構造体は、メタデータの検索基準を詳細に定義します。

```
   Type ESB_PARTOTL_QRY_FILTER_T

  TimeStamp      As Long
  DimFilter      As Long
  MbrFilter      As Long
  MbrAttrFilter    As Long
End Type
```

データ型	フィールド	説明
Long	TimeStamp	この時刻以降発生したメタデータの変更のクエリー。
Long	DimFilter	次元変更を選択するためのビット・フィールド。
Long	MbrFilter	メンバー変更を選択するためのビット・フィールド。
Long	MbrAttrFilter	メンバー属性変更を選択するためのビット・フィールド。

メンバー属性変更の定数(MbrAttrFilter)

```
   #define ESB_PARTITION_OTLMBRATTR_STATUS     0x0001  /* status changes */
#define ESB_PARTITION_OTLMBRATTR_ALIAS     0x0002  /* alias changes */
#define ESB_PARTITION_OTLMBRATTR_UCALC     0x0004  /* unary calc symbol changes */
#define ESB_PARTITION_OTLMBRATTR_ATYPE     0x0008  /* account type changes */
#define ESB_PARTITION_OTLMBRATTR_CCONVERT    0x0010  /* currency conversion flag */
#define ESB_PARTITION_OTLMBRATTR_CRMBRNAME    0x0020  /* tagged currency db member */
#define ESB_PARTITION_OTLMBRATTR_UDA      0x0040  /* user defined attribute changes
*/
#define ESB_PARTITION_OTLMBRATTR_CALC      0x0080  /* calc formula changes */
#define ESB_PARTITION_OTLMBRATTR_LEVEL     0x0100  /* level number changes */
#define ESB_PARTITION_OTLMBRATTR_GENERATION   0x0200  /* generation number changes */
#define ESB_PARTITION_OTLMBRATTR_ALL      (ESB_PARTITION_OTLMBRATTR_STATUS |
                       ESB_PARTITION_OTLMBRATTR_ALIAS |
                       ESB_PARTITION_OTLMBRATTR_UCALC |
                       ESB_PARTITION_OTLMBRATTR_ATYPE |
                       ESB_PARTITION_OTLMBRATTR_CCONVERT |
                       ESB_MBRATTR_CRMBR_NAME |
                       ESB_PARTITION_OTLMBRATTR_UDA |
                       ESB_PARTITION_OTLMBRATTR_CALC |
                       ESB_PARTITION_OTLMBRATTR_LEVEL |
                       ESB_PARTITION_OTLMBRATTR_GENERATION)
#define ESB_ALLCHG    (ESB_PARTITION_OTLMBR_ALL |
           ESB_DIMCHG_ALL)
```

ESB_PARTOTL_QUERY_T

この構造体は、メタデータの変更にクエリーを行います。

```
   Type ESB_PARTOTL_QUERY_T

 OperationType   As Integer  ' ESB_PARTITION_OP_REPLICATED, _LINKED, _TRANSPARENT
 HostDatabase    As ESB_PART_CONNECT_INFO_T
 MetaFilter     As ESB_PARTOTL_QRY_FILTER_T
End Type
```

データ型	フィールド	説明
Integer	usOperationType	下に示した操作タイプ定数のいずれかになります。
1288 ページの「ESB_PART_CONNECT_INFO_T」	HostDatabase	ホスト・サーバーのデータベース名。
1291 ページの「ESB_PARTOTL_QRY_FILTER_T」	MetaFilter	名前の詳細定義の基準。

操作タイプ定数

```
   #define ESB_PARTITION_OP_REPLICATED   0x0001
#define ESB_PARTITION_OP_LINKED     0x0002
#define ESB_PARTITION_OP_TRANSPARENT   0x0004
#define ESB_PARTITION_OP_ALLTYPES    (ESB_PARTITION_OP_REPLICATED |
                  ESB_PARTITION_OP_LINKED |
                  ESB_PARTITION_OP_TRANSPARENT)
```

ESB_PARTSLCT_T

この構造体は、指定されたサイトの共有パーティションにクエリーを行います。

```
   Type ESB_PARTSLCT_T

 OperationTypes    As Integer
 DirectionTypes    As Integer
 MetaDirectionTypes  As Integer
End Type
```

データ型	フィールド	説明
Integer	OperationTypes	下に示した操作タイプ定数のいずれかになります。
Integer	DirectionTypes	下に示した方向定数のいずれかになります。
Integer	MetaDirectionTypes	下に示したメタ方向定数のいずれかになります。

操作タイプ定数

```
   #define ESB_PARTITION_OP_REPLICATED  0x0001
#define ESB_PARTITION_OP_LINKED    0x0002
```

```
#define ESB_PARTITION_OP_TRANSPARENT  0x0004
#define ESB_PARTITION_OP_ALLTYPES = ESB_PARTITION_OP_REPLICATED
                + ESB_PARTITION_OP_LINKED +
                + ESB_PARTITION_OP_TRANSPARENT)
```

方向定数

```
   #define ESB_PARTITION_DATA_SOURCE    0x0001
#define ESB_PARTITION_DATA_TARGET    0x0002
#define ESB_PARTITION_DATA_BOTH = ESB_PARTITION_DATA_SOURCE
               + ESB_PARTITION_DATA_TARGET)
```

MetaDirection 定数

```
   Global Const ESB_PARTITION_META_SOURCE = 0x0001   'Source metadata partitions
Global Const ESB_PARTITION_META_TARGET = 0x0002   'Target metadata partitions
Global Const ESB_PARTITION_META_BOTH = ESB_PARTITION_META_SOURCE
                + ESB_PARTITION_META_TARGET
```

ESB_PROCSTATE_T

非同期操作(計算など)を実行すると、この構造体は **EsbGetProcessState()**への呼出しから戻されます。これによって、呼出し元は非同期操作のステータスを判定できます。

注： このリリースの VB API では、設定されているのは State フィールドのみです。その他は今後使用するためのフィールドです。

```
   Type ESB_PROCSTATE_T

  Action    As Integer
  State     As Integer
  Reserved1   As Integer
  Reserved2   As Long
  Reserved3   As Long
End Type
```

VB データ型	フィールド	説明
As Integer	Action	現在プロセスのアクション(使用されていません)
As Integer	State	現在プロセスの状態(終了または進行中)。値: ● ESB_STATE_DONE (0) ● ESB_STATE_INPROGRESS (1) ● ESB_STATE_FINALSTAGE (5)

VB データ型	フィールド	説明
As Integer	Reserved1	今後の使用に予約
As Long	Reserved2	今後の使用に予約
As Long	Reserved3	今後の使用に予約

ESB_RATEINFO_T

この構造体は、通貨レートについての情報を含んでいます。この構造体のフィールドは、VB API を使用して変更できません。フィールドは次のとおりです:

```
   Type ESB_RATEINFO_T

  MbrName    As String * ESB_MBRNAMELEN
  RateMbr    As String * ESB_RATEINFOLEN

End Type
```

VB データ型	フィールド	説明
As String * ESB_MBRNAMELEN	MbrName	メンバー名(ESB_MBRNAMELEN)
As String * ESB_MBRNAMELEN	RateMbr	レート・メンバー名の配列(ESB_MBRNAME_T)

ESB_TIMERECORD_T

この構造体は 1273 ページの「ESB_DBREQINFO_T」構造体で使用されます。フィールドは次のとおりです:

```
   Type ESB_TIMERECORD_T

  TimeValue  As Long
  Seconds    As Integer
  Minutes    As Integer
  Hours      As Integer
  Day        As Integer
  Month      As Integer
  Year       As Integer
  Weekday    As Integer
  Reserved   As Integer
End Type
```

VB データ型	フィールド	説明
As Long	TimeValue	1/1/70 以降の秒単位の時間値
As Integer	Seconds	分の後の秒。値: 0-59。

VB データ型	フィールド	説明
As Integer	Minutes	時間の後の分。値: 0-59。
As Integer	Hours	深夜から数えた時間数。値: 0-23。
As Integer	Day	月の日付。値: 1-31。
As Integer	Month	1 月から数えた月数。値: 0-11。1 月=0。
As Integer	Year	1990 年から数えた年数。
As Integer	Weekday	日曜から数えた日数。値: 0-6。日曜=0。

ESB_USERAPP_T、ESB_GROUPAPP_T

この構造体は、ユーザーまたはグループ、および特定のアプリケーション向けのアクセス権情報を含んでいます。この構造体の Access および MaxAccess フィールドのみ、VB API を使用して変更できます。フィールドは次のとおりです:

```
   Type ESB_USERAPP_T

  Access    As Integer
  MaxAccess   As Integer
  userName   As String * ESB_USERNAMELEN
  AppName    As String * ESB_APPNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_ USERNAMELEN	UserName	ユーザーまたはグループ名(ESB_USERNAMELEN)
As String * ESB_ APPNAMELEN	AppName	アプリケーション名(ESB_APPNAMELEN)
As Integer	Access	ユーザーまたはグループに対して割り当てられたアプリケーションへのアクセス権。値: ● ESB_PRIV_NONE ● ESB_PRIV_APPLOAD ● ESB_PRIV_APPDESIGN
As Integer	MaxAccess	ユーザーまたはグループに割り当てられた、すべてのソースからのアプリケーションへの最大アクセス権

ESB_USERDB_T、ESB_GROUPDB_T

この構造体にはユーザーまたはグループ、および特定のデータベースに関するアクセス権限情報が含まれています。この構造体の中でアクセス、MaxAccess およびフィルタフィールドのみが、VB API を使用して変更できます。フィールドは次のとおりです:

```
  Type ESB_USERDB_T

 Access    As Integer
 MaxAccess  As Integer
 AppName    As String * ESB_APPNAMELEN
 DbName    As String * ESB_DBNAMELEN
 userName   As String * ESB_USERNAMELEN
 FilterName  As String * ESB_FTRNAMELEN
End Type
```

VB データ型	フィールド	説明
As String * ESB_USERNAMELEN	UserName	ユーザー名またはグループ名(ESB_USERNAMELEN)
As String * ESB_APPNAMELEN	AppName	アプリケーション名(ESB_APPNAMELEN)
As String * ESB_DBNAMELEN	DbName	データベース名(ESB_DBNAMELEN)
As Integer	Access	ユーザーまたはグループに対して割り当てられたデータベースへのアクセス権限。値: ● ESB_PRIV_NONE ● ESB_PRIV_READ ● ESB_PRIV_WRITE ● ESB_PRIV_CALC ● ESB_PRIV_DBLOAD ● ESB_PRIV_DBDESIGN これらの値は表 15 のサブセットです。
As Integer	MaxAccess	ユーザーまたはグループに割り当てられた、すべてのソースからのデータベースへの最大アクセス権限
As String * ESB_FTRNAMELEN	FilterName	割り当てられたデータベース・フィルタの名前(ある場合)。ない場合、このフィールドは空の文字列です。

ESB_USERINFO_T, ESB_GROUPINFO_T

この構造体は、ユーザーまたはグループに関する情報を保管します。

```
  Type ESB_USERINFO_T

 LastLogin   As Long
 DbConnectTime As Long
 LoginId    As Long
 Login     As Integer
 Type      As Integer
 Access     As Integer
 MaxAccess   As Integer
 Expiration  As Integer
```

```
  FailCount   As Integer
  Name      As String * ESB_USERNAMELEN
  AppName    As String * ESB_APPNAMELEN
  DbName     As String * ESB_DBNAMELEN
  Description  As String * ESB_DESCLEN
  EMailID    As String * ESB_DESCLEN
  LockedOut   As Boolean
  PwdChgNow   As Boolean
End Type
```

一部のフィールドはユーザーに特有であり、グループには使用できません。この構造体の「Access」、「Expiration」、および「PwdChgNow」フィールドのみ、API を使用して変更できます。フィールドは次のとおりです:

VB データ型	フィールド	説明
As Long	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付(ユーザーのみ)。
As Long	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。EsbSetUser では設定できません。
As Long	LoginId	ユーザー・ログイン識別タグ(ユーザーのみ)。
As Integer	Login	ログインしたかどうかを示すフラグ(ユーザーのみ)。
As Integer	Type	構造体のタイプ(ユーザーまたはグループ)。値: ● ESB_TYPE_USER ● ESB_TYPE_GROUP
As Integer	Access	ユーザーまたはグループに割り当てられたデフォルトのアクセス権限。このフィールドの値は、次のビット値を任意に組み合せられます: ● ESB_ACCESS_SUPER /*管理者用に全ビットを設定*/ ● ESB_PRIV_APPCREATE ● ESB_PRIV_USERCREATE
As Integer	MaxAccess	ユーザーの最大アクセス権限(ユーザーのみ)。これは、個別のアクセス権とグループ・メンバーシップによって付与されたアクセス・レベルを組み合せたものです。
As Integer	Expiration	今後使用するために予約されています。
As Integer	FailCount	最後に正常にログインしてからの、失敗したログインの回数(ユーザーのみ)。
As String * ESB_USERNAMELEN	Name	ユーザー名またはグループ名(ESB_USERNAMELEN)。
As String * ESB_APPNAMELEN	AppName	現在接続されているアプリケーションの名前(該当する場合)(ESB_APPNAMELEN)。
As String * ESB_DBNAMELEN	DbName	現在接続されているデータベースの名前(該当する場合)(ESB_DBNAMELEN)。

VB データ型	フィールド	説明
As String	Description	ユーザーまたはグループに関する説明(ESB_DESCLEN)。 将来使用するために予約されています。ユーザーは設定できません。
As String	EMailID	ユーザーまたはグループの電子メール・アドレス(ESB_DESCLEN)。 将来使用するために予約されています。ユーザーは設定できません。
As Boolean	LockedOut	ユーザーがロック・アウトされていることを示すフラグ。
As Boolean	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ。

ESB_USERINFOEX_T

この構造体は、ユーザーまたはグループに関する情報を保管します。

```
  Type ESB_USERINFOEX_T

 LastLogin   As Long
 DbConnectTime As Long
 LoginId    As Long
 Login     As Integer
 Type      As Integer
 Access     As Integer
 MaxAccess   As Integer
 Expiration   As Integer
 FailCount   As Integer
 Name      As String * ESB_USERNAMELEN
 AppName    As String * ESB_APPNAMELEN
 DbName     As String * ESB_DBNAMELEN
 Password    As String * ESB_PASSWORDLEN      ' Authentication Password
 Description  As String * ESB_DESCLEN
 EMailID    As String * ESB_DESCLEN
 LockedOut   As Boolean
 PwdChgNow   As Boolean
 protocol    As String * ESB_PROTOCOLNAMELEN     ' External Authentication Protocol
 connparam   As String * ESB_CONNPARAMLEN      ' External Authentication Connection

End Type
```

一部のフィールドはユーザーに特有であり、グループには使用できません。この構造体の「Access」、「Expiration」、および「PwdChgNow」フィールドのみ、API を使用して変更できます。フィールドは次のとおりです:

VB データ型	フィールド	説明
As Long	LastLogin	グリニッジ標準時刻で示した、ユーザーが最後に正常にログインした日付(ユーザーのみ)。
As Long	DbConnectTime	データベース接続のローカル(サーバー)時刻。読取り専用。EsbSetUser では設定できません。
As Long	LoginId	ユーザー・ログイン識別タグ(ユーザーのみ)。

VB データ型	フィールド	説明
As Integer	Login	ログインしたかどうかを示すフラグ(ユーザーのみ)。
As Integer	Type	構造体のタイプ(ユーザーまたはグループ)。値: ● ESB_TYPE_USER ● ESB_TYPE_GROUP
As Integer	Access	ユーザーまたはグループに割り当てられたデフォルトのアクセス権限。このフィールドの値は、次のビット値を任意に組み合せられます: ● ESB_ACCESS_SUPER /*管理者用に全ビットを設定*/ ● ESB_PRIV_APPCREATE ● ESB_PRIV_USERCREATE
As Integer	MaxAccess	ユーザーの最大アクセス権限(ユーザーのみ)。これは、個別のアクセス権とグループ・メンバーシップによって付与されたアクセス・レベルを組み合せたものです。
As Integer	Expiration	今後使用するために予約されています。
As Integer	FailCount	最後に正常にログインしてからの、失敗したログインの回数(ユーザーのみ)。
As String * ESB_ USERNAMELEN	Name	ユーザー名またはグループ名(ESB_USERNAMELEN)。
As String * ESB_ APPNAMELEN	AppName	現在接続されているアプリケーションの名前(該当する場合)(ESB_ APPNAMELEN)。
As String * ESB_ DBNAMELEN	DbName	現在接続されているデータベースの名前(該当する場合)(ESB_ DBNAMELEN)。
As String * ESB_ PASSWORDLEN	Password	外部認証済ユーザーのパスワード。Essbase の認証済メカニズムに外部認証済ユーザーを設定する場合にのみ、これを使用します。このパスワードは、サーバーから外部認証済ユーザーの情報を取得するなど、他の状況では無視されます。
As String	Description	ユーザーまたはグループに関する説明(ESB_DESCLEN)。 将来使用するために予約されています。ユーザーは設定できません。
As String	EMailID	ユーザーまたはグループの電子メール・アドレス(ESB_DESCLEN)。 将来使用するために予約されています。ユーザーは設定できません。
As Integer	LockedOut	ユーザーがロック・アウトされていることを示すフラグ。
As Integer	PwdChgNow	ユーザーがパスワードを変更する必要があることを示すフラグ。
As String * ESB_ PROTOCOLNAMELEN	protocol	外部認証プロトコル。
As String * ESB_ CONNPARAMLEN	connparam	外部認証接続。

ESB_VARIABLE_T

ESB_VARIABLE_T はプライマリ代替変数データ型です。代替変数の値と名前、および変数が定義される Essbase データベース、アプリケーション、サーバーを識別します。

サーバー名はオプションですが、推奨します。サーバー名を指定しない場合、現在のサーバーがデフォルトになります。AppName はオプションです。DbName はオプションですが、存在する場合は、AppName メンバーが必要です。VarName は必須です。VarValue は必須です。

```
   Type ESB_VARIABLE_T

  Server    As String * ESB_SVRNAMELEN
  AppName    As String * ESB_APPNAMELEN
  DbName    As String * ESB_DBNAMELEN
  VarName    As String * ESB_MBRNAMELEN
  VarValue    As String * ESB_VARVALUELEN

End Type
```

VB データ型	フィールド	説明
ESB_SVRNAME_T	Server	変数が定義されているサーバーの名前(オプション)
ESB_APPNAME_T	AppName	変数を制限するアプリケーションの名前
ESB_DBNAME_T	DbName	変数を制限するデータベースの名前。使用する場合、アプリケーションの設定が必要です。
ESB_MBRNAME_T	VarName	代替変数の名前。
ESB_CHAR_T	VarValue[256]	代替変数の値。

20 Visual BasicのメインAPI関数

この章の内容

Visual Basic のメイン API 関数のカテゴリ

VB のメイン API 別名テーブル関数

別名テーブルの関数は、データベースの別名テーブルを管理します。

関数	説明
EsbListAliases	アクティブ・データベース内のすべての別名テーブルをリストします。

関数	説明
EsbLoadAlias	構造化されたテキスト・ファイルから、アクティブ・データベースの別名テーブルをロードします
EsbGetAlias	1 人のユーザーについて、アクティブなデータベースからアクティブな別名テーブル名を取得します。
EsbSetAlias	1 人のユーザーについて、アクティブなデータベースにアクティブな別名テーブルを設定します。
EsbDisplayAlias	アクティブ・データベース内の別名テーブルのコンテンツをダンプします。
EsbRemoveAlias	アクティブなデータベースから別名テーブルを削除します。
EsbClearAliases	アクティブ・データベースのすべての別名テーブルを消去します。

VB のメイン API アプリケーション関数

アプリケーション関数は、新規アプリケーションの作成および既存のアプリケーションの変更、コピー、情報の取得、管理を行います。

関数	説明
EsbGetActive	呼出し元の現在のアクティブなアプリケーションとデータベースの名前を取得します。
EsbSetActive	呼出し元のアクティブなアプリケーションとデータベースを設定します。
EsbClearActive	ユーザーの現在のアクティブなアプリケーションおよびデータベースを消去します。
EsbListApplications	呼出し元がアクセスできる、すべてのアプリケーションをリストします。
EsbCreateApplication	クライアントまたはサーバー上で、新規アプリケーションを作成します。
EsbCreateStorageTypedApplication	多次元または集約ストレージのいずれかのオプションで新しいアプリケーションを作成します。
EsbDeleteApplication	クライアント上またはサーバー上で、既存のアプリケーションを削除します。
EsbRenameApplication	クライアント上またはサーバー上で、既存のアプリケーションの名前を変更します。
EsbGetApplicationInfoEx	1 つ以上のアプリケーションから情報を取得します。
EsbCopyApplication	クライアント上またはサーバー上の既存のアプリケーションを、関連するすべてのデータベースとオブジェクトも含めて、新規アプリケーションにコピーします。
EsbGetApplicationState	ユーザーが構成可能なアプリケーションのパラメータが含まれている、アプリケーションの状態構造体を取得します。
EsbSetApplicationState	アプリケーションの状態構造体を使用して、ユーザーが構成可能なアプリケーションのパラメータを設定します。

関数	説明
EsbGetApplicationInfo	ユーザーが構成不可能なアプリケーションのパラメータが含まれている、アプリケーションの情報構造体を取得します。
EsbLoadApplication	サーバー上のアプリケーションを開始します。
EsbUnloadApplication	サーバー上のアプリケーションを停止します。

VB のメイン API 属性関数

次の Visual Basic メイン関数は、属性に関するものです。

関数	説明
EsbCheckAttributes	指定した属性次元、基本次元、属性メンバーおよび基本メンバーに対する属性のタイプを戻します
EsbGetAssociatedAttributesInfo	指定した基本メンバーに関連付けられている属性メンバーを戻します
EsbGetAttributeInfo	指定した属性メンバーまたは属性次元に関する属性情報を戻します
EsbGetAttributeSpecifications	アウトラインの属性指定を取得します

VB アウトライン API1620 ページの「VB アウトライン API 属性の関数」を参照するには、ここをクリックしてください。

VB のメイン API データベース関数

データベース関数は、データベース管理タスクを実行し、データベース情報構造体の取得と変更を行います。

関数	説明
EsbClearDatabase	アクティブ・データベース内にロードされているすべてのデータを消去します。
EsbCopyDatabase	クライアント上またはサーバー上の既存のデータベースを、関連するすべてのデータベースおよびオブジェクトも含めて、新規のデータベースにコピーします。
EsbCreateDatabase	クライアントまたはサーバー上で、アプリケーション内に新規データベースを作成します。
EsbDeleteDatabase	クライアントまたはサーバー上で、アプリケーションから既存のデータベースを削除します。
EsbGetCurrencyRateInfo	アクティブ・データベース・アウトライン内の、タグ付き通貨パーティション次元のすべてのメンバーについてのレート情報が含まれている構造体のリストを取得します。
EsbGetDatabaseInfo	ユーザーが構成不可能なデータベースのパラメータが含まれている、データベースの情報構造体を取得します。

関数	説明
EsbGetDatabaseNote	データベースの最新情報に関するメッセージを取得します。
EsbGetDatabaseState	ユーザーが構成可能なデータベースのパラメータが含まれている、データベースの状態構造体を取得します。
EsbGetDatabaseStats	データベースに関する統計情報が含まれている、アクティブ・データベースの統計構造体を取得します。
EsbListCurrencyDatabases	呼出し元がアクセス可能な、特定のアプリケーション内のすべての通貨データベースをリストします。
EsbListDatabases	呼出し元がアクセス可能な、特定のアプリケーション内またはサーバー全体の、すべてのデータベースをリストします。
EsbLoadDatabase	アプリケーション内のデータベースをサーバー上で開始します。
EsbRenameDatabase	クライアントまたはサーバー上で、アプリケーション内の既存のデータベースの名前を変更します。
EsbSetDatabaseNote	データベースの最新情報に関するメッセージを設定します。
EsbSetDatabaseState	データベースの状態構造体を使用して、ユーザーが構成可能なデータベースのパラメータを設定します。
EsbUnloadDatabase	サーバー上でアプリケーション内のデータベースを停止します。
EsbValidateDB	データの整合性のためにデータベースを検証します。

VB のメイン API データベース・メンバー関数

これらの関数は、データベース・メンバーに関する情報を取得し、データベースの次元を構築します。

関数	説明
EsbQueryDatabaseMembers	レポートスタイルのクエリーを実行して、選択したデータベース・メンバーの情報をリストします。
EsbCheckMemberName	文字列がアクティブ・データベース・アウトライン内で有効なメンバー名であるかどうか確認します。
EsbGetMemberInfo	アクティブ・データベース・アウトライン内の、特定のメンバーに関する情報が含まれている構造体を取得します。
EsbGetMemberCalc	アクティブ・データベース・アウトライン内の、特定のメンバーの計算式を取得します。
EsbGetDimensionInfo	次元に関する情報を取得します。
EsbBuildDimension	データ・ファイルおよびルール・ファイルからのアクティブ・データベース内での次元の作成を可能にします。
EsbBuildDimFile	この関数は、アクティブ・データベースのアウトラインからのメンバーの追加または削除に使用するデータ・ファイルを作成します。
EsbBuildDimStart	この関数は、アクティブ・データベースのアウトラインからメンバーの追加または削除プロセスを開始します。

VB のメイン API のドリルスルー関数

次に示すドリルスルー関数は、Oracle ERP および EPM アプリケーション上でホストされている情報にドリルスルーするためのドリルスルー URL を管理します。

関数	説明
EsbCreateDrillThruURL	アクティブなデータベース・アウトライン内に、指定されたリンクと名前を使用してドリルスルー URL を作成します。
EsbDeleteDrillThruURL	アクティブなデータベース・アウトライン内で、指定された URL 名のドリルスルー URL を削除します。
EsbGetCellDrillThruReports	データ・セルに関連付けられたドリルスルー・レポートを、セルのメンバーの組合せを使用し、URL XML のリストとして取得します。
EsbGetDrillThruURL	アクティブなデータベース・アウトライン内のドリルスルー URL 名のリストを取得します。
EsbListDrillThruURLs	アクティブなデータベース・アウトライン内のドリルスルー URL をリストします。
EsbUpdateDrillThruURL	アクティブなデータベース・アウトライン内で、指定された名前のドリルスルー URL を更新します。

VB のメイン API ファイル関数

ファイル関数によって、アプリケーションは定義済のレポート・スクリプト、データ・ファイル、計算スクリプトをアクティブなデータベースに対して使用できます。テキスト・ファイルとバイナリ・ファイルの両方との間で、データのインポートとエクスポートも行います。

関数	説明
EsbArchiveBegin	READ-ONLY ステータスに設定して、データベースをアーカイブ操作のために準備します。
EsbArchiveEnd	アーカイブ操作の後、データベース・ステータスを READ-WRITE に戻します。
EsbCalcFile	ファイルからアクティブなデータベースに対して計算スクリプトを実行します。
EsbExport	現在のデータベースからテキスト・ファイルへのデータ・エクスポートを可能にします。
EsbImport	テキスト・ファイルおよびその他のソースから現在のデータベースへのデータ・インポートを可能にします。
EsbListDbFiles	指定したインデックスおよびデータ・ファイルに関する情報を取得します
EsbReportFile	ファイルからアクティブなデータベースへレポート指定を送信します。
EsbSetDefaultCalcFile	計算スクリプト・ファイルからアクティブ・データベースに対してデフォルト計算スクリプトを設定します。
EsbUpdateFile	ファイルからアクティブ・データベースに対して更新指定を送信します。

VB のメイン API グループ管理関数

これらの関数は、グループの作成、グループ属性の設定と変更、および既存のグループに関する情報の取得を行います。

関数	説明
EsbListGroups	特定の Essbase サーバーに対してアクセス権を所有しているすべてのグループをリストします。
EsbCreateGroup	グループを新規作成します。
EsbDeleteGroup	既存のグループを削除します。
EsbRenameGroup	既存のグループの名前を変更します。
EsbGetGroup	グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。
EsbSetGroup	グループ情報構造体を設定します。
EsbGetGroupList	グループのメンバーであるユーザーのリスト(またはユーザーが属するグループのリスト)を取得します。
EsbSetGroupList	グループのメンバーであるユーザーのリストを設定します。
EsbAddToGroup	グループ・メンバーのリストにユーザーを追加します。
EsbDeleteFromGroup	グループ・メンバーのリストからユーザーを削除します

VB のメイン API 初期化およびログイン関数

この種の関数を使用して API の初期化、Essbase サーバーへのログインとログアウトを行います。バージョン情報の入手、アプリケーションによるローカル・コンテンツの作成と削除も行えます。

関数	説明
EsbAutoLogin	ユーザーが Essbase サーバーにログインするためのダイアログ・ボックスを表示します。また、オプションでアクティブなアプリケーションとデータベースを選択します。
EsbCreateLocalContext	ローカル API 操作で使用するローカル API コンテキストを作成します
EsbDeleteLocalContext	以前に EsbCreateLocalContext()で作成されたローカル・コンテキストをリリースします
EsbGetAPIVersion	使用中の API DLL のバージョン番号を取得します
EsbGetVersion	接続されている Essbase サーバーの完全なバージョン番号を取得します。
EsbInit	API およびメッセージ・データベースを初期化します。
EsbLogin	ユーザーを Essbase サーバーにログインさせます。
EsbLoginSetPassword	ユーザーをログインさせ、パスワードを変更します。

関数	説明
EsbLogout	ユーザーを Essbase サーバーからログアウトさせます。
EsbLogoutUser	管理者またはアプリケーション・マネージャが他のユーザーを Essbase サーバーから切断できるようにします。
EsbShutdownServer	管理者がリモートからエージェントを停止できるようにします。
EsbTerm	API を終了し、API で使用されているすべてのシステム・リソースを解放します。
EsbValidateHCtx	特定の API コンテキスト・ハンドル(hCtx)を検証します。

VB のメイン API LRO 関数

これらの関数は、LRO を作成、取得および削除し、LRO に関する情報を戻します。

関数	説明
EsbLROAddObject	レポート・オブジェクトを Essbase データベースのデータ・セルにリンクします。
EsbLRODeleteCellObjects	Essbase データベースの指定されたデータ・セルにリンクされているすべてのオブジェクトを削除します。
EsbLRODeleteObject	Essbase データベースのデータ・セルにリンクされている特定のオブジェクトを削除します。
EsbLROGetCatalog	Essbase データベース内の指定したデータ・セルについて、LRO カタログ・エントリのリストを取得します。
EsbLROGetMemberCombo	現在の LRO のメンバー組合せリストから、n 番目のメンバーを取得します。
EsbLROGetObject	Essbase データベース内のデータ・セルにリンクされているオブジェクトを取得します。
EsbLROListObjects	指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトのリストを取得します。
EsbLROPurgeObjects	指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトを削除します。
EsbLROUpdateObject	サーバーに LRO の更新済バージョンを保管します。

VB のメイン API のロケーション別名関数

これらの関数は、ロケーション別名を作成、削除およびリストします。

関数	説明
EsbCreateLocationAlias	別名を、ホスト名、アプリケーション名、データベース名、ユーザー・ログイン名およびユーザー・パスワードにマッピングします

関数	説明
EsbDeleteLocationAlias	既存のロケーション別名を削除します
EsbGetLocationAliasList	すべてのロケーション別名およびそのロケーション別名がマッピングされている名前を戻します

VB のメイン API のその他の関数

これらの関数は、非同期プロセスの管理、状態情報の取得、ログ・ファイルのプロセスおよびメッセージの取得を行います。

関数	説明
EsbGetProcessState	計算またはデータ・インポートなどの、非同期プロセスの現在の状態を取得します。
EsbCancelProcess	まだ完了していない非同期プロセスを取り消します。
EsbGetLogFile	アプリケーション・ログ・ファイルの一部または全部を、サーバーからクライアントにコピーします。
EsbDeleteLogFile	サーバー上のアプリケーション・ログ・ファイルを削除します。
EsbGetGlobalState	システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を取得します。
EsbSetGlobalState	システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を設定します。
EsbGetNextItem	システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を取得します。
EsbGetMessage	初期化の際に **ESB_INIT_T** 構造体の ClientError が ESB_TRUE に設定されていた場合は、この関数は、VB API 関数の実行中に積み上げられたメッセージ・スタックから 1 番上のメッセージを取得します。
EsbSetPath	現在のプロセスに対して、ARBORPATH 環境変数を設定します。

VB のメイン API オブジェクト関数

これらの関数は、オブジェクトを作成、削除、移動、コピーします。また、オブジェクト情報を取得して表示し、オブジェクトへのアクセスを制御します。

関数	説明
EsbGetLocalPath	クライアント上のオブジェクト・ファイルの完全なローカル・ファイルを取得します。
EsbListObjects	指定したタイプのすべてのオブジェクトをリストします。
EsbGetObjectInfo	指定したオブジェクトに関する情報を取得します。
EsbGetObject	オブジェクトをサーバーからローカル・ファイルにコピーし、オプションでオブジェクトをロックします。

関数	説明
EsbPutObject	オブジェクトをローカル・ファイルからサーバーにコピーし、オプションでオブジェクトのロックを解除します。
EsbLockObject	他のユーザーによる更新を防ぐため、サーバー上のオブジェクトをロックします。
EsbUnlockObject	サーバー上のロックされたオブジェクトのロックを解除します。
EsbCreateObject	オブジェクトを新規作成します。
EsbDeleteObject	既存のオブジェクトを削除します。
EsbRenameObject	既存のオブジェクトの名前が変更されます。
EsbCopyObject	オブジェクトをコピーします。

VB のメイン API のレポート、更新、計算関数

これらの関数は、アクティブ・データベースに対してレポート作成タスク(データの取得)、更新タスク(データのロード)および計算タスク(データの集約)を実行します。

関数	説明
EsbBeginCalc	計算スクリプトの送信を開始し、オプションでアクティブ・データベースに対して計算スクリプトを実行します。
EsbBeginReport	アクティブなデータベースへのレポート指定の送信を開始します。
EsbBeginUpdate	アクティブ・データベースに対して更新指定の送信を開始します。
EsbCalc	アクティブ・データベースに対して計算スクリプトを単一の文字列として送信し、オプションで計算スクリプトを実行します。
EsbDefaultCalc	アクティブ・データベースのデフォルト計算を実行します。
EsbEndCalc	アクティブなデータベースに送信される計算スクリプトの終わりをマークします。
EsbEndReport	アクティブなデータベースに送信されるレポート指定の終わりをマークします。
EsbEndUpdate	アクティブなデータベースに送信される更新指定の終了をマークします。
EsbGetDefaultCalc	アクティブ・データベースのデフォルト計算スクリプトを取得します。
EsbGetString	アクティブ・データベースから文字列データを取得します。
EsbGetStringBuf	アクティブ・データベースから入手可能なデータがすべて戻されるまで、または呼出し元のバッファがいっぱいになるまでデータを取得します。
EsbReport	レポート指定を単一文字列としてアクティブなデータベースに送信します。
EsbSendString	アクティブなデータベースにデータの文字列を送信します。
EsbSetDefaultCalc	データベースのデフォルト計算スクリプトを設定します。

関数	説明
EsbUpdate	アクティブ・データベースに対して単一文字列として更新を送信します。

VB のメイン API セキュリティ・フィルタ関数

セキュリティ・フィルタ関数は、フィルタのコンテンツの設定、ユーザー・グループへのフィルタの割当て、データベースのフィルタ・リストの表示およびセキュリティ・フィルタに関するその他のデータの取得を行います。

関数	説明
EsbListFilters	データベースのすべてのフィルタをリストします。
EsbGetFilter	フィルタのコンテンツの取得を開始します。
EsbGetFilterRow	フィルタの次の行を取得します。
EsbSetFilter	フィルタのコンテンツの設定を開始します。
EsbSetFilterRow	フィルタの次の行を取得します。
EsbGetFilterList	フィルタを割り当てられたユーザーのリストを取得します。
EsbSetFilterList	フィルタを割り当てられたユーザーのリストを設定します。
EsbDeleteFilter	既存のフィルタを削除します。
EsbRenameFilter	既存のフィルタの名前を変更します。
EsbCopyFilter	既存のフィルタをコピーします。
EsbVerifyFilter	指定したデータベースに照らしあわせて、一連のフィルタ行の文字列の構文を確認します。
EsbVerifyFilterRow	指定したデータベースに照らしあわせて、単一のフィルタ行の文字列の構文を確認します。

VB のメイン API 代替変数の関数

これらの関数は、代替変数を作成、取得および削除し、代替変数に関する情報を戻します。

関数	説明
EsbCreateVariable	この関数は代替変数を新規作成します。または同一のサーバー値、アプリケーション値およびデータベース値の変数名がすでに存在する場合は、既存の代替変数を変更します。
EsbDeleteVariable	この関数は、代替変数を削除します。
EsbGetVariable	この関数は、代替変数の値を取得します。
EsbGetVariable	この関数は、入力基準に適合する代替変数をすべてリストします。

VB のメイン API ユーザー管理関数

ユーザー管理関数では、ユーザーの作成、パスワードの割当て、データベース、アプリケーション、計算スクリプトへのアクセス権の設定を行います。この関数は、ユーザー機能に関する情報を取得するためにも使用できます。

関数	説明
EsbListUsers	特定の Essbase サーバーへのアクセス権があるすべてのユーザーをリストします。
EsbCreateUser	新規ユーザーを作成します。
EsbDeleteUser	既存のユーザーを削除します。
EsbRenameUser	既存のユーザー名を変更します。
EsbGetUser	ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。
EsbSetUser	ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。
EsbResetUser	ユーザーのセキュリティ構造体を最初の状態にリセットします。
EsbSetPassword	既存のパスワードを消去して、ユーザーのパスワードを設定します。
EsbGetApplicationAccess	アプリケーションへのユーザーのアクセス権情報が含まれているユーザー・アプリケーション・アクセス構造体のリストを取得します。
EsbSetApplicationAccess	ユーザー・アプリケーション・アクセス構造体のリストを設定します。
EsbGetDatabaseAccess	ユーザー・データベース・アクセス構造体のリストを取得します。
EsbSetDatabaseAccess	ユーザー・データベース・アクセス構造体のリストを設定します。
EsbGetCalcList	ユーザーがアクセス可能な計算スクリプト・オブジェクトのリストを取得します。
EsbSetCalcList	ユーザーが使用可能な計算スクリプト・オブジェクトのリストを設定します。
EsbListConnections	現在のアプリケーションおよびデータベースに接続されているすべてのユーザーをリストします。
EsbListLocks	特定のアプリケーションおよびデータベースに接続されているすべてのユーザーをリストします。
EsbRemoveLocks	現在ユーザーがロックしているデータベースのデータ・ブロックのロックをすべて解除します。

Visual Basic のメイン API 関数のリファレンス

「コンテンツ」ペインで、**Esb** が前に付いた Visual Basic のメイン API 関数のアルファベット順リストを参照してください。

EsbAddToGroup

グループ・メンバーのリストにユーザーを追加します。

構文

```
      EsbAddToGroup
       (
      hCtx, GroupName, User
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
ByVal
      User
        As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
GroupName	グループ名。
User	グループ・リストに追加するユーザー名。

備考

指定したグループのメンバー・リストに指定したユーザーが追加されるのみでなく、この関数はユーザーに関連付けられたグループ・リストにグループも追加します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbAddToGroup Lib "ESBAPIN" (ByVal hCtx As Long,
ByVal GroupName As String, ByVal User As String) As Long

Sub ESB_AddToGroup ()
 Dim sts As Long
 Dim GroupName As String
 Dim User As String
 GroupName = "PowerUsers"
 User = "Jim Smith"
 '******************
 ' Add user to group
 '******************
```

```
 sts = EsbAddToGroup (hCtx, GroupName, User)
End Sub
```

関連トピック

- EsbDeleteFromGroup
- EsbGetGroupList
- EsbListGroups
- EsbSetGroupList

EsbArchive

使用されなくなりました。

この関数は、Essbase の以前のリリースとの互換性のために保持されています。現在の Essbase アーカイブについては、EsbArchiveBegin および EsbArchiveEnd を参照してください。この関数は、エラー・メッセージ **ESB_STS_OBSOLETE** を戻します。

関連トピック

- EsbRestore
- EsbGetProcessState
- EsbArchiveBegin
- EsbArchiveEnd

EsbArchiveBegin

サーバー・モードを「読取り専用」に変更して、サーバーでアーカイブの準備をします。

構文

```
      EsbArchiveBegin
       (
      hCtx, AppName, DbName, FileName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FileName
       As String
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
AppName	アーカイブするアプリケーション名。
DbName	アーカイブするデータベース名。
FileName	アーカイブ情報を含むファイルの名前。

備考

- この関数はサーバー・モードを読取り専用に変更します。これによって、データベース管理者はサーバー上のすべてのファイルをバックアップでき、バックアップ中にファイルに書き込まれないようにできます。バックアップするデータベース・ファイルは、FileName パラメータで指定される app\db ディレクトリにリストされます。
- 指定したファイル内の既存の情報はすべて、アーカイブされたデータによって上書きされます。

戻り値

なし。

アクセス

呼出し元は、データベースに対して、少なくとも読取りアクセス権(ESB_PRIV_READ)を持っている必要があります。また、**EsbSetActive()**を使用して、そのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbArchiveBegin Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, ByVal FileName As String) As Long

Sub ESB_ArchiveBegin ()
  Dim sts As Long
  Dim AppName As String
  Dim DbName As String
  Dim FileName As String
  AppName = "Sample"
  DbName = "Basic"
  FileName = "Test.arc"
sts = EsbArchiveBegin (hCtx, AppName, DbName, FileName)
****
At this point, you can back up the server safely.
```

関連トピック

- EsbArchiveEnd
- EsbRestore
- EsbGetProcessState

EsbArchiveEnd

アーカイブが完了した後、サーバーが読取り書込みモードに戻されます。

構文

```
      EsbArchiveEnd
       (
      hCtx, AppName, DbName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
AppName	アーカイブされたアプリケーション名。
DbName	アーカイブされたデータベースの名前。

備考

EsbArchiveBegin()を呼び出した場合、後で **EsbArchiveEnd()**を呼び出して読取り/書込みモードへ戻す必要があります。

戻り値

なし。

アクセス

呼出し元は、データベースに対して、少なくとも読取りアクセス権(ESB_PRIV_READ)を持っている必要があります。また、**EsbSetActive()**を使用して、そのデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbArchiveEnd Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String) As Long

Sub ESB_ArchiveEnd()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 AppName = "Sample"
 DbName = "Basic"
 '**** Archive End ***
```

```
 sts = EsbArchiveEnd (hCtx, AppName, DbName)
End Sub
```

関連トピック

- EsbArchiveBegin
- EsbRestore

EsbAutoLogin

ユーザーが Essbase サーバーにログインするためのダイアログ・ボックスを表示します。オプションでアクティブなアプリケーションとデータベースを選択できます。

構文

```
      EsbAutoLogin (
      hInst, Server, User, Password, AppName, DbName, opt, pAccess, phCtx
      )
ByVal
      hInst
        As Long
ByVal
      Server
       As String
ByVal
      User
         As String
ByVal
      Password
      As String
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      opt
         As Integer

      pAccess
       As Integer

      phCtx
        As Long
```

パラメータ	説明
hInst	VB API インスタンス・ハンドル。

パラメータ	説明

Server	ネットワーク・サーバー名の文字列。
	サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
	`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
	保護モード(SSL)の場合、URL の構文は次のとおりです
	`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
	たとえば、
	`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

User	ユーザー名の文字列。
Password	パスワード文字列。
AppName	アプリケーション名。
DbName	データベース名。
Options	オプションのフラグ。値: ● ESB_AUTO_NODIALOG - (前述の引数の)デフォルト設定を使用して、ダイアログを表示しないでユーザーのログインを試みます。 ● ESB_AUTO_NOSELECT - ユーザーは、アプリケーションおよびデータベースを選択しないでもログインできます(ダイアログの下側の部分が表示されません)。 ESB_AUTO_NODIALOG と ESB_AUTO_NOSELECT の両方を指定すれば、ダイアログ・ボックスは表示されず、ユーザーはアプリケーションとデータベースを選択しなくてもログインできます: `ESB_AUTO_NODIALOG + ESB_AUTO_NOSELECT` ● ESB_AUTO_DEFAULT - ダイアログ・ボックスが表示され、ユーザーは対話形式でログインし、アプリケーションとデータベースを選択できます。
pAccess	データベース・アクセス・レベルを受け取る変数のアドレス。
phCtx	Essbase コンテキスト・ハンドルを受け取る変数のアドレス。

備考

- この関数によって戻されるダイアログ・ボックスは自動的に関数によって管理され、ログイン・ダイアログでのユーザー・パスワードの変更やデータベース・ノート・メッセージの表示などの機能を提供します。これにより、VB API を使用したすべてのアプリケーションで、標準化された、強力なログイン画面が使用できます。

- Windows 環境でプログラミングする場合は、この関数を EsbLogin 関数のかわりに使用します。
- この関数は、EsbInit が正しく実行された後で、かつコンテキスト・ハンドル引数を必要とするその他すべての VB API が呼び出される前に呼び出す必要があります。
- この関数は Windows 環境でのみサポートされます。UNIX 環境ではサポートされません。
- 文字列引数 Server、User、Password、AppName または DbName が含まれている必要があります。それらはオプションで空の文字列でもかまいません。いずれかが空の文字列でない場合、それらがポイントするバッファが、関数から戻される際に、ダイアログ・ボックスからユーザーによって選択された実績値で更新されます。渡された引数のいくつかが有効な文字列をポイントする場合、それらは、デフォルトでダイアログに表示される値として使用されます。これらの引数のバッファは、渡された値のみでなく、戻り値も含むことができる大きさである必要があります。
- ログインに成功すると、サーバー名とユーザー名が(ファイル ESSBASE.INI に)自動的に保管され、次にこの関数が呼び出される際にデフォルトとして使用されます(この引数が後続の呼出しに指定されていない場合)。正常に接続したすべてのサーバー名も保管され、表示されます。
- 「自動ログイン」ダイアログ・ボックスは、現在のアクティブなウィンドウ(フォーカスがあるウィンドウ)の子ウィンドウです。したがって、「自動ログイン」ダイアログが表示されている間は、アクティブなウィンドウを破棄したり、フォーカスを変更しないでください。
- ユーザーがダイアログ・ボックスで取消しボタンまたは[Esc]キーを押した場合、この関数は ESB_STS_CANCEL の値を戻します。
- Windows 環境では、エンド・ユーザーが「ヘルプ」ボタンをクリックすると、Essbase システムのログインに関するヘルプ・トピックが開きます。ESB_INIT_T 構造体の異なるヘルプ・ファイル名を指定すると、異なるヘルプ・ファイルをポイントするように「ヘルプ」ボタンを変更できます。

戻り値

正常終了の場合、phCtx の Essbase コンテキスト・ハンドルを戻します。それを、他の VB API 関数への後の呼出しで、引数として渡せます。また、pAccess で選択されたアプリケーションおよびデータベース(選択されている場合)へのユーザーのアクセス・レベルを戻します。

アクセス

この関数を呼び出す前に、EsbInit 関数を呼び出して、最初に VB API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

例

```
   Declare Function EsbAutoLogin Lib "ESBAPIN" (ByVal hInst As Long, ByVal Server As
String, _
                    ByVal User As String, ByVal Password As String, _
                    ByVal AppName As String, ByVal DbName As String, _
                    ByVal Opt As Integer, pAccess As Integer, _
```

```
                    phCtx As Long) As Long
Sub ESB_AutoLogin ()
 Dim sts As Long
 Dim Server As String * ESB_SVRNAMELEN
 Dim User As String * ESB_USERNAMELEN
 Dim Password As String * ESB_PASSWORDLEN
 Dim AppName As String * ESB_APPNAMELEN
 Dim DbName As String * ESB_DBNAMELEN
 Dim pOption As Integer
 Dim pAccess As Integer
 Dim hCtx As Long
 '**********************
 ' Initialize parameters
 '**********************
 Server = "Server"
 User = "User"
 Password = "Password"
 AppName = ""
 DbName = ""
 pOption = ESB_AUTO_DEFAULT
 '************************
 ' Login to Essbase Server
 '************************
 sts = EsbAutoLogin (hInst, Server, User, Password, AppName, DbName, pOption,
pAccess, hCtx)
End Sub
```

関連トピック

- EsbInit
- EsbListDatabases
- EsbLogin
- EsbLogout
- EsbSetActive

EsbBeginCalc

計算スクリプトの送信を開始し、オプションでアクティブ・データベースに対して計算スクリプトを実行します。その後 **EsbSendString()**を呼び出して計算スクリプトを送信し、最後に **EsbEndCalc()**を呼び出す必要があります。計算スクリプトの長さは、合計で 64KB 未満であることが必要です。計算を開始することも、または計算スクリプトの確認のみを行いエラーを戻すこともできます。

構文

```
     EsbBeginCalc
      (
     hCtx, isCalculate
     )
ByVal
     hCtx
         As Long
ByVal
```

```
      isCalculate
       As Integer
```

パラメータ　説明

hCtx　　VB API コンテキスト・ハンドル。

isCalculate 計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行されます。

備考

- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- Calculate フラグが FALSE に設定されている場合、データベースは計算スクリプトの構文チェックのみを行います。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbBeginCalc Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Calculate
As Integer) As Long

Sub ESB_BeginCalc ()
 Dim sts As Long
 Dim Script As String
 Dim Calculate As Integer
 Dim ProcState As ESB_PROCSTATE_T
 Script = "CALC ALL;"
 Calculate = ESB_YES
 '***********
 ' Begin Calc
 '***********
 sts = EsbBeginCalc (hCtx, Calculate)
 '***********************
 ' Send Calc script
 '***********************
 sts = EsbSendString (hCtx,Script)
 '*********
 ' End Calc
 '*********
 sts = EsbEndCalc (hCtx)
 '***********************************
 ' Check process state until it is done
 '***********************************
 sts = EsbGetProcessState (hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
```

```
   sts = EsbGetProcessState (hCtx, ProcState)
  Loop
End Sub
```

関連トピック

- EsbCalc
- EsbCalcFile
- EsbDefaultCalc
- EsbEndCalc
- EsbGetDefaultCalc
- EsbGetProcessState
- EsbSendString
- EsbSetDefaultCalc

EsbBeginDataload

アクティブ・データベースに更新指定の送信を開始し、更新用にロックされたデータ・ブロックのロックを解除できます。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。

構文

```
     Declare Function EsbBeginDataload Lib "esbapin" (
ByVal hCtx As Long,
ByVal isStore As Integer,
ByVal isUnlock As Integer,
ByVal isAbortOnError As Integer,
pRules As ESB_OBJDEF_T) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
Store	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
Unlock	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
abortOnError	TRUE の場合、最初のエラーでデータ・ロードが停止します。それ以外の場合は、データ・ロードを続行します。
pRules	ルール・ファイル・オブジェクト定義構造体へのポインタ。

備考

- **EsbBeginDataload()**を呼び出した後、**EsbSendString()**を 1 回以上呼び出して更新指定を送信し、最後に **EsbEndDataload()**を呼び出す必要があります。
- **EsbBeginDataload()**の後に呼び出した **EsbSendString()**へ渡される各文字列の末尾は、改行復帰文字シーケンス("\r\n")であることが必要です。

- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。
- 誤った入力行以降の入力行(レコード)を無視する **EsbBeginUpdate()**とは異なり、**EsbBeginDataload()**は残りの入力行も処理し、必要に応じてコミットします。

戻り値

なし。

アクセス

EsbBeginDataload()は、呼出し元がアクティブなデータベースに対する書込み権限(ESS_PRIV_WRITE)を持っていることを必要とします。

EsbBeginReport

アクティブなデータベースへのレポート指定の送信を開始します。この呼出しの後に、EsbSendString を呼び出してレポートの送信を行い、最後に EsbEndReport を呼び出す必要があります。レポート・データの出力を行うことも、確認のみ行い、エラーがあれば戻させることもできます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。

構文

```
      EsbBeginReport
       (
      hCtx, isOutput, isLock
      )
ByVal
      hCtx
        As Long
ByVal
      isOutput
       As Integer
ByVal
      isLock
        As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
isOutput	データの出力を制御します。TRUE の場合は、指定したレポートに従ってサーバーから出力されます。FALSE の場合は、データは出力されません。
isLock	ブロックのロックを制御します。TRUE の場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSE の場合は、ブロックのロックは行われません。

備考

- この関数に続いて、**EsbSendString()**を少なくとも 1 回呼び出し、その後**EsbEndReport()**を呼び出す必要があります。

- この関数によってデータが出力される場合(Output フラグが TRUE)、**EsbGetString()**を呼び出して戻されたデータを読み取ることができます。
- この関数によってブロックがロックされる場合(Lock フラグが TRUE)、呼出し元がロックされたブロックのロック解除を行う必要があります(たとえば、Unlock フラグを TRUE に設定して **EsbUpdate()**を呼び出します)。
- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbBeginReport Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Output
As Integer, ByVal Lock As Integer) As Long

Sub ESB_BeginReport ()
 Dim sts As Long
 Dim pOutput As Integer
 Dim pLock As Integer
 Dim Query As String
 Dim RString as String * 256
 Const szRString = 256
 Query = "<Desc Year !"
 Output = ESB_YES
 Lock = ESB_NO
 '*************
 ' Begin Report
 '*************
 sts = EsbBeginReport (hCtx, pOutput, pLock)  '*************************
 ' Send report specification
 '*************************
 sts = EsbSendString (hCtx, Query)
 '***********
 ' End Report
 '***********
 sts = EsbEndReport (hCtx)
 '**********************
 ' Print out all strings
 '**********************
 If sts = 0 Then
  sts = EsbGetString (hCtx, RString, szRString)
     Do While Mid$(RString, 1, 1) <> Chr$(0)
       Print RString
       sts = EsbGetString (hCtx, RString, szRString)
     Loop
 End If
End Sub
```

関連トピック

- EsbBeginUpdate
- EsbEndReport
- EsbGetString
- EsbReport
- EsbReportFile
- EsbSendString

EsbBeginUpdate

アクティブ・データベースに対して更新指定の送信を開始します。この呼出しの後に EsbSendString を呼び出して更新指定を送信し、最後に EsbEndUpdate を呼び出す必要があります。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

構文

```
      EsbBeginUpdate
       (
      hCtx, isStore, isUpdate
      )
ByVal
      hCtx
        As Long
ByVal
      isStore
       As Integer
ByVal
      isUpdate
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
isStore	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
isUpdate	データ・ブロックの更新を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。

備考

- この関数に続いて、**EsbSendString()**を少なくとも 1 回呼び出し、その後**EsbEndUpdate()**を呼び出す必要があります。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベースに対して、呼出し元が書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
   Declare Function EsbBeginUpdate Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Store As
Integer, ByVal Update As Integer) As Long

Sub ESB_BeginUpdate ()
 Dim sts As Long
 Dim Store As Integer
 Dim pUnlock As Integer
 Dim Query As String
 Query = "Year Market Scenario AcItemss Product 12345"
 Store = ESB_YES
 Unlock = ESB_NO
 '*************
 ' Begin Update
 '*************
 sts = EsbBeginUpdate (hCtx, Store, pUnlock)
 '**************************
 ' Send update specification
 '**************************
 sts = EsbSendString (hCtx, Query)
 '************
 ' End Update
 '************
 sts = EsbEndUpdate (hCtx)
End Sub
```

関連トピック

- EsbBeginReport
- EsbEndUpdate
- EsbSendString
- EsbUpdate
- EsbUpdateFile

EsbBuildDimension

データ・ファイルおよびルール・ファイルからのアクティブ・データベース内での次元の作成を可能にします。

構文

```
     EsbBuildDimension
      (
     hCtx, pRules, pData, pUser, ErrName
     )
ByVal
     hCtx
       As Long
```

```
      pRules
       As ESB_OBJDEF_T

      pData
        As ESB_OBJDEF_T

      pUser
        As ESB_MBRUSER_T
ByVal
      ErrName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
pRules	ルール・ファイル・オブジェクト定義構造体へのポインタ。
pData	データ・ファイル・オブジェクト定義構造体へのポインタ。
pUser	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
ErrName	ローカルに作成されるエラー出力ファイルの名前。

備考

- pMbrUser が NULL 以外の場合、SQL データ・ソースとみなされます。
- ルールまたはデータ・オブジェクト定義構造体の hCtx フィールドにサーバーの VB API コンテキストを指定した場合は、現在アクティブなアプリケーションおよびデータベースに対するオブジェクトが存在している必要があります。
- ルールまたはデータ・オブジェクト定義構造体の hCtx フィールドにローカルの VB API コンテキストを指定した場合は、この構造体の FileName を絶対パスにする必要があります。
- データ・ソースのインポートの詳細は、EsbImport の説明を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、指定されたデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbBuildDimension Lib "ESBAPIN" (ByVal hCtx As Long, Rules As
ESB_OBJDEF_T,
                        Data As ESB_OBJDEF_T, User As ESB_MBRUSER_T,
                        ByVal ErrName As String) As Long
Sub ESB_BuildDimension ()
 Dim sts    As Long
```

```
 Dim Rules   As ESB_OBJDEF_T
 Dim Data    As ESB_OBJDEF_T
 Dim User    As ESB_MBRUSER_T
 Dim ErrorName As String  '*********************************
 ' Rules file resides at the server
 '*********************************
 Rules.hCtx   = hCtx
 Rules.Type   = ESB_OBJTYPE_RULES
 Rules.FileName = "Test"        '*********************************
 ' Data file resides at the server
 '*********************************
 Data.hCtx    = hCtx
 Data.Type    = ESB_OBJTYPE_TEXT
 Data.FileName  = "Data"

 '*********************************
 ' Specify file to redirect errors
 ' to if any
 '*********************************
 ErrorName    = "BUILDDIM.ERR"  '*****************
 ' Build Dimensions
 '*****************
 sts      = EsbBuildDimension (hCtx, Rules, Data, User, ErrorName)
 '*****************************************************************************
 '*
 '* When a SQL data source is defined in the rules file, define
 '* the variables in the ESB_OBJDEF_T Data structure as follows:
 '*   Data.hCtx   = hCtx
 '*   Data.AppName  = ""
 '*   Data.DbName   = ""
 '*   Data.ObjType  = ESB_OBJTYPE_NONE
 '*   Data.FileName = ""
 '*
 '* Also, provide strings for the variables in the ESB_MBRUSER_T
 '* User structure; for example:
 '*   User.User   = "Dbusernm"
 '*   User.Password = "Dbpasswd"
 '*
 '* Use a blank string for User and Password, if the SQL source
 '* does not require user and password information; for example:
 '*   User.User   = ""
 '*   User.Password = ""
 '*
 '* Also, define sts as follows:
 '*   sts      = EsbBuildDimension (hCtx, Rules, Data, User, ErrorName)

 '*
 '***************************************************************************** End
Sub
```

関連トピック

- EsbImport
- EsbBuildDimFile
- EsbBuildDimStart

EsbBuildDimFile

アクティブなデータベース・アウトラインに対するメンバーの追加または削除に使用する、データ・ファイルを作成します。EsbBuildDimension を参照してください。

構文

```
      EsbBuildDimFile
       (
      hCtx, RulesObj, DataObj, MbrUser,
ErrorName, fOverwriteErrorFile
      )
ByVal
      hCtx
            As Long

      pRules
           As ESB_OBJDEF_T

      pData
            As ESB_OBJDEF_T

      pUser
           As ESB_MBRUSER_T
ByVal
      ErrName
          As String
ByVal
      ErrFileOverwrite
       As Integer
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
RulesObj	ルール・ファイル・オブジェクト定義構造体へのポインタ。
DataObj	データ・ファイル・オブジェクト定義構造体へのポインタ。
MbrUser	SQL ユーザー構造体(データ・ソースが SQL データベースの場合)。構造体が NULL の場合は、SQL 以外のデータ・ソースです。
ErrorName	クライアントでのエラー出力ファイルの名前。
fOverwriteErrorFile	この関数によって既存のファイル ErrorFile が上書きされるかどうかを示すブール値。

備考

- MbrUser が NULL 以外の場合、SQL データ・ソースとみなされます。
- EsbImport の説明に、データ・ソースのインポートに関する情報があります。
- データベースは、アクティブ・データベースである必要があります。EsbSetActive の説明を参照してください。

- EsbBuildDimStart は、**EsbBuildDimFile()**を使用する前に呼び出しておく必要があります。
- 再構築を行う前に繰り返し **EsbBuildDimFile()**を呼び出し、複数のルール・ファイルまたはデータ・ファイル(あるいはその両方)を使用してアウトラインにメンバーを追加できます。
- **EsbBuildDimFile()**の呼出しが完了した後、データベースを再構築する必要があります。
- 再構築後、アウトラインのロックを解除する必要があります。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、指定したデータベースに対してデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbBuild  Dimension Lib "ESBAPIN" (ByVal hCtx As Long, Rules As
ESB_OBJDEF_T, Data As ESB_OBJDEF_T, User As ESB_MBRUSER_T, ByVal ErrName As String)
As Long

Sub ESB_BuildDimFile()
 Dim sts As Long
 Dim Rules As ESB_OBJDEF_T
 Dim Data As ESB_OBJDEF_T
 Dim User As ESB_MBRUSER_T
 Dim ErrorName As String

 '*********************************
 ' Rules file resides at the server
 '*********************************
 Rules.hCtx = hCtx
 Rules.Type = ESB_OBJTYPE_RULES
 Rules.FileName = "Test"

 '*********************************
 ' Data file resides at the server
 '*********************************
 Data.hCtx = hCtx
 Data.Type = ESB_OBJTYPE_TEXT
 Data.FileName = "Data"
 '*********************************
 ' For a non SQL data source provide
 ' empty strings in User structure
 '*********************************
 User.User = ""
 User.Password = ""

 '*********************************
 ' Specify file to redirect errors
 ' to if any
 '*********************************
```

```
 ErrorName = "BUILDDIM.ERR"

 '*****************
 ' Build Dimensions
 '*****************

 sts = EsbBuildDimFile (hCtx, Rules, Data, User, ErrorName)
End Sub
```

関連トピック

- EsbImport
- EsbBuildDimension
- EsbBuildDimStart
- EsbOtlRestructure
- EsbUnlockObject

EsbBuildDimStart

アクティブ・データベース・アウトラインにメンバーを追加または削除するプロセスを開始します。

構文

```
      EsbBuildDimStart
       (
      hCtx
      );
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。

備考

- EsbImport の説明に、データ・ソースのインポートに関する情報があります。
- データベースは、アクティブ・データベースである必要があります。EsbSetActive の説明を参照してください。
- アウトライン・オブジェクトは **EsbBuildDimStart** を呼び出す前にロックする必要があります。EsbLockObject の説明を参照してください。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、指定したデータベースに対してデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Public Sub ESB_BuildDimStart()

 Dim sts As Long

 sts = EsbBuildDimStart (hCtx)

End Sub
```

関連トピック

- EsbImport
- EsbBuildDimension
- EsbBuildDimFile
- EsbLockObject

EsbCalc

アクティブ・データベースに対して計算スクリプトを単一の文字列として送信し、オプションで計算スクリプトを実行します。この関数は、EsbBeginCalc を呼び出した後に **EsbSendString()**を呼び出し、最後に **EsbEndCalc()**を呼び出すのと同じです。計算を開始することも、または計算スクリプトの確認のみを行いエラーを戻すこともできます。

構文

```
     EsbCalc
      (
     hCtx, isCalculate, cscQuery
     )
ByVal
     hCtx
        As Long
ByVal
     isCalculate
      As Integer
ByVal
     cscQuery
       As String
```

パラメータ　説明

hCtx	VB API コンテキスト・ハンドル。
isCalculate	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行されます。
cscQuery	単一の文字列としての計算スクリプト(64KB 未満)。

備考

- 計算スクリプトの文字列は、64KB 未満にする必要があります。

- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- Calculate フラグが FALSE に設定されている場合、データベースは計算スクリプトの構文チェックのみを行います。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbCalc Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Calculate As
Integer, ByVal Script As String) As Long

Sub ESB_Calc ()
 Dim sts As Long
 Dim Script As String
 Dim Calculate As Integer
 Dim ProcState As ESB_PROCSTATE_T  Script = "CALC ALL;"
 Calculate = ESB_YES  '**********
 ' Calculate
 '**********
 sts = EsbCalc (hCtx, Calculate, Script)  '************************************
 ' Check process state till it is done
 '************************************
 sts = EsbGetProcessState (hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState (hCtx, ProcState)
 Loop
End Sub
```

関連トピック

- EsbBeginCalc
- EsbCalcFile
- EsbDefaultCalc
- EsbEndCalc
- EsbGetDefaultCalc
- EsbGetProcessState
- EsbSendString
- EsbSetDefaultCalc

EsbCalcFile

ファイルからアクティブなデータベースに対して計算スクリプトを実行します。

構文

```
      EsbCalcFile
       (
      hDestCtx, hSrcCtx, AppName, DbName, FileName, isCalculate
      )
ByVal
      hDestCtx
        As Long
ByVal
      hSrcCtx
         As Long
ByVal
      AppName
        As String
ByVal
      DbName
         As String
ByVal
      FileName
        As String
ByVal
      isCalculate
       As Integer
```

パラメータ	説明
hDestCtx	サーバー上のターゲット・データベースの VB API コンテキスト・ハンドル。
hSrcCtx	計算スクリプト・ファイルの場所に対する VB API コンテキスト・ハンドル。計算スクリプト・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	計算スクリプト・ファイルの場所のアプリケーション名。
DbName	計算スクリプト・ファイルの場所のデータベース名。
FileName	計算スクリプト・ファイル名。
isCalculate	計算スクリプトの計算を制御します。TRUE の場合は、計算スクリプトが実行されます。

備考

- 計算スクリプトは、64KB 未満にする必要があります。
- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを定期的に確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbCalcFile Lib "ESBAPIN" (ByVal hDestCtx As Long, ByVal hSrcCtx
As Long, ByVal AppName As String, ByVal DbName As String, ByVal FileName As String,
ByVal isCalculate As Integer) As Long

Sub ESB_CalcFile ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FileName As String
 Dim Calculate As Integer
 Dim hSrcCtx As Long
 Dim ProcState As ESB_PROCSTATE_T  AppName = "Sample"
 DbName = "Basic"  '*************************************
 ' Calc script is an object at the server *
 '*************************************
 hSrcCtx = hCtx
 FileName = "calc"
 Calculate = ESB_YES '**********
 ' Calc File
 '**********
 sts = EsbCalcFile (hCtx, hSrcCtx, AppName,
 DbName, FileName, Calculate)  '***********************************
 ' Check process state till it is done
 '***********************************
 sts = EsbGetProcessState (hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState (hCtx, ProcState)
 Loop
End Sub
```

関連トピック

- EsbBeginCalc
- EsbCalc
- EsbDefaultCalc
- EsbSetDefaultCalcFile
- EsbGetProcessState

EsbCancelProcess

まだ完了していない非同期プロセスを取り消します。

構文

```
      EsbCancelProcess
       (
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

備考

- この関数を使用してプロセスを取り消した場合、一部のデータのみ再計算され、一貫性が失われた状態のままデータベースが残ることがあります。
- 非同期データベース操作(たとえば計算)が正しく開始された後以外にこの関数を呼び出すと、エラーが発生します。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbCancelProcess Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_CancelProcess ()
 Dim sts As Long
 Dim CalcScript As String
 Dim Calculate As Integer
 Dim ProcState As ESB_PROCSTATE_T
 Dim Items As Integer  CalcScript = "CALC ALL;"
 Calculate = ESB_YES  '***********
 ' Begin Calc
 '***********
 sts = EsbBeginCalc (hCtx, Calculate)
 '***********************
 ' Send Calc script
 ' It is possible to send
 ' more than one string
 '***********************
 sts = EsbSendString (hCtx, CalcScript)  '*********
 ' End Calc
 '*********
 sts = EsbEndCalc (hCtx)  '***********************************
 ' Check process state and cancel it if
 ' it takes too long
 '***********************************
 sts = EsbGetProcessState (hCtx, ProcState)
 Items = 1
  Do While ProcState.State = ESB_STATE_INPROGRESS
  Items = Items + 1
  If Items = 1000 Then    '***************
   ' Cancel process
   '***************
   sts = EsbCancelProcess (hCtx)
  End If
 Exit Do
   sts = EsbGetProcessState (hCtx, ProcState)
 Loop
```

```
End Sub
```

関連トピック

- EsbBeginCalc
- EsbCalc
- EsbGetProcessState
- EsbImport

EsbCheckAttributes

指定された各メンバーの属性情報を戻します。

構文

```
      EsbCheckAttributes
       (
      hCtx
      ,
      Count
      ,
      AttrNameArray()
      ,
      AttrTypeArray
      )
ByVal
      hCtx
           As Long
ByVal
      Count
           As Integer

      AttrNameArray()
       As String

      AttrTypeArray
        As Variant
```

パラメータ	説明
hCtx	コンテキスト・ハンドル
Count	指定した次元およびメンバーの数
AttrNameArray()	指定した次元およびメンバーの名前の配列

パラメータ	説明
AttrTypeArray	属性のタイプの配列に対する、次の定数識別子のいずれかになります: ● ESB_ATTRIBUTE_DIMENSION ● ESB_ATTRIBUTE_MEMBER ● ESB_STANDARD_DIMENSION ● ESB_STANDARD_MEMBER ● ESB_BASE_DIMENSION ● ESB_BASE_MEMBER ● ESB_ATTRIBUTED_MEMBER ● ESB_INVALID_MEMBER

備考

- 入力対象の名前のカウントおよびメンバー名のリストを予期します。
- 単一のメンバー名またはメンバー名の配列を受け入れ、属性のタイプの情報を入力した各メンバーに戻します。
- メンバー名は、属性次元名か属性メンバー名と、基本次元名か基本メンバー名のいずれかの組合せになります。

戻り値

正常終了の場合は sts = 0 が戻され、AttrTypeArray()の値が入力されます。無効なメンバー名が渡された場合は、エラーが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
     ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text
box.
Sub ESB_CheckAttributes()
Dim hCtx as long
Dim sts as long
Dim MbrNameArr() As String
Dim AttrTypeArr As Variant
Dim Count As Integer
Dim index As Integer
Dim test As Integer

 Count = InputBox("Enter the number of attribute members")
 ReDim MbrNameArr(Count)
 For index = 0 To Count - 1
    MbrNameArr(index) = InputBox("Enter attribute member name")
 Next index

 sts = EsbCheckAttributes(hCtx, Count, MbrNameArr, AttrTypeArr)
 If sts = 0 Then
    For index = LBound(AttrTypeArr) To UBound(AttrTypeArr)
   test = AttrTypeArr(index)
     Select Case test
       Case ESB_STANDARD_MEMBER
```

```
            Out MbrNameArr(index) & " is of type  ESB_STANDARD_MEMBER"
         Case ESB_STANDARD_DIMENSION
            Out MbrNameArr(index) & " is of type  ESB_STANDARD_DIMENSION"
         Case ESB_BASE_MEMBER
            Out MbrNameArr(index) & " is of type  ESB_BASE_MEMBER"
         Case ESB_BASE_DIMENSION
            Out MbrNameArr(index) & " is of type  ESB_BASE_DIMENSION"
         Case ESB_ATTRIBUTE_MEMBER
           Out MbrNameArr(index) & " is of type  ESB_ATTRIBUTE_MEMBER"
         Case ESB_ATTRIBUTE_DIMENSION
            Out MbrNameArr(index) & " is of type  ESB_ATTRIBUTE_DIMENSION"
         Case ESB_ATTRIBUTED_MEMBER
            Out MbrNameArr(index) & " is of type  ESB_ATTRIBUTED_MEMBER"
         Case Else
            Out MbrNameArr(index) & " is of INVALID Type or Invalid Member Name "
       End Select
     Next index
  Else
     Out "EsbCheckAttributes failed:" & sts: Exit Sub
  End If
End Sub
```

関連トピック

- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbCheckMemberName

文字列がアクティブ・データベース・アウトライン内で有効なメンバー名である
かどうか確認します。

構文

```
      EsbCheckMemberName
       (
      hCtx, MemName, isOk
      )
ByVal
      hCtx
        As Long
```

```
ByVal
      MemName
       As String

      isOk
        As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
MemName	確認されるメンバー名。
isOk	有効なメンバー・フラグを受け取る変数のアドレス。メンバーが有効な場合は TRUE に設定します。

戻り値

正常終了の場合、名前の文字列 MbrName がアクティブ・データベース・アウトラインの中で有効なメンバー名であるかどうかを示すフラグ pValid が戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbCheckMemberName Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
MbrName As String, isOk As Integer) As Long

Sub ESB_CheckMemberName ()
 Dim MbrName As String
 Dim Valid As Integer
 Dim sts As Long MbrName = "Year"
 '******************
 ' Check member name
 '******************
 sts = EsbCheckMemberName (hCtx, MbrName, Valid)
  if Valid = ESB_YES
   Print "Valid Member Name"
  End If
End Sub
```

関連トピック

- EsbGetMemberInfo
- EsbQueryDatabaseMembers
- EsbVerifyFilter
- EsbSetActive

EsbClearActive

ユーザーの現在のアクティブなアプリケーションおよびデータベースを消去します。

構文

```
      EsbClearActive

      (hCtx)

ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbClearActive Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_ClearActive ()
 Dim sts As Long  '**************
 ' Clear Active
 '**************
 sts = EsbClearActive (hCtx)
End Sub
```

関連トピック

- EsbGetActive
- EsbSetActive

EsbClearAliases

アクティブなデータベースのすべての別名テーブルを完全に削除します。

構文

```
      EsbClearAliases
       (
      hCtx
      )
ByVal
```

```
hCtx
 As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

備考

- 「デフォルト」または現在アクティブな別名テーブルは、削除できません。
- **EsbListConnections()**を呼び出して、別名テーブルを消去するデータベースと同じデータベースを他に使用している人がいないことを確認してください。
- この VB API 関数を使用する前に、**EsbSetAlias()**を使用してアクティブな別名を"デフォルト"に設定します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbClearAliases Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_ClearAliases ()
 Dim sts As Long  '***************
 ' Remove Aliases
 '***************
 sts = EsbClearAliases (hCtx)
End Sub
```

関連トピック

- EsbListAliases
- EsbRemoveAlias
- EsbSetAlias
- EsbListConnections
- EsbSetActive

EsbClearDatabase

アクティブ・データベース内にロードされているすべてのデータを消去します。

構文

```
EsbClearDatabase

(hCtx)
```

```
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

備考

この関数を使用して削除されたデータは復元できません。注意して使用してください!

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対して書込み権限(ESB_PRIV_WRITE)を持っており、**EsbSetActive()**を使用してデータベースをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbClearDatabase Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_ClearDatabase ()
 Dim sts As Long
 '***************
 ' Clear Database
 '***************
 sts = EsbClearDatabase (hCtx)
End Sub
```

関連トピック

- EsbDeleteDatabase
- EsbUnloadDatabase
- EsbSetActive

EsbClrSpanRelationalSource

Essbase に関係するデータが接続したリレーショナル・ストアに存在することを通知する、ブール bSpanRelPart フィールドを消去します。EsbQueryDatabaseMembers などのその他の API 関数の一部は、bSpanRelPart を読み込み、bSpanRelPart が設定されている場合はリレーショナル・ストアにアクセスします。

構文

```
     Declare Function EsbClrSpanRelationalSource Lib "esbapin" (ByVal hCtx As Long)
As Long
```

パラメータ 説明

hCtx API コンテキスト・ハンドル。

備考

一部の API 関数は、リレーショナル・ストアから情報を取得するように拡張されています。

- EsbQueryDatabaseMembers- リレーショナル・ストアからメンバー名を戻します。
- EsbGetMemberInfo- リレーショナル・ストア内のメンバーに関する情報を戻します。
- EsbCheckMemberName- リレーショナル・ストアで有効なメンバー名を確認します。
- EsbGetMemberCalc- 入力として渡されたリレーショナル・メンバーを認識し、すべてのリレーショナル・メンバーに対して NULL 文字列を戻します。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbClrRelationalSource Lib "ESBAPIN" (ByVal hCtx As Long) As
Long

 Sub ESB_ClrRelationalSource ()
  Dim sts As Long

  '**************************
  ' Clear the bSpanRelPart  field
  '**************************
  sts = EsbClrRelationalSource (hCtx)

 End Sub
```

関連トピック

- EsbSetSpanRelationalSource

EsbCommitDatabase

コミットは Essbase サーバーが自動的に処理するため、使用されません。この関数はエラー・メッセージ **ESB_STS_OBSOLETE** を戻します。データのコミットの詳細は、『Oracle Essbase データベース管理者ガイド』を参照してください。

EsbCopyApplication

クライアント上またはサーバー上の既存のアプリケーションを、関連するすべてのデータベースとオブジェクトも含めて、新規アプリケーションにコピーします。アプリケーションがサーバーにコピーされると、新しいアプリケーションが起動します。

構文

```
      EsbCopyApplication
      (
      hCtx, hSrcCtx, AppName, nAppName
      )
ByVal
      hCtx
        As Long
ByVal
      hSrcCtx
       As Long
ByVal
      AppName
       As String
ByVal
      nAppName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
hSrcCtx	使用されていません。hCtx と同じになります。
AppName	コピーする既存のアプリケーションの名前。
nAppName	新規アプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

- クライアント・アプリケーションをコピーすると、ローカル・アプリケーションのディレクトリとコンテンツもコピーされます。
- この関数は、クライアント・アプリケーションをクライアント上の新規アプリケーションにコピーする場合、またはサーバー・アプリケーションを同じサーバー上の新規アプリケーションにコピーする場合にのみ使用できます。**EsbCopyObject()**を使用して、異なるサーバー間でアプリケーションをコピーします。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESB_PRIV_APPCREATE)を持っている必要があります。

例

```
    Declare Function EsbCopyApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
hSrcCtx As Long, ByVal SrcApp As String, ByVal DestApp As String) As Long

Sub ESB_CopyApplication ()
 Dim sts As Long
 Dim SrcApp As String
 Dim DestApp As String
 Dim hSrcCtx As Long  hSrcCtx = hCtx
 SrcApp = "Sample"
 DestApp = "NewTest"  '*****************
 ' Copy Application
 '*****************
 sts = EsbCopyApplication (hCtx, hSrcCtx,
 SrcApp, DestApp)
End Sub
```

関連トピック

- EsbCopyDatabase
- EsbCopyObject

EsbCopyDatabase

クライアント上またはサーバー上の既存のデータベースを、関連するすべてのデータベースおよびオブジェクトも含めて、新規のデータベースにコピーします。サーバーにデータベースがコピーされた場合、新しいデータベースが起動されます。

構文

```
      EsbCopyDatabase
       (
      hCtx, hSrcCtx, AppName, nAppName, DbName, nDbName
      )
ByVal
      hCtx
        As Long
ByVal
      hSrcCtx
       As Long
ByVal
      AppName
       As String
ByVal
      nAppName
       As String
ByVal
      DbName
        As String
ByVal
      nDbName
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
hSrcCtx	使用されていません – hCtx と同じになります。
AppName	コピー元アプリケーションの名前。
nAppName	コピー先アプリケーションの名前。
DbName	コピーする既存のデータベースの名前。
nDbName	新規データベースの名前。1903 ページの「データベース名の制限」を参照してください。

備考

- クライアントのデータベースをコピーすると、ローカル・データベースのディレクトリとコンテンツもコピーされます。
- この関数は、クライアントのデータベースをクライアント上の新規データベースにコピーする場合、またはサーバーのデータベースを同じサーバー上の新規データベースにコピーする場合にのみ使用できます。異なるサーバーにデータベースをコピーする場合は、**EsbCopyObject()**を使用します。

戻り値

なし。

アクセス

サーバー・データベースの場合、呼出し元はデータベースの作成/削除/編集権限(ESB_PRIV_DBCREATE)を持っている必要があります。

例

```
    Declare Function EsbCopyDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal hSrcCtx
As Long, ByVal SrcApp As String, ByVal DestApp As String, ByVal SrcDb As String,
ByVal DestDb As String) As Long

Sub ESB_CopyDatabase ()
 Dim sts As Long
 Dim SrcApp As String
 Dim DestApp As String
 Dim SrcDb As String
 Dim DestDb As String
 Dim hSrcCtx As Long  hSrcCtx = hCtx
 SrcApp = "Sample"
 DestApp = "NewSamp"
 SrcDb = "Basic"
 DestDb = "NewBasic"  '***************
 ' Copy database
 '***************  sts = EsbCopyDatabase (hCtx, hSrcCtx, SrcApp,
 DestApp, SrcDb, DestDb)
End Sub
```

関連トピック

- EsbCopyApplication
- EsbCopyObject

EsbCopyFilter

既存のフィルタをコピーします。

構文

```
      EsbCopyFilter
       (
      hCtx, hSrcCtx, AppName, nAppName, DbName, nDbName, FltName, nFltName
      )
ByVal
      hCtx
        As Long
ByVal
      hSrcCtx
       As Long
ByVal
      AppName
       As String
ByVal
      nAppName
       As String
ByVal
      DbName
        As String
ByVal
      nDbName
       As String
ByVal
      FltName
       As String
ByVal
      nFltName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

hSrcCtx　　使用されていません – hCtx と同じになります。

AppName　　コピー元アプリケーション名。

nAppName　　コピー先アプリケーション名。

DbName　　コピー元データベース名。

nDbName　　宛先データベース名。

FltName　　コピーする既存のフィルタのコピー元での名前。

パラメータ 説明

nFltName　コピーしたフィルタのコピー先での名前。1904 ページの「フィルタ名の制限」を参照してください。

備考

- ソース・フィルタが存在している必要があります。
- 既存のフィルタを誤って上書きするのを防ぐため、呼出し元はコピー先フィルタが存在しているかどうかを確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbCopyFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal hSrcCtx
As Long, ByVal SrcApp As String, ByVal DestApp As String, ByVal SrcDb As String,
ByVal DestDb As String, ByVal SrcName As String, ByVal DestName As String) As Long

Sub ESB_CopyFilter ()
 Dim sts As Long
 Dim SrcApp As String
 Dim SrcDb As String
 Dim SrcName As String
 Dim DestApp As String
 Dim DestDb As String
 Dim DestName As String
 Dim hDestCtx As Long  hDestCtx = hCtx
 SrcApp = "Sample"
 SrcDb = "Basic"
 SrcName = "Filter"
 DestApp = "NewSamp"
 DestDb = "NewBasic"
 DestName = "NewFilter"  '************
 ' Copy Filter
 '************
 sts = EsbCopyFilter (hCtx, hDestCtx, SrcApp,
  DestApp, SrcDb, DestDb, SrcName, DestName)
End Sub
```

関連トピック

- EsbDeleteFilter
- EsbListFilters
- EsbRenameFilter
- EsbSetFilter

EsbCopyObject

サーバーまたはクライアントのオブジェクト・システムのオブジェクトをコピーします。

構文

```
      EsbCopyObject
       (
      hCtx, hDestCtx, ObjType, AppName, nAppName, DbName, nDbName,
objName, nobjName
      )
ByVal
      hCtx
        As Long
ByVal
      hDestCtx
       As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      nAppName
       As String
ByVal
      DbName
        As String
ByVal
      nDbName
       As String
ByVal
      objName
       As String
ByVal
      nObjName
       As String
```

パラメータ 説明

hCtx　コピー元オブジェクトの VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。

hDestCtx　コピー先オブジェクトの VB API コンテキスト・ハンドル。

ObjType　オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。

AppName　コピー元アプリケーション名。

nAppName　コピー先アプリケーション名。

DbName　コピー元データベース名。空の文字列の場合は、コピー元のアプリケーションのサブディレクトリを使用します。

パラメータ 説明

nDbName　宛先データベース名。空の文字列の場合は、コピー先のアプリケーションのサブディレクトリを使用します。

objName　コピー元のオブジェクト名。

nobjName　コピー先のオブジェクトの名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- オブジェクトはクライアントからサーバーへ、サーバーからクライアントへ、同一サーバー内で、または異なるサーバー間でコピーできます。いずれの場合も、コピー先のオブジェクトは存在していない、または呼出し元がロックしている必要があります。
- アウトライン・オブジェクトはコピーできません。関連するアウトラインも含めてデータベースをコピーする場合は、**EsbCopyDatabase()**関数を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元は、オブジェクトがある指定したコピー元アプリケーションまたはデータベース(あるいはその両方)に対して、適切なレベルのアクセス権(オブジェクト・タイプによる)を持っている必要があります。さらに呼出し元は、指定したコピー先アプリケーションまたはデータベースに対して、アプリケーションデザイン権限またはデータベースデザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbCopyObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal hDestCtx
As Long, ByVal ObjType As Integer, ByVal SrcApp As String, ByVal DestApp As String,
ByVal SrcDb As String, ByVal DestDb As String, ByVal SrcObj As String, ByVal DestName
As String) As Long

Sub ESB_CopyObject ()
 Dim sts As Long
 Dim hDestCtx As Long
 Dim SrcApp As String
 Dim SrcDb As String
 Dim SrcObj As String
 Dim DestApp As String
 Dim DestDb As String
 Dim DestObj As String
 Dim ObjType As Integer  hDestCtx = hCtx
 SrcApp = "Sample"
 SrcDb = "Basic"
 SrcObj = "Basic"
 DestApp = "NewSamp"
 DestDb = "NewBasic"
```

```
 DestObj = "NewBasic"
 ObjType = ESB_OBJTYPE_RULES  '******************
 ' Copy rules object
 '******************
 sts = EsbCopyObject (hCtx, hDestCtx, ObjType,
 SrcApp, DestApp, SrcDb, DestDb, SrcObj,
 DestObj)
End Sub
```

関連トピック

- EsbCreateObject
- EsbDeleteObject
- EsbListObjects
- EsbRenameObject
- EsbLockObject

EsbCreateApplication

クライアントまたはサーバー上で、新規アプリケーションを作成します。アプリケーションがサーバーで作成された場合は、起動も行われます。

構文

```
      EsbCreateApplication
       (
      hCtx, AppName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。**EsbCreateDatabase()**または **EsbCreateApplication()**を呼び出した後で **EsbSetActive()**を呼び出し、**EsbRestructure()**などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションやデータベース)に対して実行されないようにします。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/
編集権限(ESB_PRIV_APPCREATE)を持っている必要があります。

例

```
    Declare Function EsbCreateApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String) As Long

Sub ESB_CreateApplication ()
 Dim sts As Long
 Dim AppName As String  AppName = "Sample"  '********************
 ' Create Application
 '********************
 sts = EsbCreateApplication (hCtx, AppName)
End Sub
```

関連トピック

- EsbCreateDatabase
- EsbCreateObject

EsbCreateDatabase

クライアントまたはサーバー上で、アプリケーション内に新規データベースを作
成します。データベースがサーバー上で作成された場合、起動も行われます。

構文

```
      EsbCreateDatabase
       (
      hCtx, AppName, DbName, DbType
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      DbType
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

パラメータ	説明
AppName	データベースを含むアプリケーションの名前。
DbName	作成するデータベースの名前。1903 ページの「データベース名の制限」を参照してください。
DbType	作成するデータベースのタイプ: (ESB_DBTYPE_NORMAL/ESB_DBTYPE_CURRENCY)。

備考

- クライアント・データベースを作成すると、ローカル・データベース・ファイルを含むディレクトリが作成されます。
- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。**EsbCreateDatabase**()または **EsbCreateApplication**()を呼び出した後で **EsbSetActive**()を呼び出し、**EsbRestructure**()などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションやデータベース)に対して実行されないようにします。

戻り値

なし。

アクセス

サーバー・データベースの場合、呼出し元はデータベースの作成/削除/編集権限(ESB_PRIV_DBCREATE)を持っている必要があります。

例

```
    Declare Function EsbCreateDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, ByVal DbType As Integer) As Long

Sub ESB_CreateDatabase ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String  AppName = "Sample"
 DbName = "Basic"  '*****************
 ' Create database
 '*****************
 sts = EsbCreateDatabase (hCtx, AppName, DbName,
 ESB_DBTYPE_NORMAL)
End Sub
```

関連トピック

- EsbCreateApplication
- EsbCreateObject

EsbCreateDrillThruURL

アクティブなデータベース・アウトライン内に、指定されたリンクと名前を使用してドリルスルー URL を作成します。

1904 ページの「ドリルスルー URL の制限」.

構文

```
     Declare Function EsbCreateDrillThruURL Lib "esbapin" (ByVal hCtx As Long, ByRef
symRegions() As String, ByRef pUrl As ESB_DURLINFO_T) As Long
```

パラメータ	説明
hCtx	Visual Basic API のコンテキスト・ハンドル
symRegions()	対称領域の指定を含む配列
pUrl	URL 定義

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内にドリルスルー URL が作成されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESB_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
    Sub ESB_CreateGLDrillThru()
 Dim sts           As Long
 Dim url           As ESB_DURLINFO_T
 Dim cppDrillRegions(0 To 1)  As String

 '*************************************************************
 ' Need to create a local context, if files are not on the server
 '*************************************************************
 url.bIsLevel0 = 0

 cppDrillRegions(0) = "sales"
 cppDrillRegions(1) = "cogs"
 url.cpURLXML = "<?xml version="1.0" encoding="UTF-8"?>
<foldercontents path="/">
 <resource name="Assets Drill through GL" description="" type="application/x-hyperion-
applicationbuilder-report">
  <name xml:lang="fr">Rapport de ventes</name>
  <name xml:lang="es">Informe de ventas</name>
  <action name="Display HTML" description="Launch HTML display of Content"
shortdesc="HTML">
   <url>/fusionapp/Assetsdrill.jsp?$SSO_TOKEN$&$CONTEXT$&$ATTR(ds,pos,gen,level.edge)
$
   </url>
  </action>
 </resource>
</foldercontents>
"
```

```
 url.cpURLName = "VB URL7"
 url.iURLXMLSize = 512

 sts = EsbCreateDrillThruURL(hCtx, cppDrillRegions, url)

 Debug.Print "EsbCreateDrillThruURL sts: " & sts
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbCreateExtUser

新規の外部認証ユーザーを作成します。

構文

```
     Declare Function EsbCreateExtUser Lib "esbapin" (
ByVal hCtx As Long,
ByVal UserName As String,
ByVal Password As String,
ByVal Protocol As String,
ByVal Connparam As String) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
UserName	作成するユーザーの名前。1904 ページの「ユーザー名の制限」を参照してください。
Password	新規ユーザーのセキュリティ・パスワード。1904 ページの「パスワードの制限」を参照してください。
SecurityProvider	外部認証メカニズムの名前。
ProviderConnectionParameters	外部認証メカニズムで使用されるパラメータ(ある場合)。

備考

- 指定したユーザーが存在していないことが必要です。
- ユーザーのアクセス・レベルおよびその他のパラメータは、**EsbSetUser()**関数を使用して設定します。
- この関数を呼び出す前に、パスワードが正しく入力されたことをプログラムで確認する必要があります; たとえば、パスワードを 2 回入力するようにします。一度入力したパスワードは取得できません。ただし、**EsbSetPassword()**関数を使用すればパスワードを変更できます。
- Password パラメータは、Shared Services を変更することで重複可能です。このパラメータには空の文字列を使用できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

EsbCreateFilter

新規フィルタを作成し、そのコンテンツの設定を開始します。

構文

```
     Declare Function EsbCreateFilter Lib "esbapin" (
ByVal hCtx As Long,
ByVal AppName As String,
ByVal DbName As String,
ByVal FltName As String,
ByVal isActive As Integer,
ByVal pAccess As Integer) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル
AppName	アプリケーション名
DbName	データベース名
FilterName	フィルタ名。1904 ページの「フィルタ名の制限」を参照してください。
Active	フィルタのアクティブ・フラグ。TRUE の場合はフィルタがアクティブに設定され、TRUE でない場合は非アクティブに設定されます。
Access	デフォルトのフィルタ・アクセス・レベル

備考

- フィルタが存在しない場合は、この呼出しによってフィルタが作成されます。
- フィルタがすでに存在する場合は、エラー・メッセージが戻されます。
- この呼出しの後に **EsbSetFilterRow()**を続けて呼び出して、フィルタのすべての行を設定する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が、指定したデータベースに対するデータベース・マネージャ権限(ESS_PRIV_DBDESIGN)を持っている必要があります。

EsbCreateGroup

グループを新規作成します。

構文

```
      EsbCreateGroup
       (
      hCtx, GrpName
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

GrpName　　作成するグループの名前。1904 ページの「グループ名の制限」を参照してください。

備考

- 指定したグループが存在していないことが必要です。
- グループのアクセス権のレベルは、**EsbSetGroup()**関数を使用して設定します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbCreateGroup Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String) As Long

Sub ESB_CreateGroup ()
 Dim sts As Long
 Dim GroupName As String  GroupName = "PowerUsers"  '*************
 ' Create Group
 '*************
 sts = EsbCreateGroup (hCtx, GroupName)
End Sub
```

関連トピック

- EsbDeleteGroup
- EsbListGroups
- EsbRenameGroup
- EsbSetGroup

EsbCreateLocalContext

ローカルの VB API 操作で使用するローカル VB API コンテキストを作成します。

構文

```
      EsbCreateLocalContext
       (
      hInst, User, Password, phCtx
      )
ByVal
      hInst
        As Long
ByVal
      User
         As String
ByVal
      Password
       As String

      phCtx
        As Long
```

パラメータ 説明

hInst	VB API インスタンス・ハンドル。
User	現在使用されていません。空の文字列になります。
Password	現在使用されていません。空の文字列になります。
phCtx	Essbase サーバー・ローカル・コンテキスト・ハンドルを受け取る変数のアドレス。

備考

- この関数はローカル VB API 操作(ローカル・ファイル/オブジェクト関数など)へのアクセスを必要とする場合に呼び出す必要があります。**EsbInit()**を呼び出した後に呼び出します。
- 関数はクライアント・アプリケーションにつき 1 回呼び出すのみです。コンテキスト・ハンドルは、ローカルの API 操作すべてに使用できます。
- アプリケーションがローカル・オブジェクトへのアクセスを終了した時点で、**EsbDeleteLocalContext()**を呼び出す必要があります。

戻り値

正常終了の場合は、有効なローカル・コンテキスト・ハンドルが phLocalCtx に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbCreateLocalContext Lib "ESBAPIN" (ByVal hInst As Long, ByVal
```

```
User As String, ByVal Password As String, hCtx As Long) As Long

Sub ESB_CreateLocalContext ()
 Dim sts As Long
 Dim User As String
 Dim Password As String
 Dim hCtx As Long  '*********************
 ' Create Local Context
 '*********************
 sts = EsbCreateLocalContext (hInst, User, Password, hCtx)
End Sub
```

関連トピック

- EsbDeleteLocalContext
- EsbInit

EsbCreateLocationAlias

新規にロケーション別名を作成します。つまり、別名の文字列を、次の 5 個の文字列が順に並んだ形式へマッピングします。5 個の文字列とは: ホスト名、アプリケーション名、データベース名、ユーザー・ログイン名、ユーザー・パスワードです。

構文

```
     EsbCreateLocationAlias
      (
     hCtx
     ,
     AliasName
     ,
     HostName
     , _

     AppName
     ,
     DbName
     ,
     Login
     ,
     Password
     )
ByVal
     hCtx
        As Long
ByVal
     AliasName
      As String
ByVal
     HostName
       As String
ByVal
     AppName
```

```
       As String
ByVal
      DbName
        As String
ByVal
      Login
         As String
ByVal
      Password
       As String
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル

AliasName　ロケーション別名

HostName　ターゲットのホスト

AppName　ターゲットのアプリケーション

DbName　　ターゲットのデータベース

Login　　ユーザー・ログイン名

Password　ユーザー・パスワード

戻り値

AliasName と同じ名前のロケーション別名がすでに存在する場合は、エラーが戻されます。

例

```
    Public Sub LocationAliasTest()

 Dim status   As Long
 Dim ListCount As Integer
 Dim Aliases  As Variant
 Dim HostNames As Variant
 Dim AppNames  As Variant
 Dim DbNames  As Variant
 Dim UserNames As Variant

 status = EsbCreateLocationAlias(hCtx, "blah1", "LocalHost", "Demo", "Basic", _
              "admin", "password")
 If (status <> 0) Then
  MsgBox "Create routine Failed"
  Exit Sub
 End If

 status = EsbCreateLocationAlias(hCtx, "blah2", "LocalHost", "Demo", "Basic", _
               "admin", "password")
 If (status <> 0) Then
  MsgBox "Create routine Failed"
  Exit Sub
 End If
```

```
 status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
                AppNames, DbNames, UserNames)
 If (status <> 0) Then
  MsgBox "Get routine Failed"
  Exit Sub
 End If

 If (ListCount > 0) Then
    ' Retrieve the elements as Aliases(0) to Aliases(ListCount -1)
 End If

 status = EsbDeleteLocationAlias(hCtx, "blah1")
 If (status <> 0) Then
  MsgBox "Delete routine Failed"
  Exit Sub
 End If

 status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
                AppNames, DbNames, UserNames)
 If (status <> 0) Then
  MsgBox "Get routine Failed"
  Exit Sub
 End If
End Sub
```

関連トピック

- EsbDeleteLocationAlias
- EsbGetLocationAliasList

EsbCreateObject

サーバーまたはクライアントのオブジェクト・システムで、オブジェクトを新規作成します。

構文

```
      EsbCreateObject
       (
      hCtx, ObjType, AppName, DbName, ObjName
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
```

```
ByVal
      ObjName
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。**EsbCreateLocalContext()**によって戻されるローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	作成するオブジェクトの名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- 作成するオブジェクトが存在していないことが必要です。
- サーバー上で新規作成されたオブジェクトにはデータが含まれておらず、単なるプレースホルダとして機能し、他のユーザーによるオブジェクトの作成を防止します。作成されたオブジェクトを更新する場合は、**EsbLockObject()**を使用してロックし、**EsbPutObject()**を使用して保存する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元は、指定したアプリケーションまたはオブジェクトの保存先データベースに対して、アプリケーション・デザイン権限またはデータベース・デザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbCreateObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String)
As Long

Sub ESB_CreateObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
 ObjType = ESB_OBJTYPE_RULES  '********************
 ' Create Rules Object
 '********************
```

```
  sts = EsbCreateObject (hCtx, ObjType, AppName, DbName, ObjName)
End Sub
```

関連トピック

- EsbDeleteObject
- EsbListObjects
- EsbLockObject
- EsbPutObject
- EsbCopyObject
- EsbRenameObject

EsbCreateStorageTypedApplication

ブロック(多次元)または集約のいずれかのデータ・ストレージ・モード・オプションで、新規アプリケーションを作成します。

構文

```
      EsbCreateStorageTypedApplication
       (
      hCtx, AppName, StorageType
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      StorageType
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
AppName	作成するアプリケーションの名前。1903 ページの「アプリケーション名の制限」を参照してください。
StorageType	新しいアプリケーションのデータ・ストレージ・タイプ。 StorageType に対して有効な値は、次のとおりです: ● ESB_DEFAULT_DATA_STORAGE ● ESB_MULTIDIM_DATA_STORAGE - ブロック・ストレージ(多次元)で、デフォルトのストレージ・タイプです ● ESB_ASO_DATA_STORAGE - 集約ストレージ

備考

- 新しいアプリケーションは、非 Unicode モードで作成されます。
- クライアント・アプリケーションを作成すると、ローカル・アプリケーション・ファイルを含むディレクトリが作成されます。

- 新規作成されたデータベースやアプリケーションは自動的にアクティブに設定されません。**EsbCreateDatabase()**、**EsbCreateApplication()**または**EsbCreateStorageTypedApplication()**を呼び出した後で、**EsbSetActive()**を呼び出し、**EsbRestructure()**などの以降の関数が間違ったデータベースやアプリケーション(アクティブなアプリケーションやデータベース)に対して実行されないようにします。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESB_PRIV_APPCREATE)を持っている必要があります。

例

```
    Declare Function EsbCreateStorageTypedApplication Lib "ESBAPIN" (ByVal hCtx As
Long, ByVal AppName As String, ByVal StorageType As Integer) As Long

Sub ESB_CreateStorageTypedApplication ()
 Dim sts As Long
 Dim AppName As String
 Dim StorageType as Integer  AppName = "Sample"  '********************
 ' Create Storage Typed Application
 '********************
 sts = EsbCreateStorageTypedApplication (hCtx, AppName, StorageType)
End Sub
```

関連トピック

- EsbCreateApplication
- EsbCreateDatabase
- EsbCreateObject
- EsbGetApplicationInfo

EsbCreateUser

新規ユーザーを作成します。

構文

```
     EsbCreateUser
      (
     hCtx, userName, Password
     )
ByVal
     hCtx
       As Long
ByVal
     userName
      As String
ByVal
```

```
    Password
     As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
userName	作成するユーザーの名前。1904 ページの「ユーザー名の制限」を参照してください。
Password	新規ユーザーのセキュリティ・パスワード。1904 ページの「パスワードの制限」を参照してください。

備考

- 指定したユーザーが存在していないことが必要です。
- ユーザーのアクセス・レベルおよびその他のパラメータは、**EsbSetUser()**関数を使用して設定します。
- この関数を呼び出す前に、パスワードが正しく入力されたことをプログラムで確認する必要があります(パスワードを 2 回入力するなど)。一度入力したパスワードは取得できません。ただし、**EsbSetPassword()**関数を使用すればパスワードを変更できます。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbCreateUser Lib "ESBAPIN" (ByVal hCtx As Long, ByVal User As
String, ByVal Password As String) As Long

Sub ESB_CreateUser ()
 Dim sts As Long
 Dim User As String
 Dim Password As String  User = "Joseph"
 Password = "Password"  '************
 ' Create user
 '************
 sts = EsbCreateUser (hCtx, User, Password)
End Sub
```

関連トピック

- EsbDeleteUser
- EsbListUsers
- EsbRenameUser
- EsbSetPassword
- EsbSetUser

EsbCreateVariable

代替変数を新規作成、または同一のサーバー値、アプリケーション値およびデータベース値を持つ変数名がすでに存在している場合には既存の代替変数を変更します。

構文

```
      EsbCreateVariable
       (
      hCtx, pVariable
      )
ByVal
      hCtx
         As Long

      pVariable
       As ESB_PVARIABLE_T
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

pVariable　作成された代替変数の説明を含む構造体を指すポインタ。

備考

- 変数範囲はサーバー、アプリケーション、またはデータベースに適用できます。範囲は 1300 ページの「ESB_VARIABLE_T」構造体で制御されます。サーバー、アプリケーション、データベースがすべて指定されている場合は、代替変数は指定されたデータベースにのみ適用されます。サーバーとアプリケーションのみが指定されている場合、代替変数は指定されたアプリケーション内のすべてのデータベースに適用されます。サーバーのみが指定されている場合、代替変数は指定されたサーバーのすべてのアプリケーションとデータベースに適用されます。
- 新規変数が既存の変数と同じ名前および適用範囲で作成された場合、Esbbase からエラー・メッセージは戻されずに新しい値で古い値が置換されます。
- 指定した 1 台のサーバー上では、同じ名前で適用範囲(アプリケーションとデータベース)の異なる複数の代替変数を作成できます。

戻り値

成功の場合、ゼロが戻されます。

例

```
    Declare Function EsbCreateVariable Lib "esbapin" (ByVal hCtx As Long, pVariable
As ESB_VARIABLE_T) As Long

Sub Esb_CreateVariable()

Dim sts As Long
Dim oVariable As ESB_VARIABLE_T
```

```
'  Create "QuarterName" Susbtitution Variable at the Sample application level
  oVariable.Server = "Localhost"
  oVariable.AppName = "Sample"
'  ** Note that DbName has been left empty
  oVariable.VarName = "QuarterName"
  oVariable.VarValue = "Qtr1"

  sts = EsbCreateVariable(hCtx, oVariable)

End Sub
```

関連トピック

- 1300 ページの「ESB_VARIABLE_T」
- EsbDeleteVariable
- EsbGetVariable
- EsbListVariables

EsbDefaultCalc

アクティブ・データベースのデフォルト計算を実行します。

構文

```
      EsbDefaultCalc
       (
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。

備考

- この関数が正しく実行され、計算を開始すると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを定期的に確認する必要があります。
- デフォルト計算スクリプトを取得および設定するには、関数 **EsbGetDefaultCalc()**、**EsbSetDefaultCalc()**および **EsbSetDefaultCalcFile()**を使用します。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbDefaultCalc Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_DefaultCalc ()
 Dim sts As Long
 Dim ProcState As ESB_PROCSTATE_T  '************************
 ' Run default calc script
 '************************
 sts = EsbDefaultCalc (hCtx)  '***********************************
 ' Check process state till it is done
 '***********************************
 sts = EsbGetProcessState (hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState (hCtx, ProcState)
 Loop
End Sub
```

関連トピック

- EsbBeginCalc
- EsbCalc
- EsbGetDefaultCalc
- EsbSetDefaultCalc
- EsbSetDefaultCalcFile

EsbDeleteApplication

クライアント上またはサーバー上で、既存のアプリケーションを削除します。アプリケーションがサーバー上で実行されている場合、最初に停止されます。

構文

```
      EsbDeleteApplication
       (
      hCtx, AppName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

パラメータ 説明

AppName　削除するアプリケーションの名前。

備考

- クライアント・アプリケーションを削除すると、ローカルのアプリケーションのディレクトリとコンテンツが削除されます。アプリケーションで保管されているすべてのオブジェクトも、すべてのデータベースも含めて削除されます。
- サーバー・アプリケーションを削除するには、接続しているユーザーはアプリケーションの作成/削除権限を持っている必要があります。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESB_PRIV_APPCREATE)を持っている必要があります。

例

```
    Declare Function EsbDeleteApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String) As Long

Sub ESB_DeleteApplication ()
 Dim sts As Long
 Dim AppName As String   AppName = "Sample"
 '********************
 ' Delete Application
 '********************
 sts = EsbDeleteApplication (hCtx, AppName)
End Sub
```

関連トピック

- EsbDeleteDatabase
- EsbDeleteObject

EsbDeleteDatabase

クライアントまたはサーバー上で、アプリケーションから既存のデータベースを削除します。データベースがサーバー上で実行されている場合、最初に停止されます。

構文

```
     EsbDeleteDatabase
      (
     hCtx, AppName, DbName
     )
ByVal
```

```
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	データベースを含むアプリケーション名。
DbName	削除するデータベースの名前。

備考

- クライアント・データベースを削除すると、ローカルのデータベースのディレクトリとコンテンツも削除されます。
- サーバー・データベースを削除すると、そのデータベースに関連付けられているすべてのオブジェクトも削除されます。

戻り値

なし。

アクセス

サーバー・データベースの場合、呼出し元はデータベースの作成/削除/編集権限(ESB_PRIV_DBCREATE)を持っている必要があります。

例

```
    Declare Function EsbDeleteDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String) As Long

Sub ESB_DeleteDatabase ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String  AppName = "Sample"
 DbName = "Basic"
 '****************
 ' Delete database
 '****************
 sts = EsbDeleteDatabase (hCtx, AppName, DbName)
End Sub
```

関連トピック

- EsbDeleteApplication
- EsbDeleteObject

EsbDeleteDrillThruURL

アクティブなデータベース・アウトライン内で、指定された URL 名のドリルスルー URL を削除します。

構文

```
    Declare Function EsbDeleteDrillThruURL Lib "esbapin" (ByVal hCtx As Long, ByVal
URLName As String) As Long
```

パラメータ	説明
hCtx	Visual Basic API のコンテキスト・ハンドル
URLName	ドリルスルー URL 名

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内の指定されたドリルスルー URL が削除されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESB_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
    Sub ESB_DeleteGLDrillThru()
 Dim URLName    As String

 URLName = "VB URL7"
 sts = EsbDeleteDrillThruURL(hCtx, URLName)

 Debug.Print "EsbDeleteDrillThruURL sts: " & sts
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbDeleteFilter

既存のフィルタを削除します。

構文

```
    EsbDeleteFilter
     (
    hCtx, AppName, DbName, FltName
```

```
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FltName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　データベース名。

FltName　　フィルタ名。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbDeleteFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, ByVal FltName As String) As Long

Sub ESB_DeleteFilter ()
  Dim sts As Long
  Dim AppName As String
  Dim DbName As String
  Dim FilterName As String  AppName = "Sample"
  DbName = "Basic"
  FilterName = "Filter"  '**************
  ' Delete Filter
  '**************
  sts = EsbDeleteFilter (hCtx, AppName, DbName,
  FilterName)
End Sub
```

関連トピック

- EsbCopyFilter
- EsbListFilters
- EsbRenameFilter

- EsbSetFilter

EsbDeleteFromGroup

グループ・メンバーのリストからユーザーを削除します。

構文

```
      EsbDeleteFromGroup
       (
      hCtx, GrpName, User
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
ByVal
      User
        As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
GrpName	グループ名。
User	グループ・リストから削除するユーザー名。

備考

指定したグループのメンバー・リストから指定したユーザーを削除するのみでなく、この関数は削除するユーザーの関連グループのリストから指定したグループも削除します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbDeleteFromGroup Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String, ByVal User As String) As Long

Sub ESB_DeleteFromGroup ()
  Dim sts As Long
  Dim GroupName As String
  Dim User As String  GroupName = "PowerUsers"
  User = "Jim Smith"
```

```
  '***********************
  ' Delete user from group
  '***********************
 sts = EsbDeleteFromGroup (hCtx, GroupName, User)
End Sub
```

関連トピック

- EsbAddToGroup
- EsbGetGroupList
- EsbListGroups
- EsbSetGroupList

EsbDeleteGroup

既存のグループを削除します。

構文

```
      EsbDeleteGroup
       (
      hCtx, GrName
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

GrpName　　削除するグループの名前。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbDeleteGroup Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String) As Long

Sub ESB_DeleteGroup ()
 Dim sts As Long
 Dim GroupName As String  GroupName = "PowerUsers"  '*************
 ' Delete Group
```

```
 '*************
 sts = EsbDeleteGroup (hCtx, GroupName)
End Sub
```

関連トピック

- EsbCreateGroup
- EsbListGroups
- EsbRenameGroup

EsbDeleteLocalContext

以前に **EsbCreateLocalContext()**で作成されたローカル・コンテキストをリリースします。

構文

```
     EsbDeleteLocalContext
      (
     hCtx
     )
ByVal
     hCtx
      As Long
```

パラメータ 説明

hCtx　　VB API ローカル・コンテキスト・ハンドル。

備考

この関数はローカル・コンテキストに対してのみ使用してください。ログイン・コンテキストには、**EsbLogout()**関数を使用します。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbDeleteLocalContext Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_DeleteLocalContext ()
 Dim sts As Long  '*********************
 ' Delete Local Context
 '*********************
 sts = EsbDeleteLocalContext (hCtx)
End Sub
```

関連トピック

- EsbCreateLocalContext
- EsbLogout
- EsbTerm

EsbDeleteLocationAlias

既存のロケーション別名を削除します。

構文

```
      EsbDeleteLocationAlias
       (
      hCtx
      ,
      AliasName
      )
ByVal
      hCtx
         As Long
ByVal
      AliasName
       As String
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル

AliasName　ロケーション別名

戻り値

AliasName という名前が付いたロケーション別名が見つからない場合は、エラーが戻されます。

例

```
    Public Sub LocationAliasTest()

 Dim status  As Long
 Dim ListCount As Integer
 Dim Aliases  As Variant
 Dim HostNames As Variant
 Dim AppNames  As Variant
 Dim DbNames  As Variant
 Dim UserNames As Variant

 status = EsbCreateLocationAlias(hCtx, "blah1", "LocalHost", "Demo", "Basic", _
              "admin", "password")
 If (status <> 0) Then
  MsgBox "Create routine Failed"
  Exit Sub
 End If
```

```
status = EsbCreateLocationAlias(hCtx, "blah2", "LocalHost", "Demo", "Basic", _
              "admin", "password")
If (status <> 0) Then
 MsgBox "Create routine Failed"
 Exit Sub
End If

status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
               AppNames, DbNames, UserNames)
If (status <> 0) Then
 MsgBox "Get routine Failed"
 Exit Sub
End If

If (ListCount > 0) Then
   ' Retrieve the elements as Aliases(0) to Aliases(ListCount -1)
End If

status = EsbDeleteLocationAlias(hCtx, "blah1")
If (status <> 0) Then
 MsgBox "Delete routine Failed"
 Exit Sub
End If

status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
               AppNames, DbNames, UserNames)
If (status <> 0) Then
 MsgBox "Get routine Failed"
 Exit Sub
End If

End Sub
```

関連トピック

- EsbCreateLocationAlias
- EsbGetLocationAliasList

EsbDeleteLogFile

サーバー上のアプリケーション・ログ・ファイルを削除します。

構文

```
     EsbDeleteLogFile
      (
     hCtx, AppName
     )
ByVal
     hCtx
       As Long
ByVal
     AppName
```

```
      As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル
AppName	アプリケーション名。AppName が NULL または""(空の文字列)の場合、**EsbDeleteLogFile()**は、essbase.log ログ・ファイルを削除します。

戻り値

なし。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアプリケーション・デザイン権限(ESB_PRIV_APPDESIGN)を持っている必要があります。

例

```
    Declare Function EsbDeleteLogFile Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String) As Long

Sub ESB_DeleteLogFile ()
 Dim sts As Long
 Dim AppName As String  AppName = "Sample"  '****************
 ' Delete Log file
 '****************
 sts = EsbDeleteLogFile (hCtx, AppName)
End Sub
```

関連トピック

- EsbGetLogFile

EsbDeleteObject

サーバーまたはクライアントのオブジェクト・システムから既存のオブジェクトを削除します。

構文

```
      EsbDeleteObject
       (
      hCtx, ObjType, AppName, DbName, ObjName
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
```

```
      As String
ByVal
      DbName
       As String
ByVal
      ObjName
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。**EsbCreateLocalContext()**によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	削除するオブジェクトの名前。

備考

- オブジェクトを削除するには、そのオブジェクトがロックされていないことが必要です。
- アウトライン・オブジェクトは削除できません。関連付けられているアウトラインを含めて、データベースを削除するには **EsbDeleteDatabase()**関数を使用します。

戻り値

なし。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベースに対して、呼出し元がアプリケーション・デザイン権限またはデータベース・デザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
   Declare Function EsbDeleteObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String)
As Long

Sub ESB_DeleteObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
```

```
 ObjType = ESB_OBJTYPE_RULES  '********************
 ' Delete Rules Object
 '********************
 sts = EsbDeleteObject (hCtx, ObjType, AppName,
 DbName, ObjName)
End Sub
```

関連トピック

- EsbCreateObject
- EsbCopyObject
- EsbListObjects
- EsbRenameObject
- EsbUnlockObject

EsbDeleteUser

既存のユーザーを削除します。

構文

```
      EsbDeleteUser
       (
      hCtx, userName
      )
ByVal
      hCtx
        As Long
ByVal
      userName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
userName	削除するユーザー名。

備考

呼出し元は、サーバー上で、呼出し元のユーザーと前回の管理者のいずれも削除できません。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbDeleteUser Lib "ESBAPIN" (ByVal hCtx As Long, ByVal User As
String) As Long

Sub ESB_DeleteUser ()
 Dim sts As long
 Dim User As String  User = "Joseph"  '************
 ' Delete user
 '************
 sts = EsbDeleteUser (hCtx, User)
End Sub
```

関連トピック

- EsbCreateUser
- EsbListUsers
- EsbRenameUser

EsbDeleteVariable

代替変数を削除します。

構文

```
      EsbDeleteVariable
       (
      hCtx, pVariable
      )
ByVal
      hCtx
         As Long

      pVariable
       As ESB_PVARIABLE_T
```

パラメータ 説明

hCtx　　API へのコンテキスト・ハンドル。

pVariable　削除される代替変数の説明を含む構造体を指すポインタ。

戻り値

成功の場合、ゼロが戻されます。

例

```
    Declare Function EsbDeleteVariable Lib "esbapin" (ByVal hCtx As Long, pVariable
As ESB_VARIABLE_T) As Long

Sub Esb_DeleteVariable ()
```

```
Dim sts As Long
Dim oVariable As ESB_VARIABLE_T

'  Delete "QuarterName" Susbtitution Variable at the Sample application level
 oVariable.Server = "Localhost"
 oVariable.AppName = "Sample"
'  ** Note that DbName has been left empty
 oVariable.VarName = "QuarterName"
 oVariable.VarValue = "Qtr1"

 sts = EsbDeleteVariable(hCtx, oVariable)

End Sub
```

関連トピック

- 1300 ページの「ESB_VARIABLE_T」
- EsbCreateVariable
- EsbGetVariable
- EsbListVariables

EsbDisplayAlias

アクティブ・データベース内の別名テーブルのコンテンツを表示します。

構文

```
      EsbDisplayAlias
       (
      hCtx, AltName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AltName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AltName　　別名テーブル名。

pItems　　別名のアイテムを受け取る変数のアドレス。

備考

Windows のみ: このコマンドで戻される情報は、Windows のメモリー割当て機能を超過しています。Windows のメモリー制限は 64K です。

戻り値

この関数によりテーブルの別名の数が戻され、**EsbGetNextItem()**でアクセスできるMBRALT 構造体の配列が生成されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbDisplayAlias Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Name As
String, Items As Integer) As Long

Sub ESB_DisplayAlias ()
 Dim pItems As Integer
 Dim Name As String
 Dim MbrAlt As ESB_MBRALT_T
 Dim sts As Long  Name = "Default"  '**************
 ' Display Alias
 '**************
 sts = EsbDisplayAlias (hCtx, Name,
 pItems)  For n = 1 To Items   '***************************
  ' Get next Member/Alias Name
  ' combination from the list
  '***************************
  sts = EsbGetNextItem (hCtx,
  ESB_MBRALT_TYPE, MbrAlt)
 Next
End Sub
```

関連トピック

- EsbListAliases
- EsbGetNextItem

EsbEndCalc

アクティブなデータベースに送信される計算スクリプトの終わりをマークします。この関数は、**EsbSendString()**を使用して計算スクリプトを送信した後に呼び出す必要があります。

構文

```
      EsbEndCalc
       (
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx VB API コンテキスト・ハンドル。

備考

- この関数の前に、**EsbBeginCalc()**を呼び出して、**EsbSendString()**を少なくとも 1 回呼び出しておく必要があります。
- **EsbBeginCalc()**、**EsbSendString()**および EsbEndCalc への呼出しが正常終了した場合は、呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
Declare Function EsbEndCalc Lib "ESBAPIN" (ByVal hCtx As Long) As Long
```

EsbBeginCalc の例を参照してください。

関連トピック

- EsbBeginCalc
- EsbCalc
- EsbSendString

EsbEndDataload

アクティブなデータベースに送信される更新指定の終了をマークします。この関数は、**EsbSendString()**を使用して更新定義を送信した後に呼び出す必要があります。

構文

```
Declare Function EsbEndDataload Lib "esbapin" (
ByVal hCtx As Long,
ByVal ErrorName As String) As Long
```

パラメータ 説明

hCtx API コンテキスト・ハンドル。

パラメータ 説明

ErrorName エラー・リストが含まれているテキスト・ファイルの名前。テキスト・ファイル内で予想されるエラー(およびエラー文字列)は次のとおりです:

- ESS_MBRERR_UNKNOWN（未定義メンバー[membername]がデータロード中です。[number]レコードが戻されました。）
- ESS_MBRERR_DBACCESS（アクセス権限が不適切なため、このデータベースではロックを実行できません。）
- ESS_MBRERR_BADDATA（データ列に無効なメンバー[membername]が存在します。）
- ESS_MBRERR_DUPLICATE（データ・レコードの同じ次元からのメンバーが重複しています。[number]レコードが完了しました。）
- AD_MSGDL_ERRORLOAD（アイテム/レコード[number]でのデータロードができません。）

備考

- **EsbEndDataload()**より前に **EsbBeginDataload()**を呼び出し、少なくとも 1 回は **EsbSendString()**を呼び出している必要があります。
- **EsbEndDataload()**からは、エラー・リストを含むテキスト・ファイルの名前が戻されます。

戻り値

成功の場合、0 が戻されます。それ以外の場合は、次のエラー・コードが戻されます:

- abortOnError が TRUE の場合:
  - 最初のエラー条件のエラー・コードが戻されます。
  - エラー・リストは NULL です。
- abortOnError が FALSE の場合:
  - サーバーがデータを処理可能で続行できる場合は、エラー・ファイルが戻されます。
  - それ以外の場合は、例外状況でサーバーが続行できない理由を説明する次のようなエラー・コードが戻されます。例:

    AD_MSGDL_COLS (レコードに含まれるデータ値が多すぎる)

    AD_MSGDL_MISDIM (すべての次元を選択する前にデータ値を検出した)

アクセス

この関数を使用するには、呼出し元が、アクティブなデータベースに対して書込み権限(ESS_PRIV_WRITE)を持っている必要があります。

EsbEndReport

アクティブなデータベースに送信されるレポート指定の終わりをマークします。この関数は、(**EsbSendString()**を使用して)レポート指定を送信した後、かつ(**EsbGetString()**を使用して)戻されたデータを読み取る前に、呼び出す必要があります。

構文

```
      EsbEndReport
       (
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

備考

- この関数の前に **EsbBeginReport()**を呼び出し、**EsbSendString()**を少なくとも 1 回呼び出しておく必要があります。
- レポート・シーケンスを開始する **EsbBeginReport()**の呼出しで、出力フラグが TRUE の場合は、**EsbEndReport()**への呼出しの後に、空の文字列が戻されるまで、**EsbGetString()**への呼出しを繰り返す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbEndReport Lib "ESBAPIN" (ByVal hCtx As Long) As Long
```

EsbBeginReport の例を参照してください。

関連トピック

- EsbBeginReport
- EsbGetString
- EsbSendString

EsbEndUpdate

アクティブなデータベースに送信される更新指定の終了をマークします。この関数は、**EsbSendString()**を使用して更新指定を送信した後に呼び出す必要があります。

構文

```
      EsbEndUpdate
       (
```

```
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

備考

この関数の前に **EsbBeginUpdate()**を呼び出し、**EsbSendString()**を少なくとも 1 回呼び出しておく必要があります。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベースに対して、呼出し元が書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbEndUpdate Lib "ESBAPIN" (ByVal hCtx As Long) As Long
```

EsbBeginUpdate の例を参照してください。

関連トピック

- EsbBeginUpdate
- EsbSendString
- EsbUpdate

EsbExport

データベースを ASCII ファイルにエクスポートします

構文

```
      EsbExport
       (
      hCtx, AppName, DbName, FilePath, Level, isColumns
      )
ByVal
      hCtx
         As Long
ByVal
      AppName
        As String
ByVal
      DbName
```

```
       As String
ByVal
     FilePath
      As String
ByVal
     Level
       As Integer
ByVal
     isColumns
      As Integer
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。

AppName　アーカイブするアプリケーション名。

DbName　アーカイブするデータベース名。

FilePath　アーカイブ情報を含むサーバー・ファイルのフル・パス名。

Level　エクスポートするデータのレベルを制御します。次のいずれかになります:

- ESB_DATA_ALL – すべてのレベルのデータをエクスポート
- ESB_DATA_LEVEL0 – レベル・ゼロのブロックのデータのみをすべてエクスポート
- ESB_DATA_INPUT – 入力レベルのブロックのデータのみをエクスポート

isColumns　列フォーマットのデータ・ブロックの出力を制御します。

備考

この関数が正しく実行されると、この呼出しから戻った後も、サーバー上で非同期プロセスとして続行します。呼出し元は ESB_STATE_DONE が戻されるまで **EsbGetProcessState()**を呼び出して、プロセスが完了したことを定期的に確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対するアクセス権を持っており、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbExport Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, ByVal FilePath As String, ByVal Level As Integer,
ByVal Columns As Integer) As Long

Sub ESB_Export ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim PathName As String
```

```
 Dim Level As Integer
 Dim Columns As Integer
 Dim ProcState As ESB_PROCSTATE_T   AppName = "Sample"
 DbName = "Basic"
 PathName = "c:\essbase\main.txt"
 Level = ESB_DATA_INPUT
 Columns = ESB_YES
 '*****************************
 ' Export input level data only
 '*****************************
 sts = EsbExport (hCtx, AppName, DbName,
 PathName, Level, Columns)
 '***********************************
 ' Check process state till it is done
 '***********************************
 sts = EsbGetProcessState (hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState (hCtx, ProcState)
 Loop
End Sub
```

関連トピック

- EsbImport

EsbGetActive

呼出し元の現在のアクティブなアプリケーションとデータベースの名前を取得します。

構文

```
      EsbGetActive
       (
      hCtx, AppName, szApp, DbName, szDb, pAccess
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      szApp
        As Integer
ByVal
      DbName
       As String
ByVal
      szDb
        As Integer
ByVal
      pAccess
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名の文字列を受け取るバッファ。
szApp	アプリケーション名の文字列バッファのサイズ。
DbName	データベース名の文字列を受け取るバッファ。
szDb	データベース名の文字列バッファのサイズ。
pAccess	選択したデータベースに対するユーザーのアクセス・レベルを受け取る変数のアドレス。

備考

アプリケーション/データベース名の長さがバッファのサイズより大きい場合、名前は切り捨てられます。

戻り値

正常終了の場合、ユーザーの選択済のアクティブなアプリケーションとデータベースが AppName と DbName に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetActive Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal szApp As Integer, ByVal DbName As String, ByVal szDb As Integer, Access
As Integer) As Long

Sub ESB_GetActive ()
 Dim AppName As String * ESB_APPNAMELEN
 Dim DbName As String * ESB_DBNAMELEN
 Dim sts As Long
 Dim szApp As Integer
 Dim szDb As Integer
 Dim pAccess As Integer   szApp = ESB_APPNAMELEN
 szDb = ESB_DBNAMELEN  '*********************************
 ' Get active Application & Database
 '*********************************
 sts = EsbGetActive (hCtx, AppName, szApp, DbName, szDb, Access)
End Sub
```

関連トピック

- EsbClearActive
- EsbSetActive

EsbGetAlias

1 人のユーザーについて、アクティブなデータベースからアクティブな別名テーブル名を取得します。

構文

```
      EsbGetAlias
       (
      hCtx, AltName, szName
      )
ByVal
      hCtx
        As Long
ByVal
      AltName
       As String
ByVal
      szName
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AltName	アクティブな別名テーブルの名前を受け取るバッファ。
szName	アクティブな別名テーブルの名前を受け取るバッファのサイズ。

備考

別名の長さがバッファのサイズより大きい場合、名前は切り捨てられます。

戻り値

正常終了の場合は、アクティブな別名テーブルの名前が AliasName に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbGetAlias Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Name As
String, ByVal szName As Integer) As Long

Sub ESB_GetAlias ()
 Dim sts As Long
 Const szName = 80
 Dim pName As String * szName  '**********
 ' Get Alias
 '**********
 sts = EsbGetAlias (hCtx, pName, szName)
End Sub
```

関連トピック

- EsbListAliases
- EsbSetAlias

- EsbSetActive

EsbGetAPIVersion

Essbase API の現在のバージョンを戻します。

構文

```
EsbGetAPIVersion
 (
lVersion
)

lVersion
 As Long
```

パラメータ 説明

Version　API のバージョン番号。次のフォーマットの、Visual Basic 記法の 16 進値:

```
&H00000000
```

たとえば、H00040000 はリリース 4.0 で、&H00030002 はリリース 3.2 を示します。

- 右から最初の 4 つの数字(下位ワード): バージョン間のリリース番号
- 残りの数字(上位ワード): バージョン番号

備考

プログラムで特定のバージョンが必要な場合に、この関数で API のバージョンを確認できます。

例

```
    Declare Function EsbGetAPIVersion Lib "ESBAPIN" (lVersion As Long) As Long

Sub ESB_GetAPIVersion()
 Dim sts As Long
 Dim Version As Long  '***************
 'Get API Version
 '***************
 sts = EsbGetAPIVersion(Version)
End Sub
```

関連トピック

- EsbGetObjectInfo

EsbGetApplicationAccess

アプリケーションへのユーザーのアクセス権情報が含まれているユーザー・アプリケーション・アクセス構造体のリストを取得します。

構文

```
      EsbGetApplicationAccess
       (
      hCtx, User, AppName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      User
        As String
ByVal
      AppName
       As String

      pItems
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
User	ユーザー名。空の文字列の場合は、指定したアプリケーションのすべてのユーザーがリストされます。
AppName	アプリケーション名。空の文字列の場合は、指定したユーザーのすべてのアプリケーションがリストされます。
pItems	ユーザー・アプリケーション構造体のアイテムを受け取る変数のアドレス。

備考

- User が空の文字列の場合は、指定したアプリケーションのすべてのユーザーがリストされます。AppName が空の文字列の場合は、指定したユーザーのすべてのアプリケーションがリストされます。ただし、User と AppName の両方を空の文字列とすることはできません
- ユーザー・アプリケーションの構造体の Access フィールドは、アプリケーションに対してユーザーに与えられたアクセス権を表すのに使用されます。一方 MaxAccess フィールドは、すべてのソースから得られるユーザーの最高のアクセス権(たとえばグループを介したアクセス権やデフォルトのアプリケーション・アクセス権など)を表します。

戻り値

正常終了の場合、ユーザーおよびアプリケーションのアイテムが pItems に戻され、ユーザー・アプリケーションの構造体のリストが生成されます。このリストには **EsbGetNextItem()**を使用してアクセスできます

アクセス

この関数を使用するには、独自のアプリケーションのアクセス情報を取得する場合を除き、呼出し元は指定されたアプリケーションに対してアプリケーション・デザイン権限(ESB_PRIV_APPDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetApplicationAccess Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
User As String, ByVal AppName As String, Items As Integer) As Long

Sub ESB_GetApplicationAccess ()
 Dim Items As Integer
 Dim AppName As String
 Dim User As String
 Dim UserApp As ESB_USERAPP_T
 Dim sts As Long  AppName = "Demo"
 User = "Joseph"  '***********************
 ' Get Application Access
 '***********************
 sts = EsbGetApplicationAccess (hCtx,
 User, AppName, Items)  For n = 1 To Items    '*********************************
  ' Get next User Application Access
  ' structure from the list
  '*********************************
  sts = EsbGetNextItem (hCtx,
  ESB_USERAPP_TYPE, UserApp)
 Next
End Sub
```

関連トピック

- EsbGetDatabaseAccess
- EsbListUsers
- EsbSetApplicationAccess
- EsbSetUser
- EsbGetNextItem

EsbGetApplicationInfo

ユーザーが構成不可能なアプリケーションのパラメータが含まれている、アプリケーションの情報構造体を取得します。

構文

```
      EsbGetApplicationInfo
       (
      hCtx, AppName, pAppInfo, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
```

```
      As String
ByVal
     pAppInfo
      As ESB_APPINFO_T

     pItems
      As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル(ログイン済)。

AppName　　アプリケーション名。必須。NULL にはできません。

pAppInfo　　アプリケーションの情報構造体を受け取るバッファ。

pItems　　戻されたデータベースのアイテムを受け取る変数のアドレス。

備考

この関数は、サーバー上のアプリケーションに対してのみ呼び出せます。

戻り値

正常終了の場合、アプリケーションの情報構造体が pAppInfo に戻され、データベースの数が pItems に戻されて、**EsbGetNextItem()**を介してアクセス可能なデータベース名文字列のリストが生成されます。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアクセス権を持っている必要があります。

例

```
    Declare Function EsbGetApplicationInfo Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, AppInfo As ESB_APPINFO_T, Items As Integer) As Long

Sub ESB_GetApplicationInfo ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String * ESB_DBNAMELEN
 Dim AppInfo As ESB_APPINFO_T
 Dim sts As Long  AppName = "Sample"  '*******************************
 ' Get Application info structure
 '*******************************
 sts = EsbGetApplicationInfo (hCtx, AppName,
 AppInfo, Items)  For n = 1 To Items    '*******************************
  ' Get next Database name string
  ' from the list
  '*******************************
  sts = EsbGetNextItem (hCtx,
  ESB_DBNAME_TYPE, ByVal DbName)
 Next
End Sub
```

関連トピック

- EsbGetApplicationInfoEx
- EsbGetApplicationState
- EsbGetDatabaseInfo
- EsbGetNextItem

EsbGetApplicationInfoEx

ユーザーが構成不可能なアプリケーションのパラメータも含めて、複数のデータベースから情報を取得します。

構文

```
      EsbGetApplicationInfoEx
       (
      hCtx, AppName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル(ログイン済)。
AppName	アプリケーション名。
pItems	戻されたデータベースのアイテムを受け取る変数のアドレス。

備考

- この関数は、サーバー上のアプリケーションに対してのみ呼び出せます。
- この関数の呼出し元は、EsbGetNextItem を ESB_APPINFOEX_TYPE パラメータを指定して呼び出す必要があります。これにより構造体 ESB_APPINFOEX_T が戻されます。ESB_APPINFOEX_T と ESB_APPINFO_T は、ESB_APPINFOEX_ がデータベース情報を含まないという点を除いて同じです。

戻り値

正常終了の場合は、アプリケーション情報構造体の配列が ppAppInfo に戻されます。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアクセス権を持っている必要があります。

例

```
   Declare Function EsbGetApplicationInfoEx Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, pItems As Integer) As Long

Sub ESB_GetApplicationInfoEx()
 Dim sts As Long
 Dim AppName As String
 Dim Items As Integer
 Dim AppInfoEx As ESB_APPINFOEX_T
 AppName = ""
 '***********************
 'Get application info Ex
 '***********************
 sts = EsbGetApplicationInfoEx(hCtx, AppName,
 Items)
 For n = 1 To Items
  '*********************************
  ' Get next Application Info item
  ' from the list
  '*********************************'
  sts = EsbGetNextItem(hCtx,ESB_APPINFOEX_TYPE,
  AppInfoEx)
 Next
End Sub
```

関連トピック

- EsbGetApplicationInfo
- EsbGetApplicationState
- EsbGetDatabaseInfo
- EsbGetNextItem

EsbGetApplicationState

ユーザーが構成可能なアプリケーションのパラメータが含まれている、アプリケーションの状態構造体を取得します。

構文

```
      EsbGetApplicationState
       (
      hCtx, AppName, pAppState
      )
ByVal
      hCtx
         As Long
ByVal
      AppName
       As String

      pAppState
       As ESB_PAPPSTATE_T
```

パラメータ 説明

hCtx VB API コンテキスト・ハンドル。

AppName アプリケーション名。

pAppState アプリケーションの状態構造体を受け取るバッファ。

備考

この関数はローカル・アプリケーションに対しては呼び出せません。この関数はサーバーのアプリケーションに対してのみ呼び出せます。

戻り値

正常終了の場合は、アプリケーションの状態構造体が pAppState に戻されます。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアクセス権を持っている必要があります。

例

```
    Declare Function EsbGetApplicationState Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, AppState As ESB_APPSTATE_T As Long

Sub ESB_GetApplicationState ()
 Dim sts As Long
 Dim AppName As String
 Dim AppState As ESB_APPSTATE_T  AppName = "Sample"
 '*******************************
 ' Get Application State structure
 '*******************************
 sts = EsbGetApplicationState (hCtx, AppName,
 AppState)
End Sub
```

関連トピック

- EsbGetApplicationInfo
- EsbGetDatabaseState
- EsbSetApplicationState

EsbGetAssociatedAttributesInfo

指定した基本メンバーに関連付けられている属性情報を戻します。

構文

```
EsbGetAssociatedAttributesInfo
 (
hCtx
,
MbrName
,
```

```
      AttrDimName
      ,
      Count
      )
ByVal
      hCtx
          As Long
ByVal
      MbrName
        As String
ByVal
      AttrDimName
       As String

      Count
         As Long
```

パラメータ　説明

hCtx	コンテキスト・ハンドル
MbrName	基本メンバー名
AttrDimName	(オプション)属性次元名
Count	戻された属性メンバー数

備考

- この関数を利用すると、属性メンバーに関する情報が EsbQueryDatabaseMembers の場合よりも多く取得できます。
- AttrDimName を NULL に設定すると、基本メンバーに関連付けられているすべての属性メンバーが戻されます。
- オプションで、属性次元名を指定すると、基本メンバーに関連付けられているその次元メンバーに関する情報のみを取得できます。
- **EsbGetAssociatedAttributesInfo()**を呼び出した後に、1269 ページの「ESB_ATTRIBUTEINFO_T」を使用して **EsbGetNextItem()**を呼び出すと、必要な属性情報の構造体を取得できます。
- この関数によって戻される属性情報が無効になる状況には、次の 2 つがあります:

  1. Visual Basic API では、指定した属性次元の名前から、属性データ型が派生します。そのため、属性次元の値が有効でない場合もあります。**EsbGetAssociatedAttributesInfo()**に渡された名前が属性次元の名前と同じ場合、アプリケーションは、戻された ESB_ATTRIBUTEINFO_T 構造体の属性フィールドの値を無視する必要があります。MbrInfo.MbrName フィールドおよび MbrInfo.DimName フィールドが等しいかどうかを確認します。等しい場合は、基本次元を参照しているので、属性情報を無視する必要があります。
  2. 日付属性には、Visual Basic によって自動的に処理される時刻情報(タイム・スタンプ)が含まれています(日付計算が実施される)。これによって日付属性の値に間違いが生じることがあります(指定したクライアント・マシン

で指定されたタイムゾーンによります)。自動処理を避けるため、日付属性情報の表示では、属性値ではなく、属性名を使用します。

戻り値

正常終了の場合は sts = 0 が戻されます。それ以外の場合は、エラー番号が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_GetAssociatedAttributesInfo()
    ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text
box.
    Dim hCtx as long
    Dim sts as long
    Dim MbrName As String
    Dim AttrDimName As String
    Dim Count As Long
    Dim Attribinfo As ESB_ATTRIBUTEINFO_T
    Dim index As Integer
    Dim tempstring As String

    MbrName = InputBox("Base member name", "Base Member Name")
    AttrDimName = InputBox("Attribute Dimension Name (Optional)", "Attribute
Dimension Name")

    sts = EsbGetAssociatedAttributesInfo(hCtx, MbrName, AttrDimName, Count)

    If sts <> 0 Then
        MsgBox "Error in ESB_GetAssociatedAttributesInfo: " & sts: Exit Sub
    Else
        tempstring = "...count = " & Count & "..."
        out (tempstring)

        Out "Associated Attr info for " & "[" & MbrName & "]"
        Out "-------------------------------------"

        For index = 1 To Count
            sts = EsbGetNextItem(hCtx, ESB_ATTRIBUTEINFO_TYPE, Attribinfo)
            Out "Dim Name: " & Attribinfo.DimName
            Out "Mbr Name: " & Attribinfo.MbrName

        ' NOTE: use of select case statement to discern (and act upon) type of
attribute returned
        Select Case VarType(Attribinfo.Attribute)
            Case vbDouble
                Out "Data Type   : Numeric(Double)"
                Out "Data Value  : " & Attribinfo.Attribute
                Out ""
            Case vbBoolean
                Out "Data Type   : Boolean"
                Out "Data Value  : " & Attribinfo.Attribute
                Out ""
```

```
            Case vbDate
                Out "Data Type   : Date"
                 ' Suggested way to get Date Attribute value for display
                Out "Data Value  : " & Attribinfo.DimName
                Out ""
            Case vbString
                Out "Data Type   : String"
                Out "Data Value  : " & Attribinfo.Attribute
                Out ""
                End Select
                Out ""
            Next index
    End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbGetAttributeInfo

指定した属性メンバーまたは属性次元に関する属性情報を戻します。

構文

```
      EsbGetAttributeInfo
       (
      hCtx
      ,
      AttrName
      ,
      AttrInfo
      )
ByVal
      hCtx
        As Long
ByVal
      AttrName
       As String
```

```
      AttrInfo
       As ESB_ATTRIBUTEINFO_T
```

パラメータ 説明

hCtx　　コンテキスト・ハンドル

AttrName　属性メンバーまたは次元の名前

AttrInfo　属性情報

備考

基本メンバーまたは次元が渡されると情報が戻されますが、属性固有の情報は表示されません。また、基本次元が渡された場合では、構造体の次元とメンバー名のフィールドは同一の値を保持します。

戻り値

正常終了の場合は sts = 0 が戻され、ESB_ATTRIBUTEINFO_T 構造体に入力されます。それ以外の場合は、エラー番号が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_GetAttributeInfo()
 ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text box.
 Dim hCtx as long
 Dim sts as long
 Dim MbrName As String
 Dim OutAttrInfo As ESB_ATTRIBUTEINFO_T

 MbrName = InputBox("Member Name")
 sts = EsbGetAttributeInfo(hCtx, MbrName, OutAttrInfo)
 If sts = 0 Then
  Out "ESB_OtlGetAttributeInfo passed" & sts
  Out "MbrName : " & OutAttrInfo.MbrName
  Out "DimName : " & OutAttrInfo.DimName
  Out "Attribute : " & OutAttrInfo.Attribute
 Else
  Out "ESB_OtlGetAttributeInfo failed" & sts: Exit Sub
 End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember

- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbGetAttributeSpecifications

アウトラインの属性指定を取得します。

構文

```
      EsbGetAttributeSpecifications
       (
      hCtx
      ,
      AttrSpecs
      )
ByVal
      hCtx
         As Long

      AttrSpecs
       As As ESB_ATTRSPECS_T
```

パラメータ	説明
hCtx	コンテキスト・ハンドル
AttrSpecs	属性指定

備考

- アウトラインの属性指定を設定するには、**EsbOtlSetAttributeSpecifications()**を使用します。
- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_GetAttributeSpecifications()
 ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text box.
 Dim OutAttrSpecs As ESB_ATTRSPECS_T
```

```
Dim sts as long
Dim hCtx as long
Dim test As String

sts = EsbGetAttributeSpecifications(hCtx, OutAttrSpecs)

If sts <> 0 Then Out "ESB_GetAttributeSpecifications failed" & sts: Exit Sub

Out "ESB_GetAttributeSpecifications passed: " & sts
Out "DefaultTrueString : " & OutAttrSpecs.DefaultTrueString
Out "DefaultFalseString : " & OutAttrSpecs.DefaultFalseString
Out "DefaultAttrCalcDimName : " & OutAttrSpecs.DefaultAttrCalcDimName
Out "DefaultSumMbrName : " & OutAttrSpecs.DefaultSumMbrName
Out "DefaultCountMbrName : " & OutAttrSpecs.DefaultCountMbrName
Out "DefaultAverageMbrName : " & OutAttrSpecs.DefaultAverageMbrName
Out "DefaultMinMbrName : " & OutAttrSpecs.DefaultMinMbrName
Out "DefaultMaxMbrName : " & OutAttrSpecs.DefaultMaxMbrName

test = OutAttrSpecs.GenNameBy
Select Case test
 Case ESB_GENNAMEBY_PREFIX
  Out "GenNameBy : ESB_GENNAMEBY_PREFIX"
 Case ESB_GENNAMEBY_SUFFIX
  Out "GenNameBy : ESB_GENNAMEBY_SUFFIX"
 Case Else
  Out "GenNameBy : invalid"
End Select

test = OutAttrSpecs.UseNameOf
Select Case test
 Case ESB_USENAMEOF_NONE
  Out "UseNameOf : ESB_USENAMEOF_NONE"
 Case ESB_USENAMEOF_PARENT
  Out "UseNameOf : ESB_USENAMEOF_PARENT"
 Case ESB_USENAMEOF_GRANDPARENTANDPARENT
  Out "UseNameOf : ESB_USENAMEOF_GRANDPARENTANDPARENT"
 Case ESB_USENAMEOF_ALLANCESTORS
  Out "UseNameOf : ESB_USENAMEOF_ALLANCESTORS"
 Case ESB_USENAMEOF_DIMENSION
  Out "UseNameOf : ESB_USENAMEOF_DIMENSION"
 Case Else
  Out "UseNameOf : invalid"
End Select

test = OutAttrSpecs.Delimiter
Select Case test
 Case ESB_DELIMITER_UNDERSCORE
  Out "Delimiter : ESB_DELIMITER_UNDERSCORE"
 Case ESB_DELIMITER_PIPE
  Out "Delimiter : ESB_DELIMITER_PIPE"
 Case ESB_DELIMITER_CARET
  Out "Delimiter : ESB_DELIMITER_CARET"
 Case Else
End Select

test = OutAttrSpecs.DateFormat
Select Case test
```

```
  Case ESB_DATEFORMAT_MMDDYYYY
    Out "DateFormat : ESB_DATEFORMAT_MMDDYYYY"
  Case ESB_DATEFORMAT_DDMMYYYY
    Out "DateFormat : ESB_DATEFORMAT_DDMMYYYY"
  Case Else
    Out "Delimiter : invalid"
 End Select

 test = OutAttrSpecs.BucketingType
 Select Case test
  Case ESB_UPPERBOUNDINCLUSIVE
    Out "BucketingType : ESB_UPPERBOUNDINCLUSIVE"
  Case ESB_LOWERBOUNDINCLUSIVE
    Out "BucketingType : ESB_ESB_LOWERBOUNDINCLUSIVE"
  Case ESB_UPPERBOUNDNONINCLUSIVE
    Out "BucketingType : ESB_UPPERBOUNDNONINCLUSIVE"
  Case ESB_LOWERBOUNDNONINCLUSIVE
    Out "BucketingType : ESB_LOWERBOUNDNONINCLUSIVE"
  Case Else
    Out "BucketingType : invalid"
 End Select
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbGetCalcList

ユーザーがアクセス可能な計算スクリプト・オブジェクトを入手します。プログラマは **EsbGetNextItem()**を使用して、使用可能なスクリプトのリストにアクセスする必要があります。

構文

```
      EsbGetCalcList
       (
      hCtx, UserName, AppName, DbName, isAllCalcs, pItems
      )
ByVal
```

```
      hCtx
         As Long
ByVal
      UserName
       As String
ByVal
      AppName
        As String
ByVal
      DbName
        As String

      isAllCalcs
       As Integer

      pItems
        As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
UserName	ユーザー名。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます
isAllCalcs	AllowAllCalcs フラグを含む整数。AllowAllCalcs が ESB_FALSE に設定されている場合は、ユーザーはすべての計算スクリプト・オブジェクトにアクセスできます。それ以外の場合は、CalcList 引数で指定されたスクリプト・オブジェクトにのみアクセスできます。
pItems	その内容が使用可能な計算スクリプト・オブジェクトの数を含む整数。

備考

- 計算スクリプト・オブジェクトにアクセスするには、指定されたユーザーが、少なくとも適切なデータベースに対する計算アクセス権を持っている必要があります。
- pAllCalcs に戻された値が TRUE の場合、pItems に戻される値はゼロです。

戻り値

正常終了の場合は sts=0 とユーザーの AllowAllCalcs 設定を pAllCalcs に戻します。AllCalcs が ESB_FALSE と等しい場合、pItems には使用可能な計算スクリプト・オブジェクト数が含まれます。**EsbGetNextItem()**を使用して計算スクリプト・オブジェクト名のリストにアクセスします。

isAllCalcs が ESB_TRUE に等しい場合、pItems から 0 が戻されるので、プログラマは戻された各オブジェクトに **EsbListObjects()**(ESB_OBJTYPE_CALCSCRIPT タイプ使用)と **EsbGetObjectInfo()**の組合せを呼び出す必要があります。

アクセス

この関数を使用するには、独自の計算リストを取得する場合を除いて、呼出し元は指定したデータベースに対してデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetCalcList Lib "ESBAPIN" (ByVal hCtx As Long, ByVal User As
String, ByVal AppName As String, ByVal DbName As String, AllCalcs As Integer, Items
As Integer) As Long

Sub ESB_GetCalcList()
  Dim Items As Integer
  Dim AppName As String
  Dim DbName As String
  Dim User As String
  Dim AllCalcs As Integer
  Dim ObjName As String * ESB_OBJNAMELEN
  Dim sts As Long
  Dim ObjType As Long
  Dim ObjectInfo As ESB_OBJINFO_T  ObjType = ESB_OBJTYPE_CALCSCRIPT  AppName =
"Sample"
  DbName = "Basic"
  User = "test_user" ' Has 'calculate' access to Sample->Basic  ' If user passed in
has access to everything,
    ' then Items will ALWAYS be set to '0'!
  ' In that case, use EsbListObjects()
    ' (of type ESB_OBJTYPE_CALCSCRIPT, and
    ' then EsbGetObjectInfo()!
  sts = EsbGetCalcList(hCtx, User, AppName, DbName, AllCalcs, Items)
  If AllCalcs = ESB_NO Then
    frmMain.lstInfo.AddItem "Number of calc script items returned: " & Items
    frmMain.lstInfo.AddItem "--------------------------------------------------"
    For n = 1 To Items
      sts = EsbGetNextItem(hCtx, ESB_OBJNAME_TYPE, ByVal ObjName)
      If sts <> 0 Then MsgBox "Failure in EsbGetNextItem(): " & sts: Exit Sub
      sts = EsbGetObjectInfo(hCtx, ObjType, AppName, DbName, ObjName, ObjectInfo)
      If sts <> 0 Then MsgBox "Failure in EsbGetObjectInfo(): " & sts: Exit Sub
      frmMain.lstInfo.AddItem ObjectInfo.Name
      frmMain.lstInfo.AddItem ObjectInfo.Type
      frmMain.lstInfo.AddItem "----------"
    Next
  Else
    frmMain.lstInfo.AddItem "You need to call EsbListObjects of type
ESB_OBJTYPE_CALCSTRIPT"
  End If
End Sub
```

関連トピック

- EsbListObjects
- EsbListUsers
- EsbSetCalcList
- EsbGetNextItem

EsbGetCellDrillThruReports

データ・セルに関連付けられたドリルスルー・レポートを、セルのメンバーの組合せを使用し、URL XML のリストとして取得します。

構文

```
Declare Function EsbGetCellDrillThruReports Lib "esbapin" (ByVal hCtx As Long, ByRef pMbrs() As String, ByRef ppURLXMLLen As Variant, ByRef ppURLXML As Variant) As Long
```

パラメータ	説明
hCtx	Visual Basic API のコンテキスト・ハンドル
pMbrs	メンバー名(または別名)のリスト
ppURLXMLLen	生成された URL XML の長さが戻されます
ppURLXML	URL XML バイト・ストリームへのポインタが戻されます

備考

この呼出しを行うためには、アプリケーション・データベースをアクティブに設定する必要があります。クライアントで必要とされる追加情報をサポートするには、この関数を拡張する必要があります。

戻り値

- 正常に処理されると、URL XML のリストが取得されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESB_PRIV_READ)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
   Sub ESB_GetCellDrillThruReports()
Dim intX      As Integer
Dim mbrs(0 To 4)  As String
Dim pURLXMLLens   As Variant
Dim pURLXMLs    As Variant

mbrs(0) = "sales"
mbrs(1) = "jan"
mbrs(2) = "New York"
mbrs(3) = "actual"
mbrs(4) = "100-10"

sts = EsbGetCellDrillThruReports(hCtx, mbrs, pURLXMLLens, pURLXMLs)
```

```
  If sts = 0 Then

    Debug.Print "EsbGetCellDrillThruReports sts: " & sts
    For intX = LBound(pURLXMLLens) To UBound(pURLXMLLens)

    Debug.Print "URL XML: " & intX
    Debug.Print "URL XML Len: " & pURLXMLLens(intX)
    Debug.Print "URL XML String: " & pURLXMLs(intX)

    Next
  End If

  mbrs(0) = "profit"
  sts = EsbGetCellDrillThruReports(hCtx, mbrs, pURLXMLLens, pURLXMLs)
  If sts = 0 Then
    Debug.Print "EsbGetCellDrillThruReports sts: " & sts
    For intX = LBound(pURLXMLLens) To UBound(pURLXMLLens)
    Debug.Print "URL XML: " & intX
    Debug.Print "URL XML Len: " & pURLXMLLens(intX)
    Debug.Print "URL XML String: " & pURLXMLs(intX)
    Next
  End If
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbGetCurrencyRateInfo

アクティブ・データベース・アウトライン内の、タグ付き通貨パーティション次元のすべてのメンバーについてのレート情報が含まれている構造体のリストを取得します。

構文

```
      EsbGetCurrencyRateInfo
       (
      hCtx, pItems
      )
ByVal
      hCtx
        As Long

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pItems　　レート情報構造体のアイテムを受け取る変数のアドレス。

備考

この関数は、関連通貨データベースが指定されている標準のデータベースに対して呼び出せます。

戻り値

正常終了の場合、構造体のアイテムが pItems に戻され、通貨情報構造体の配列が生成されます。この配列には **EsbGetNextItem()**を使用してアクセスできます。

アクセス

この関数を使用するには、呼出し元がデータベースに対するアクセス権を持っており、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbGetCurrencyRateInfo Lib "ESBAPIN" (ByVal hCtx As Long, Items
As Integer) As Long

Sub ESB_GetCurrencyRateInfo ()
 Dim Items As Integer
 Dim RateInfo As ESB_RATEINFO_T
 Dim sts As Long  '************************
 ' Get Currency Rates Info
 '************************
 sts = EsbGetCurrencyRateInfo (hCtx, Items)  For n = 1 To Items
  '*************************
  ' Get next Rates Info item
  'from the list
  '*************************
  sts = EsbGetNextItem (hCtx,
  ESB_RATEINFO_TYPE, RateInfo)
 Next
End Sub
```

関連トピック

- EsbListCurrencyDatabases
- EsbGetNextItem
- EsbSetActive

EsbGetDatabaseAccess

データベースへのユーザーのアクセス権情報が含まれている、ユーザーのデータベース・アクセス構造体のリストを取得します。

構文

```
     EsbGetDatabaseAccess
      (
     hCtx, User, AppName, DbName, pItems
     )
ByVal
```

```
      hCtx
        As Long
ByVal
      User
        As String
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
User	ユーザー名。空の文字列の場合は、指定したアプリケーションとデータベースのすべてのユーザーがリストされます。
AppName	アプリケーション名。空の文字列の場合は、指定したユーザーのすべてアプリケーションとデータベースがリストされます。
DbName	データベース名。空の文字列の場合は、指定したユーザーまたはアプリケーションのすべてのデータベースがリストされます。
pItems	ユーザー・データベース構造体のアイテムを受け取る変数のアドレス。

備考

- User、AppName、または DbName のいずれかが空の文字列の場合、ワイルドカードとして扱われ、該当するタイプのすべてのアイテムがリストされます。AppName が空の文字列の場合、DbName も空の文字列としてみなされます。これらの引数のうち、2 つまでは空の文字列でもかまいませんが、3 つすべてを空の文字列とすることはできません。
- ユーザー・データベース構造体の Access フィールドは、データベースに対してユーザーに与えられたアクセス権を表します。一方 MaxAccess フィールドは、すべてのソースから得られるユーザーの最も高いアクセス権(たとえばグループを介したアクセス権やデフォルトのデータベース・アクセス権など)を表します。
- フィルタのアクセス権限は、**ESB_PRIV_DBLOAD** 権限と同じです。

戻り値

正常終了の場合は、ユーザーおよびデータベースのアイテムが pItems に戻され、ユーザー・データベース構造体のリストが生成されます。このリストには **EsbGetNextItem()**を使用してアクセスできます。

アクセス

この関数を使用するには、独自のデータベース・アクセス情報を取得する場合を除いて、呼出し元は指定したデータベースに対してデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetDatabaseAccess Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
User As String, ByVal AppName As String, ByVal DbName As String, Items As Integer) As
Long

Sub ESB_GetDatabaseAccess ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim User As String
 Dim UserDb As ESB_USERDB_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"
 User = "Joseph"  '********************
 ' Get Database Access
 '********************
 sts = EsbGetDatabaseAccess (hCtx,
 User, AppName, DbName, Items)  For n = 1 To Items   '******************************
  ' Get next User Database Access
  ' structure from the list
  '*****************************
  sts = EsbGetNextItem (hCtx,
  ESB_USERDB_TYPE, UserDb)
 Next
End Sub
```

関連トピック

- EsbGetApplicationAccess
- EsbGetUser
- EsbListUsers
- EsbSetDatabaseAccess
- EsbGetNextItem

EsbGetDatabaseInfo

ユーザーが構成不可能なデータベースのパラメータが含まれている、データベースの情報構造体を取得します。

構文

```
      EsbGetDatabaseInfo
       (
      hCtx, AppName, DbName, DbInfo, pItems
      )
ByVal
      hCtx
```

```
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      DbInfo
       As ESB_DBINFO_T

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル
AppName	アプリケーション名
DbName	データベース名
DbInfo	データベース情報構造体を受け取るバッファ。
pItems	戻された 1273 ページの「ESB_DBREQINFO_T」構造体の数。

備考

- この関数は、サーバー・データベースの情報構造体のみを取得します。
- このルーチンの呼出し元は、ESB_DBREQINFO_TYPE を指定して、EsbGetNextItem を呼び出す必要があります。これにより ESB_DBREQINFO_T 型の構造体が戻されます。この構造体は、前回の計算、データのロードおよびアウトラインの更新などの要求情報を含んでいます。

戻り値

正常終了の場合は、データベース情報構造体へのポインタが pDbInfo に戻されます。

アクセス

この関数を使用するには、呼出し元が少なくとも指定したデータベースに対して読取りアクセス権(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbGetDatabaseInfo Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, pDbInfo As ESB_DBINFO_T, Items As Integer)
As Long

Sub ESB_GetDatabaseInfo()
    Dim sts As Long
    Dim AppName As String
    Dim DbName As String
    Dim Items As Integer
    Dim n As Integer
```

```
Dim DbInfo As ESB_DBINFO_T
Dim DbReqInfo As ESB_DBREQINFO_T,
Dim Items As Integer
AppName = "Sample"
DbName = "Basic"
sts = EsbGetDatabaseInfo(hCtx, AppName, DbName, DbInfo, Items)
If sts = 0 Then
   For n = 1 To Items
      sts = EsbGetNextItem(hCtx, ESB_DBREQINFO_TYPE, DbReqInfo)
          Next  End
IfEnd Sub
```

関連トピック

- EsbGetApplicationInfo
- EsbGetDatabaseState
- EsbGetDatabaseStats

EsbGetDatabaseInfoEx

ユーザーが構成できないデータベース用パラメータを含む、1 つ以上のデータベースに関する情報を取得します。

構文

```
     Declare Function EsbGetDatabaseInfoEx Lib "esbapin" (
ByVal hCtx As Long,
ByVal AppName As String,
ByVal DbName As String,
   pItems As Integer) As Long
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
AppName	データベース情報が戻されるアプリケーション名。NULL の場合、すべてのアプリケーションとデータベースについての情報が戻されます。
DbName	データベース情報が戻されるデータベース名。NULL の場合、すべてのデータベースについての情報が戻されます。
pItems	戻される情報構造体の数。

備考

- プログラムから、**ESB_DBREQINFO_T** パラメータを指定して **EsbGetNextItem()** を呼び出す必要があります。
- この関数は、サーバー・データベースの情報構造体のみ取得できます。

戻り値

正常終了の場合、アクセス可能なデータベースの数が pCount に戻され、アプリケーションとデータベースの名前のリストを生成します。リストは、**EsbGetNextItem()**を介してアクセス可能です。

アクセス

この関数を使用するには、呼出し元が少なくとも指定したデータベースに対する読取りアクセス権(ESS_PRIV_READ)を持っている必要があります。

EsbGetDatabaseNote

データベースの最新情報に関するメッセージを取得します。このメッセージを使用して、ユーザーがデータベースに接続する前に、データベースに関する有用な情報(データがロードされているかどうか、データが最後に計算されたのはいつかなど)を表示できます。

構文

```
      EsbGetDatabaseNote
       (
      hCtx, AppName, DbName, DbNote, szDbNote
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      DbNote
        As String
ByVal
      szDbNote
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
DbNote	データベース・ノート文字列を受け取るバッファ。
szDbNote	バッファのサイズ。

備考

- データベース・ノート文字列の長さは常に、64KB 未満である必要があります。

- アプリケーションまたはデータベースの名前の長さがバッファのサイズより大きい場合、名前は切り捨てられます。
- データベースのメッセージは、**EsbSetDatabaseNote()**を使用して設定します。

戻り値

正常終了の場合は、データベースのメッセージ文字列が DbNote に戻されます。

アクセス

この関数を使用するには、呼出し元は指定したデータベースに対してアクセス権を持っている必要があります。

例

```
    Declare Function EsbGetDatabaseNote Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, ByVal DbNote As String, ByVal szDbNote As
Integer) As Long

Sub ESB_GetDatabaseNote ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Const szDbNote = 256
 Dim DbNote As String * szDbNote
 AppName = "Sample"
 DbName = "Basic"  '******************
 ' Get Database note
 '******************
 sts = EsbGetDatabaseNote (hCtx, AppName,
    DbName, DbNote, szDbNote)
End Sub
```

関連トピック

- EsbSetDatabaseNote

EsbGetDatabaseState

ユーザーが構成可能なデータベースのパラメータが含まれている、データベースの状態構造体を取得します。

構文

```
      EsbGetDatabaseState
       (
      hCtx, AppName, DbName, pDbState
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
```

```
DbName
 As String

pDbState
 As ESB_DBSTATE_T
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。

AppName　アプリケーション名。

DbName　データベース名。

pDbState　データベースの状態構造体を受け取るバッファ。

備考

この関数は、サーバー・データベースの状態構造体のみを取得します。

戻り値

正常終了の場合、データベースの状態構造体へのポインタが pDbState に戻されます。

アクセス

データベースの状態構造体を取得するには、接続しているユーザーはデータベースに対して少なくとも読取りアクセス権(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbGetDatabaseState Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, pDbState As ESB_DBSTATE_T) As Long

Sub ESB_GetDatabaseState()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim DbState As ESB_DBSTATE_T  AppName = "Sample"
 DbName = "Basic"  '*********************************
 '****** Get Database State ******
 '*********************************
 sts = EsbGetDatabaseState(hCtx, AppName, DbName,
 DbState)
End Sub
```

関連トピック

- EsbGetApplicationState
- EsbGetDatabaseInfo
- EsbSetDatabaseState
- EsbGetDatabaseStats

EsbGetDatabaseStats

データベースに関する統計情報が含まれている、アクティブ・データベースの統計構造体を取得します。

構文

```
      EsbGetDatabaseStats
       (
      hCtx, AppName, DbName, pDbStats, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String

      pDbStats
       As ESB_PDBSTATS_T

      pItems
        As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
pDbStats	データベースの統計構造体を受け取るバッファ。
pItems	次元の統計アイテムのアイテムを受け取る変数のアドレス。

備考

- この関数は、サーバー・データベースに対してのみ呼び出せます。
- データベースがまだロードされていない場合、この関数がデータベースをロードします。

戻り値

正常終了の場合は、割り当てられたデータベースの統計構造体へのポインタが pDbStats に、次元数が pItems に戻され、次元統計構造体のリストが生成されます。このリストには、**GetNextItem()**を使用してアクセスできます。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbGetDatabaseStats Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, DbStats As ESB_DBSTATS_T, Items As
Integer) As Long

Sub ESB_GetDatabaseStats ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim DbStats As ESB_DBSTATS_T
 Dim DimStats As ESB_DIMSTATS_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"  '********************
 ' Get Database stats
 '********************
 sts = EsbGetDatabaseStats (hCtx, AppName, DbName, DbStats, Items)  For n = 1 To
Items   '******************************
  ' Get next Dimension stats item
  'from the list
  '******************************
  sts = EsbGetNextItem (hCtx,
  ESB_DIMSTATS_TYPE, DimStats)
 Next
End Sub
```

関連トピック

- EsbGetDatabaseInfo
- EsbGetDatabaseState
- EsbGetNextItem
- EsbSetActive

EsbGetDefaultCalc

アクティブ・データベースのデフォルト計算スクリプトを取得します。

構文

```
     EsbGetDefaultCalc
      (
     hCtx, cscString, szString
     )
ByVal
     hCtx
        As Long
ByVal
     cscString
      As String
ByVal
     szString
      As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

cscString　計算スクリプト文字列を受け取るバッファ。

szString　計算スクリプト文字列を受け取るバッファのサイズ。

備考

- 戻される計算スクリプト文字列の長さは、64KB 未満である必要があります。
- 計算スクリプトの長さがバッファのサイズより大きい場合、スクリプトは切り捨てられます。

戻り値

正常終了の場合、データベースのデフォルト計算スクリプトが CalcScript に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESB_PRIV_READ)を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbGetDefaultCalc Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
Script As String, ByVal szScript As Integer) As Long

Sub ESB_GetDefaultCalc ()
 Dim sts As Long
 Const szScript = 1024
 Dim Script As String * szScript  '*****************
 ' Get default calc
 '*****************
 sts = EsbGetDefaultCalc (hCtx, Script, szScript)
End Sub
```

関連トピック

- EsbDefaultCalc
- EsbSetDefaultCalc
- EsbSetDefaultCalcFile
- EsbSetActive

EsbGetDimensionInfo

次元に関する情報を取得します。

構文

EsbGetDimensionInfo

```
      (
      hCtx, Dimension, pItems
      )
ByVal
      hCtx
         As Long
ByVal
      Dimension
       As String

      pItems
        As Integer
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

Dimension　情報が戻される次元メンバー名。NULL の場合、各次元についての情報が戻されます。

pItems　　戻される情報構造体の数。

備考

- 呼出し元は、ESB_DIMINFO_TYPE パラメータを指定して EsbGetNextItem を呼び出す必要があります。
- 属性:
  - 1279 ページの「ESB_DIMENSIONINFO_T」構造体の DimTag フィールドの定数値 ESB_TTYPE_ATTRIBUTE と ESB_TTYPE_ATTRCALC は、次元が属性次元であることを示します。
  - 1279 ページの「ESB_DIMENSIONINFO_T」構造体の DimDataType フィールドは、属性次元のタイプを示します。

戻り値

正常終了の場合は、次元の情報構造体の数に対する参照が戻されます。

アクセス

この関数を使用するには、指定されたデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetDimensionInfo Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
Dimension As String, pItems As Integer) As Long

Sub ESB_GetDimensionInfo()
 Dim sts As Long
 Dim Dimension As String
 Dim nDims As Integer
 Dim DimInfo As ESB_DIMENSIONINFO_T  Dimension = "Year"
 sts = EsbGetDimensionInfo(hCtx, Dimension, nDims)  If Not sts Then
  For n = 1 To nDims
  '*************************
  ' Get next Dimension Info
```

```
   ' from the list
   '*************************
   sts = EsbGetNextItem(hCtx,
    ESB_DIMINFO_TYPE, DimInfo)
  Next
 End If
End Sub
```

関連トピック

- EsbBuildDimension
- EsbGetApplicationInfo
- EsbGetApplicationInfoEx
- EsbGetDatabaseInfo

EsbGetDrillThruURL

アクティブなデータベース・アウトライン内のドリルスルー URL 名のリストを取得します。

1904 ページの「ドリルスルー URL の制限」.

構文

```
     Declare Function EsbGetDrillThruURL Lib "esbapin" (ByVal hCtx As Long, ByVal
URLName As String, pUrl As ESB_DURLINFO_T, ByRef symRegions As Variant) As Long
```

パラメータ 説明

hCtx	Visual Basic API のコンテキスト・ハンドル
URLName	ドリルスルー URL 名
pUrl	URL 定義
symRegions	対称領域のリスト

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内のドリルスルー URL のリストが取得されます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESB_PRIV_READ)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
    Sub ESB_GetGLDrillThru()
```

```
 Dim URLName      As String
 Dim url        As ESB_DURLINFO_T
 Dim intX       As Integer
 Dim cppDrillRegions  As Variant

 URLName = "VB URL2"
 sts = EsbGetDrillThruURL(hCtx, URLName, url, cppDrillRegions)

 Debug.Print "EsbGetDrillThruURL sts: " & sts

 If sts = 0 Then
  Debug.Print "URL Name: " & url.cpURLName
  Debug.Print "URL XML: " & url.cpURLXML

  For intX = LBound(cppDrillRegions) To UBound(cppDrillRegions)

  Debug.Print "URL Region: " & cppDrillRegions(intX)

 Next
 End If
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbGetFilter

フィルタのコンテンツの取得を開始します。

構文

```
      EsbGetFilter
       (
      hCtx, AppName, DbName, FltName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      FltName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル

AppName　　アプリケーション名

DbName　　データベース名

FltName　　フィルタ名

pItems　　ユーザー・アプリケーション構造体のアイテムを受け取る変数のアドレス。

備考

この呼出しの後に続いて **EsbGetFilterRow()**を呼び出し、フィルタ行をフェッチする必要があります。

戻り値

正常終了の場合、フィルタのアクティブ・フラグが pActive に、そしてデフォルトのフィルタ・アクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, ByVal FltName As String, Active As Integer, pAccess
As Integer) As Long

Sub ESB_GetFilter ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Dim Active As Integer
 Dim pAccess As Integer
 Const szRow = 512
 Dim Row As String * szRow  AppName = "Sample"
 DbName = "Basic"
 FilterName = "Filter"  '***********
 ' Get Filter
 '***********
 sts = EsbGetFilter (hCtx, AppName, DbName, FilterName, Active,
  pAccess)  '****************
 ' Get Filter Rows
 '****************
 sts = EsbGetFilterRow (hCtx, Row, szRow, pAccess)
  Do While Mid$(Row,1,1) <> chr$(0)
   sts = EsbGetFilterRow (hCtx, Row, szRow, pAccess)
 Loop
End Sub
```

関連トピック

- EsbGetFilterRow
- EsbListFilters
- EsbSetFilter

EsbGetFilterList

フィルタを割り当てられたユーザーのリストを取得します。

構文

```
      EsbGetFilterList
       (
      hCtx, AppName, DbName, FltName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      FltName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
FltName	フィルタ名。
pItems	このフィルタが割り当てられているユーザーのアイテムを受け取る変数のアドレス。

戻り値

正常終了の場合、このフィルタが割り当てられているユーザーのアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なユーザー名の文字列の配列が生成されます。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetFilterList Lib "ESBAPIN" (ByVal hCtx As
Long, ByVal AppName As String, ByVal DbName As String, ByVal
FilterName As String, Items As Integer) As Long

Sub ESB_GetFilterList ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Dim User As String * ESB_USERNAMELEN
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"
 FilterName = "Filter"  '****************
 ' Get Filter List
 '****************
 sts = EsbGetFilterList (hCtx, AppName, DbName, FilterName, Items)  For n = 1 To
Items   '*************************
  ' Get next User Name String
  ' from the list
  '*************************
  sts = EsbGetNextItem (hCtx,
  ESB_FUSERNAME_TYPE, ByVal User)
 Next
End Sub
```

関連トピック

- EsbGetFilter
- EsbListFilters
- EsbSetFilterList
- EsbGetNextItem

EsbGetFilterRow

フィルタの次の行を取得します。

構文

```
     EsbGetFilterRow
      (
     hCtx, FltRow, szRow, pAccess
     )
ByVal
     hCtx
       As Long
ByVal
     FltRow
      As String
ByVal
     szRow
      As Integer
ByVal
```

```
    pAccess
     As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

FltRow　　フィルタの次の行を受け取るバッファ。

szRow　　フィルタの次の行を受け取るバッファのサイズ。

pAccess　　フィルタ行のアクセス・レベルを受け取る変数のアドレス。

備考

- この関数は **EsbGetFilter()**を呼び出した後、空の文字列が戻されるまで繰り返し呼び出す必要があります。
- フィルタ行の文字列の長さがバッファのサイズより大きい場合、フィルタ行は切り捨てられます。

戻り値

正常終了の場合、次のフィルタ行(ある場合)が RowString に、そして行のアクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetFilterRow Lib "ESBAPIN" (ByVal hCtx As Long, ByVal FltRow
As String, ByVal szRow As Integer, Access As Integer) As Long
```

EsbGetFilter の例を参照してください。

関連トピック

- EsbGetFilter
- EsbListFilters

EsbGetGlobalState

システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を取得します。

構文

```
    EsbGetGlobalState
     (
    hCtx, pGlobal
    )
ByVal
```

```
hCtx
  As Long

pGlobal
 As ESB_GLOBAL_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル

pGlobal　　グローバルな状態構造体を受け取るバッファ

備考

pGlobal 構造体では、ステートメントはグローバルなセキュリティ・ステータス、グローバルなログイン・ステータス(使用可能/使用不可)、グローバルなデフォルトのアクセス・レベル、グローバルなパスワードの有効期間(日数)、グローバルな通貨使用可能フラグ、グローバルなパスワードの最小長、グローバルな自動ログアウト時間(秒単位)を戻します。

戻り値

正常終了の場合、現在のサーバーのグローバルな状態構造体の状態が pGlobal に戻されます。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    Declare Function EsbGetGlobalState Lib "ESBAPIN" (ByVal hCtx As Long, Global As
ESB_GLOBAL_T) As Long

Sub ESB_GetGlobalState ()
 Dim sts As Long
 Dim pGlobal As ESB_GLOBAL_T  '*****************
 ' Get Global State
 '*****************
 sts = EsbGetGlobalState (hCtx, pGlobal)
End Sub
```

関連トピック

- EsbSetGlobalState

EsbGetGroup

グループのセキュリティ情報が含まれている、グループ情報構造体を取得します。

構文

```
EsbGetGroup
(
```

```
      hCtx, GrpName, pUserInfo
      )
ByVal
      hCtx
         As Long
ByVal
      GrpName
       As String

      pUserInfo
       As ESB_USERINFO_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

GrpName　　グループ名。

pUserInfo　グループの情報構造体を受け取るバッファ。

戻り値

正常終了の場合、グループの情報構造体が pGroupInfo に戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbGetGroup Lib "ESBAPIN" (ByVal hCtx As Long, ByVal GroupName
As String, UserInfo As ESB_USERINFO_T) As Long

Sub ESB_GetGroup ()
 Dim sts As Long
 Dim GroupName As String
 Dim GroupInfo As ESB_USERINFO_T  GroupName = "PowerUsers"  '************************
 ' Get GroupInfo structure
 '************************
 sts = EsbGetGroup (hCtx, GroupName, GroupInfo)
End Sub
```

関連トピック

- EsbListGroups
- EsbSetGroup

EsbGetGroupList

グループのメンバーであるユーザーのリスト(またはユーザーが属するグループのリスト)を取得します。

構文

```
      EsbGetGroupList
       (
      hCtx, GrpName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

GrpName　　グループまたはユーザー名。

pItems　　ユーザー名のアイテムを受け取る変数のアドレス。

備考

- この関数を使用すると、ユーザー名を GroupName 引数として使用することによって、ユーザーが属するグループのリストを取得することもできます。

戻り値

正常終了の場合、ユーザー名のアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なユーザー名の文字列の配列が生成されます。

アクセス

この関数を使用するには、ユーザーが独自のグループ・リストを取得していないかぎり、ログインしたサーバーに対して呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbGetGroupList Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String, Items As Integer) As Long

Sub ESB_GetGroupList ()
 Dim Items As Integer
 Dim Group As String
 Dim GroupName As String * ESB_USERNAMELEN
 Dim sts As Long  Group = "User Group"  '***************
 ' Get Group List
 '***************
 sts = EsbGetGroupList (hCtx, Group, Items)  For n = 1 To Items
  '*************************
   ' Get next User Name String
   ' from the list
```

```
  '*************************
  sts = EsbGetNextItem (hCtx,
  ESB_GROUPNAME_TYPE, ByVal GroupName)
 Next
End Sub
```

関連トピック

- EsbAddToGroup
- EsbDeleteFromGroup
- EsbListGroups
- EsbSetGroupList
- EsbGetNextItem

EsbGetLocalPath

クライアント上にある特定のオブジェクト・ファイルの完全なローカル・ファイル・パスを取得します。

構文

```
      EsbGetLocalPath
       (
      hCtx, ObjType, AppName, DbName, ObjName, isCreate, Path, szPath
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String,
ByVal
      ObjName
       As String
ByVal
      isCreate
       As Integer
ByVal
      Path
        As String
ByVal
      szPath
       As Integer
```

パラメータ 説明

hCtx　　EsbCreateLocalContext()によって戻される API コンテキスト・ハンドル。

パラメータ	説明
ObjType	オブジェクト・タイプ(単一のタイプのみ)。オブジェクト・タイプのリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	オブジェクト名。
isCreate	ディレクトリ・フラグの作成。TRUE の場合は、必要に応じて適切なアプリケーションとデータベース・サブディレクトリが作成されます。FALSE の場合でディレクトリが存在しない場合は、エラーが発生します。
Path	割り当てられたローカル・パス名の文字列を受け取るバッファ。
szPath	割り当てられたローカル・パス名の文字列を受け取るバッファのサイズ。

備考

パスの文字列の長さがバッファのサイズより大きい場合、パスの文字列は切り捨てられます。

戻り値

正常終了の場合、該当するオブジェクト・ファイルのフル・パス名が Path に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetLocalPath Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String,
ByVal Create As Integer, ByVal Path As String, ByVal szPath As Integer) As Long

Sub ESB_GetLocalPath ()
  Dim sts As Long
  Dim AppName As String
  Dim DbName As String
  Dim ObjName As String
  Dim ObjType As Integer
  Dim Create As Integer
  Const szPath = 128
  Dim Path As String * szPath  AppName = "Sample"
  DbName = "Basic"
  ObjName = "Basic"
  ObjType = ESB_OBJTYPE_TEXT
  Create = ESB_YES  '***************
  ' Get local path
  '***************
  sts = EsbGetLocalPath (hCtx, ObjType, AppName, DbName, ObjName, Create,
  Path, szPath)
End Sub
```

関連トピック

- EsbCreateLocalContext
- EsbListObjects

EsbGetLocationAliasList

現在定義されているすべてのロケーション別名のリストを戻します。同時に、そのロケーション別名がマッピングされているホスト名、アプリケーション名、データベース名、およびユーザー名のリストも戻します。

構文

```
      EsbGetLocationAliasList
       (
      hCtx
      ,
      ListCount
      ,
      Aliases
      ,
      Hosts
      , _

      AppNames
      ,
      DbNames
      ,
      UserNames
      )
ByVal
      hCtx
         As Long
ByRef
      ListCount
       As Integer
ByRef
      Aliases
        As Variant
ByRef
      Hosts
         As Variant
ByRef
      AppNames
       As Variant
ByRef
      DbNames
        As Variant
ByRef
      UserNames
       As Variant
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル
ListCount	戻されたロケーション別名の数
Aliases	戻されたロケーション別名のリスト
Hosts	戻されたホストのリスト
AppNames	戻されたアプリケーションのリスト
DbNames	戻されたデータベースのリスト
UserNames	戻されたユーザー・ログインのリスト

備考

- hCtx は入力専用パラメータです。
- ListCount、Aliases、Hosts、AppNames、DbNames および UserNames は出力パラメータです。参照によって値が戻されます。

例

```
    Public Sub LocationAliasTest()

Dim status  As Long
Dim ListCount As Integer
Dim Aliases  As Variant
Dim HostNames As Variant
Dim AppNames  As Variant
Dim DbNames  As Variant
Dim UserNames As Variant

status = EsbCreateLocationAlias(hCtx, "blah1", "LocalHost", "Demo", "Basic", _
             "admin", "password")
If (status <> 0) Then
 MsgBox "Create routine Failed"
 Exit Sub
End If

status = EsbCreateLocationAlias(hCtx, "blah2", "LocalHost", "Demo", "Basic", _
              "admin", "password")
If (status <> 0) Then
 MsgBox "Create routine Failed"
 Exit Sub
End If

status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
               AppNames, DbNames, UserNames)
If (status <> 0) Then
 MsgBox "Get routine Failed"
 Exit Sub
End If

If (ListCount > 0) Then
   ' Retrieve the elements as Aliases(0) to Aliases(ListCount -1)
```

```
End If

status = EsbDeleteLocationAlias(hCtx, "blah1")
If (status <> 0) Then
 MsgBox "Delete routine Failed"
 Exit Sub
End If

status = EsbGetLocationAliasList(hCtx, ListCount, Aliases, HostNames, _
               AppNames, DbNames, UserNames)
If (status <> 0) Then
 MsgBox "Get routine Failed"
 Exit Sub
End If

End Sub
```

関連トピック

- EsbCreateLocationAlias
- EsbDeleteLocationAlias

EsbGetLogFile

アプリケーション・ログ・ファイルの一部または全部を、サーバーからクライアントにコピーします。

構文

```
      EsbGetLogFile
       (
      hCtx, AppName, TimeStamp, LocalName
      )
ByVal
      hCtx
         As Long
ByVal
      AppName
        As String
ByVal
      TimeStamp
       As Long
ByVal
      LocalName
       As String,
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル
AppName	アプリケーション名。AppName = ″ ″の場合は Essbase.log が戻されます。
TimeStamp	必要な最も古いログ・ファイル・エントリの日付と時刻を示すタイム・スタンプ

パラメータ 説明

LocalName クライアント上のローカル・コピー先ファイルのフル・パス名

備考

TimeStamp が示す時刻は、1970 年 1 月 1 日の午前 0 時(00:00:00)のグリニッジ標準時間以降に経過した秒数です。TimeStamp によって指定された日時以降に発生したログ・ファイル・エントリのみがクライアントにコピーされます。TimeStamp が 0(ゼロ)に設定されると、ログ・ファイル全体がコピーされます。

戻り値

正常終了の場合、オブジェクトは ByVal で指定されているローカル・ファイルにコピーされます。

アクセス

この関数を使用するには、指定されたアプリケーションまたはそのデータベースに対して、呼出し元がアプリケーション・デザイン権限(ESB_PRIV_APPDESIGN)またはデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetLogFile Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal TimeStamp As Long, ByVal LocalName As String) As Long

Sub ESB_GetLogFile ()
 Dim sts As Long
 Dim AppName As String
 Dim TimeStamp As Long
 Dim LocalName As String  AppName = "Sample"  '***************
 ' Get everything
 '***************
 TimeStamp = 0  LocalName = "c:\essbase\client\test.log"  '*************
 ' Get Log File
 '*************  sts = EsbGetLogFile (hCtx, AppName, TimeStamp,
 LocalName)
End Sub
```

関連トピック

- EsbDeleteLogFile

EsbGetMemberCalc

アクティブ・データベース・アウトライン内の、特定のメンバーの計算式を取得します。

構文

```
EsbGetMemberCalc
(
```

```
     hCtx, MbrName, MbrCalc, szMbrCalc, MbrLastCalc, szMbrLastCalc
     )
ByVal
     hCtx
          As Long
ByVal
     MbrName
        As String
ByVal
     MbrCalc
        As String
ByVal
     szMbrCalc
       As Integer
ByVal
     MbrLastCalc
      As String
ByVal
     szMbrLastCalc
      As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
MbrName	メンバー名。
MbrCalc	メンバーの計算文字列を受け取るバッファ。
szMbrCalc	メンバーの計算文字列を受け取るバッファのサイズ。
MbrLastCalc	メンバーの最後の計算文字列を受け取るバッファ。
szMbrLastCalc	メンバーの最後の計算文字列を受け取るバッファのサイズ。

備考

- 最後の計算文字列は、データベースを最後に計算したときにメンバーを計算するために使用した式です。計算スクリプトを使ってデータベースの計算を行った場合は、LastCalcStr のまま残る場合があります。
- Calc/LastCalc 文字列の長さがバッファのサイズより大きい場合、文字列は切り捨てられます。

戻り値

正常終了の場合、計算文字列および最後の計算文字列が CalcCtr および LastCalcStr に戻されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対して少なくとも読取りアクセス権(ESB_PRIV_READ)を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
Declare Function EsbGetMemberCalc Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
```

```
MbrName As String, ByVal Calc As String, ByVal szCalc As Integer, ByVal LastCalc As
String, ByVal szLastCalc As Integer) As Long

Sub ESB_GetMemberCalc ()
 Dim sts As Long
 Dim MbrName As String
 Const szCalc = 256
 Dim Calc As String * szCalc
 Const szLastCalc = 256
 Dim LastCalc As String * szLastCalc
 MbrName = "Year"  '****************
 ' Get Member Calc
 '****************
 sts = EsbGetMemberCalc (hCtx, MbrName, Calc, szCalc, LastCalc,
  szLastCalc)
End Sub
```

関連トピック

- EsbGetMemberInfo
- EsbSetActive

EsbGetMemberInfo

アクティブ・データベース・アウトライン内の、特定のメンバーに関する情報が含まれている構造体を取得します。

構文

```
      EsbGetMemberInfo
       (
      hCtx, MbrName, MbrInfo
      )
ByVal
      hCtx
        As Long
ByVal
      MbrName
       As String

      MbrInfo
       As ESB_MEMBERINFO_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

MbrName　　メンバー名。

MbrInfo　　メンバーの情報構造体を受け取るバッファ。

備考

属性:

- 1284 ページの「ESB_MEMBERINFO_T」構造体の Status フィールドの ESB_MBRSTS_ATTRIBUTE 定数は、次元またはメンバーが属性次元、または属性メンバーであることを示します。
- ESB_MEMBERINFO_T 構造体の次の 2 つのフィールドは、属性に対してのみ使用されます:
  - Attribute
  - IsAttributed

戻り値

正常終了の場合、この関数はメンバー情報の構造体を pMbrInfo に戻します。メンバーに親がない場合、この関数は空の文字列を ESB_MEMBERINFO_T 構造体の ParentMbrName フィールドに戻します。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbGetMemberInfo Lib "ESBAPIN" (ByVal hCtx As Long,
                     ByVal MbrName As String,
                       MbrInfo As ESB_MEMBERINFO_T) As Long

Sub ESB_GetMemberInfo ()
 Dim sts As Long
 Dim MbrName As String
 Dim MbrInfo As ESB_MEMBERINFO_T  MbrName = "Year"
 '*************************
 ' Get Member Info structure
 '*************************
 sts = EsbGetMemberInfo (hCtx, MbrName, MbrInfo)
End Sub
```

関連トピック

- EsbCheckMemberName
- EsbGetMemberCalc
- EsbQueryDatabaseMembers
- EsbSetActive

EsbGetMessage

ESB_INIT_T 構造体の ClientError が **EsbInit()**を実行した際に ESB_TRUE に設定されている場合、VB API 関数の実行中に積み上げられたメッセージ・スタックから一番上のメッセージを取得します。

構文

```
      EsbGetMessage
       (
      hInst, ErrLevel, ErrNum, ErrMessage, szErrMessage
      )
ByVal
      hInst
          As Long

      ErrLevel
        As Integer

      ErrNum
         As Long
ByVal
      ErrMessage
       As String
ByVal
      szErrMessage
      As Integer
```

パラメータ	説明
hInst	VB API インスタンス・ハンドル。
ErrLevel	メッセージ・レベルを受け取る変数へのポインタ。
ErrNum	メッセージ・データベース内のメッセージ番号を受け取る変数へのポインタ。
ErrMessage	メッセージ文字列を受け取るバッファ。
szErrMessage	メッセージ文字列を受け取るバッファのサイズ。

備考

- VB API 関数を呼び出すたびにメッセージ・スタックが初期化されます。前の呼出しのすべてのメッセージは失われます。
- 通知、警告およびエラー・メッセージを含むすべてのメッセージが、メッセージ・スタックに入れられます。
- 1 つの VB API 関数呼出しで生成されたメッセージの数が、ESB_INIT_T (またはデフォルト)によって生成される ErrorStack の設定を超える場合、古いメッセージは新規メッセージによって上書きされます。
- スタック内にメッセージがない場合、Message は空の文字列にリセットされ、pNumber と pLevel はゼロにリセットされます。
- メッセージ文字列の長さがバッファのサイズより大きい場合、メッセージは切り捨てられます。

戻り値

正常終了の場合は、メッセージ・レベルへのポインタ、メッセージ番号へのポインタおよびメッセージ文字列が戻されます。内部メッセージ・スタック・ポインタも減分されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetMessage Lib "ESBAPIN" (ByVal hInst As Long, ErrLevel As
Integer, ErrNum As Long, ByVal Message As String, ByVal szMessage As Integer) As Long

Sub ESB_GetMessage ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Const szMessage = 256
 Dim Message As String * szMessage
 Dim Number As Long
 Dim Level As Integer
 Dim sts As Long  AppName = "Demo"
 DbName = "Basic"
 FilterName = "Filter"  '****************
 ' Get Filter List
 '****************
 sts = EsbGetFilterList (hCtx, AppName, DbName,
  FilterName, Items)  '******************************
 ' Process all messages if error
 ' occurred till the bottom of the
 ' message stack enItemsered
 '******************************
 If sts > 0 Then
  sts = EsbGetMessage (hInst, Level, Number,
     Message, szMessage)
  Do While Mid$(Message, 1, 1) <> Chr$(0)
    Print Level
    Print Number
    Print Message
    sts = EsbGetMessage (hInst, Level,
       Number, Message, szMessage)
  Loop
 End If
End Sub
```

関連トピック

- EsbAutoLogin
- EsbInit

EsbGetNextItem

別の VB API 関数を呼び出して生成された配列またはリストから次のアイテムを取得します。

構文

```
      EsbGetNextItem
       (
      hCtx, dType, pItem
      )
ByVal
      hCtx
       As Long
ByVal
      dType
       As Integer
ByRef
      pItem
       As Any
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
dType	アイテムの取得元の配列/リストのタイプ: ESB_xxx...xx_TYPE、ここで、xxx...xx は、1296 ページの「ESB_USERINFO_T, ESB_GROUPINFO_T」など、グローバル定数の名前。
pItem	次のアイテムを受け取るバッファ。

備考

- この関数は、リスト/配列からアイテムを取得します。また、リスト/配列を生成する VB API 関数によって戻されるアイテム数に基づいた FOR ループの中で呼び出される必要があります。
- pItem は、必要なデータ型のバッファです。

戻り値

正常終了の場合は、pItem にアイテムを戻します。失敗した場合、-1 を戻して Type が適切でないことを示します。または、1 を戻して、リストにアイテムがこれ以上ないことを示します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetNextItem Lib "ESBAPIN" (ByVal hCtx As Long, ByVal dType
As Integer, pItem As Any)

Sub ESB_GetNextItem ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Dim User As String * ESB_USERNAMELEN
 Dim sts As Long  AppName = "Demo"
 DbName = "Basic"
```

```
 FilterName = "Filter"  '****************
 ' Get Filter List
 '****************
 sts = EsbGetFilterList (hCtx, AppName, DbName,
 FilterName, Items)  '************************
 ' Print out all user names
 ' from the list
 '************************
 For n = 1 To Items   '*************************
  ' Get next User Name String
  ' from the list
  '*************************   sts = EsbGetNextItem (hCtx,
     ESB_USERINFO_TYPE, Userinfo)
     Print User
 Next
End Sub
```

EsbGetNextItem のグローバル定数

定数	コンテキスト
ESB_USERINFO_TYPE = 1	ESB_USERINFO_T (EsbListUsers)
ESB_GROUPINFO_TYPE = 2	ESB_USERINFO_T (EsbListGroups)
ESB_USERAPP_TYPE = 3	ESB_USERAPP_T (EsbGetApplicationAccess)
ESB_USERDB_TYPE = 4	ESB_USERDB_T (EsbGetDatabaseAccess)
ESB_LOCKINFO_TYPE = 5	ESB_LOCKINFO_T (EsbListLocks)
ESB_OBJINFO_TYPE = 6	ESB_OBJINFO_T (EsbListObjects)
ESB_APPDB_TYPE = 7	ESB_APPDB_T (EsbListDatabases)
ESB_CAPPDB_TYPE = 8	ESB_APPDB_T (EsbListCurrencyDatabase)
ESB_APPNAME_TYPE = 9	ByVal var As String * ESB_APPNAMELEN (EsbListApplications)
ESB_DBNAME_TYPE = 10	ByVal var As String * ESB_DBNAMELEN (EsbGetApplicationInfo)
ESB_GROUPNAME_TYPE = 11	ByVal var As String * ESB_USERNAMELEN (EsbGetGroupList)
ESB_FTRNAME_TYPE = 12	ByVal var As String * ESB_FTRNAMELEN (EsbListFilters)
ESB_FUSERNAME_TYPE = 13	ByVal var As String * ESB_USERNAMELEN (EsbGetFilterList)
ESB_OBJNAME_TYPE = 14	ByVal var As String * ESB_OBJNAMELEN (EsbGetCalcList)
ESB_DIMSTATS_TYPE = 15	ESB_DIMSTATS_T (EsbGetDatabaseStats)
ESB_CUSERINFO_TYPE = 16	ESB_USERINFO_T (EsbListConnections)
ESB_LAPPDB_TYPE = 17	ESB_APPDB_T (EsbLogin)
ESB_ALIASNAME_TYPE = 18	ESB_ALIASNAME_T (EsbListAliases)

定数	コンテキスト
ESB_MBRALT_TYPE = 19	ESB_MBRALT_T (EsbDisplayAlias)
ESB_RATEINFO_TYPE = 20	ESB_RATEINFO_T (EsbGetCurrencyRateInfo)
ESB_OUTLINEINFO_TYPE = 21	ByVal var As String * ESB_ALIASNAMELEN (EsbOtlGetOutlineInfo)
ESB_OUTERROR_TYPE = 22	ESB_OUTERROR_T (EsbOtlVerifyOutline)
ESB_OTLUSERATTR_TYPE = 23	ByVal var As String * ESB_MBRNAMELEN (EsbOtlGetUserAttributes)
ESB_APPINFOEX_TYPE = 24	ESB_APPINFOEX_T (EsbGetApplicationInfoEx)
ESB_DBREQINFO_TYPE = 25	ESB_DBREQINFO_T (EsbGetDatabaseInfo)
ESB_DIMINFO_TYPE = 26	ESB_DIMENSIONINFO_T (EsbGetDimensionInfo)
ESB_HMEMBER_TYPE = 27	ESB_HMEMBER_T (EsbOtlQueryMembers)
ESB_GENLEVELNAME_TYPE = 28	ESB_GENLEVELNAME_T (EsbOtlGetGenNames)
ESB_VARIABLE_TYPE = 29	ESB_VARIABLE_T (EsbListVariables)
ESB_LRO_TYPE = 30	ESB_LRODESC_T
ESB_PART_INFO_TYPE = 31	ESB_PART_INFO_T
ESB_DTS_TYPE = 32	ESB_DTSMBRINFO_T (EsbGetEnabledDTSMembers)
ESB_MBRNAME_TYPE = 33	ESB_MBRNAME_T (EsbOtlGetDimensionUserAttributes)

関連トピック

- EsbListUsers
- EsbListGroups
- EsbGetApplicationAccess
- EsbGetDatabaseAccess
- EsbListLocks
- EsbListObjects
- EsbListDatabases
- EsbListCurrencyDatabases
- EsbListApplications
- EsbGetApplicationInfo
- EsbGetApplicationInfoEx
- EsbGetGroupList
- EsbListFilters
- EsbGetFilterList
- EsbGetCalcList
- EsbGetDatabaseState
- EsbGetDatabaseStats
- EsbListConnections
- EsbLogin
- EsbListAliases

- EsbDisplayAlias
- EsbGetCurrencyRateInfo
- EsbLROGetMemberCombo
- EsbOtlQueryMembersByName

EsbGetObject

サーバーまたはクライアントのオブジェクト・システムからローカル・ファイルにオブジェクトをコピーし、オプションでロックします。

構文

```
      EsbGetObject
       (
      hCtx, ObjType, AppName, DbName, ObjName, LocalName, isLock
      )
ByVal
      hCtx
         As Long
ByVal
      ObjType
        As Long
ByVal
      AppName
        As String
ByVal
      DbName
         As String
ByVal
      objName
        As String
ByVal
      LocalName
       As String
ByVal
      isLock
        As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。ビットマスク・データ型に関する説明を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	取得するオブジェクト名。
LocalName	クライアント上のローカル・コピー先ファイルのフル・パス名。

パラメータ 説明

isLock	オブジェクトのロックを制御するフラグ。TRUE の場合は、オブジェクトがロックされ、他のユーザーによる更新を防止します。

備考

オブジェクトをロックするには、そのオブジェクトがサーバーに存在している必要があり、かつ他のユーザーによってロックされていないことが必要です。クライアント上でのロックはサポートされていません。

戻り値

正常終了の場合は、オブジェクトは LocalName で指定されているローカル・ファイルにコピーされます。

アクセス

この関数を使用するには、呼出し元は、オブジェクトがある指定したコピー元アプリケーションまたはデータベース(あるいはその両方)に対して、適切なレベルのアクセス権(オブジェクト・タイプによって異なる)を持っている必要があります。オブジェクトをロックするには(ロック・フラグが TRUE)、呼出し元は、指定したコピー先アプリケーション、またはオブジェクトが含まれているデータベースに対して、アプリケーションまたはデータベース・マネージャの権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbGetObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType As
Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String,
ByVal LocalName As String, ByVal Lock As Integer)
 As Long

 Sub ESB_GetObject ()
  Dim sts As Long
  Dim AppName As String
  Dim DbName As String
  Dim ObjName As String
  Dim ObjType As Long
  Dim LocalName As String
  Dim Lock As Integer  AppName = "Sample"
  DbName = "Basic"
  ObjName = "Basic"
  ObjType = ESB_OBJTYPE_OUTLINE
  LocalName = "C:\ESSBASE\CLIENT\BASIC.OTL"
  Lock = ESB_YES  '***********
  ' Get Object
  '***********
  sts = EsbGetObject (hCtx, ObjType, AppName,
     DbName, ObjName, LocalName, Lock)
End Sub
```

関連トピック

- EsbGetObjectInfo

- EsbListObjects
- EsbLockObject
- EsbPutObject
- EsbUnlockObject

EsbGetObjectInfo

サーバー上またはローカルのクライアント上にある特定のオブジェクトに関する情報を取得します。

構文

```
      EsbGetObjectInfo
       (
      hCtx, ObjType, AppName, DbName, ObjName, ObjInfo
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      objName
       As String

      ObjInfo
       As ESB_OBJINFO_T
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストはビットマスク・データ型を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	オブジェクト名。
ObjInfo	オブジェクトの情報構造体を受け取るバッファ。

戻り値

正常終了の場合、該当するオブジェクトに関する情報を含むオブジェクト構造体が pObject に戻されます。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベース(あるいはその両方)に対して、呼出し元が(オブジェクト・タイプに応じて)適切なレベルのアクセス権を持っている必要があります。

例

```
    Declare Function EsbGetObjectInfo Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
ObjType As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As
String, ObjInfo As ESB_OBJINFO_T) As Long

Sub ESB_GetObjectInfo ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer
 Dim Object As ESB_OBJINFO_T  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
 ObjType = ESB_OBJTYPE_OUTLINE  '**************************
 ' Get Object info structure
 '**************************
 sts = EsbGetObjectInfo (hCtx, ObjType, AppName,
    DbName, ObjName, Object)
End Sub
```

関連トピック

- EsbGetObject
- EsbListObjects
- EsbCreateLocalContext

EsbGetProcessState

計算またはデータ・インポートなどの、非同期プロセスの現在の状態を取得します。

構文

```
      EsbGetProcessState
       (
      hCtx, ProcState
      )
ByVal
      hCtx
         As Long
```

```
ProcState
 As ESB_PROCSTATE_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

ProcState　プロセス状態構造体へのポインタ

備考

- pProcState に ESB_STATE_DONE が戻されるまで、プログラムでこの関数を定期的(5 秒-10 秒間隔)に呼び出す必要があります。
- 非同期データベース操作(たとえば計算)が正しく開始される前にこの関数を呼び出すと、エラーが発生します。

戻り値

正常終了の場合、現在のプロセス状態が状態構造体の pProcState に戻されます。
pProcState の値:

- ESB_STATE_DONE: 0 = 完了
- ESB_STATE_INPROGRESS: 1 = 進行中
- ESB_STATE_FINALSTAGE: 5 = 最終段階。取消しできません

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetProcessState Lib "ESBAPIN" (ByVal hCtx As Long, ProcState
As ESB_PROCSTATE_T) As Long
```

EsbBeginCalc、EsbCalc および EsbImport の例を参照してください。

関連トピック

- EsbBeginCalc
- EsbCalc
- EsbCancelProcess
- EsbImport

EsbGetString

アクティブ・データベースから文字列データを取得します。

構文

```
EsbGetString
 (
hCtx, getString, szString
)
```

```
ByVal
      hCtx
         As Long
ByVal
      getString
       As String
ByVal
      szString
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
getString	戻されたデータ文字列を受け取るバッファ。サーバーによって生成される最大文字列以上のサイズが必要です。最大文字列サイズは 256KB です。
szString	戻されたデータ文字列を受け取るバッファのサイズ。

備考

- データが戻された場合、**EsbReport()**、**EsbEndReport()**、**EsbQueryDatabaseMember()**の後に、この関数を呼び出してください。
- 空の文字列が戻されるまで、つまり戻すデータがまったくなくなるまで、繰り返しこの関数を呼び出す必要があります。
- このコマンドを使用するときは必ず改行してください。改行しないと、エラーになります。
- レポートが正常に実行される前にこの関数を呼び出すと、エラーが発生します。
- 戻される文字列の長さは、64KB 未満です。
- getString の長さがバッファより大きいと、文字列は切り捨てられます。
- 戻される文字列が 256 バイトより小さい場合、残りの空のバイト列は NULL 文字で満たされます。

戻り値

データ文字列は getString に戻されます。この関数は、戻すデータがなくなった場合に空の文字列バッファを戻します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetString Lib "ESBAPIN" (ByVal hCtx As Long, ByVal getString
As String, ByVal szString As Integer) As Long
```

EsbReport および EsbQueryDatabaseMembers の例を参照してください。

関連トピック

- EsbGetStringBuf

- EsbReport
- EsbEndReport
- EsbQueryDatabaseMembers

EsbGetStringBuf

アクティブ・データベースから入手可能なデータがすべて戻されるまで、または呼出し元のバッファがいっぱいになるまでデータを取得します。

構文

```
      EsbGetStringBuf
       (
      hCtx, getString, szString
      )
ByVal
      hCtx
         As Long
ByVal
      getString
       As String
ByVal
      szString
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
getString	戻されたデータ文字列を受け取るバッファ。最大バッファ・サイズは 64K です。
szString	戻されたデータ文字列を受け取るバッファのサイズ。

備考

- データが戻された場合に、**EsbReport()**、**EsbEndReport()**または**EsbQueryDatabaseMember()**の後にこの関数を呼び出します。
- レポートが正常に実行される前にこの関数を呼び出すと、エラーが発生します。
- この関数と **EsbGetString()**の違いは、**EsbGetString()**は 1 行ずつデータを戻すことです。大規模なデータ・セットに対しては **EsbGetString()**を使用してください。

戻り値

この関数は getString の 1 つ以上のデータ文字列を戻します。データがなくなると、空の文字列バッファを戻します。

この関数はバッファが収容できるすべてのデータを戻します。これによってレコードの一部を取得し、バッファの最後にこのような部分的なレコードを取得する場合もあります。次回 **EsbGetStringBuf()**を呼び出すと、バッファの最初に残りのレコード部分が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
   Declare Function EsbGetStringBuf Lib "ESBAPIN"(
 ByVal hCtx As Long,
 ByVal getString As String,
 ByVal szString As Integer)
 As Long
```

EsbReport および EsbQueryDatabaseMembers の例を参照してください。

関連トピック

- EsbGetString
- EsbReport
- EsbEndReport
- EsbQueryDatabaseMembers

EsbGetUser

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を取得します。

構文

```
      EsbGetUser
       (
      hCtx, userName, pUserInfo
      )
ByVal
      hCtx
         As Long
ByVal
      userName
       As String

      pUserInfo
       As ESB_PUSERINFO_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

userName　ユーザー名。

pUserInfo　ユーザーの情報構造体を受け取るバッファ。

戻り値

正常終了の場合は、ユーザーの情報構造体が pUserInfo に戻されます。

アクセス

この関数を使用するには、ユーザーが独自のユーザー情報を取得していないかぎり、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbGetUser Lib "ESBAPIN" (ByVal hCtx As Long,
ByVal User As String, UserInfo As ESB_USERINFO_T) As Long

Sub ESB_GetUser ()
 Dim sts As long
 Dim User As String
 Dim UserInfo As ESB_USERINFO_T  User = "Joseph"  '************************
 ' Get User Info structure
 '************************
 sts = EsbGetUser (hCtx, User, UserInfo)
End Sub
```

関連トピック

- EsbGetApplicationAccess
- EsbListUsers
- EsbSetUser

EsbGetUserEx

ユーザーのセキュリティ情報を含むユーザー情報構造体を取得します。

構文

```
     Declare Function EsbGetUserEx Lib "esbapin" (
ByVal hCtx As Long,
ByVal userName As String,
   pUserInfo As ESB_USERINFOEX_T) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
UserName	ユーザー名。
pUserInfoEx	外部認証ユーザーの情報構造体を受け取るポインタのアドレス。

備考

この関数は EsbGetUser と同様に動作します。違いは、この関数は拡張ユーザー情報構造体 ESB_USERINFOEX_T を戻すことです。

戻り値

正常終了の場合は、ユーザーの情報構造体が pUserInfo に戻されます。

アクセス

この関数では、呼出し元が自分自身のユーザー情報を取得するのでないかぎり、
ログインしているサーバーに対するユーザーの作成/削除権限
(ESS_PRIV_USERCREATE)を持っている必要があります。

EsbGetVariable

代替変数の値を取得します。

構文

```
      EsbGetVariable
       (
      hCtx, pVariable
      )
ByVal
      hCtx
         As Long

      pVariable
       As ESB_PVARIABLE_T
```

パラメータ 説明

hCtx　　API へのコンテキスト・ハンドル。

pVariable　指定された代替変数の説明を含む構造体を指すポインタ。

戻り値

正常終了の場合、**EsbGetVariable**()により構造体 ESB_VARIABLE_T の VarValue
フィールドに代替変数の値が戻されます。

例

```
    Declare Function EsbGetVariable Lib "esbapin" (ByVal hCtx As Long, pVariable As
ESB_VARIABLE_T) As Long

Sub Esb_GetVariable ()

Dim sts As Long
Dim oVariable As ESB_VARIABLE_T

'  Get value of "QuarterName" Susbtitution Variable at the Sample application level
  oVariable.Server = "Localhost"
  oVariable.AppName = "Sample"
'  ** Note that DbName has been left empty
  oVariable.VarName = "QuarterName"

  sts = EsbGetVariable(hCtx, oVariable)

  MsgBox oVariable.VarValue
```

```
End Sub
```

関連トピック

- 1300 ページの「ESB_VARIABLE_T」
- EsbCreateVariable
- EsbDeleteVariable
- EsbListVariables

EsbGetVersion

接続されている Essbase サーバーの完全なバージョン番号を、3.0.0 のようなリリース.バージョン.改訂の形式で取得します。

構文

```
      EsbGetVersion
       (
      hCtx, Release, Version, Revision
      )
ByVal
      hCtx
        As Long

      Release
       As Integer

      Version
       As Integer

      Revision
       As Integer
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。

Release　リリース番号を受け取る変数のアドレス。

Version　バージョン番号を受け取る変数のアドレス。

Revision　改訂番号を受け取る変数のアドレス。

備考

Essbase サーバー・バージョンがプログラムで使用されているすべての機能をサポートすることを確認するため、サーバーに接続してからこの関数を呼び出せます。

戻り値

正常終了の場合、完全な Essbase サーバーのバージョン番号が pRelease、pVersion および pRevision の形式で戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbGetVersion Lib "ESBAPIN" (ByVal hCtx As Long, Release As
Integer, Version As Integer, Revision As Integer) As Long

Sub ESB_GetVersion ()
 Dim sts As Long
 Dim Release As Integer
 Dim Version As Integer
 Dim Revision As Integer  '************
 ' Get Version
 '************
 sts = EsbGetVersion (hCtx, Release, Version, Revision)
End Sub
```

EsbImport

様々なソースから Essbase サーバーへのデータのインポートを許可します。

構文

```
      EsbImport
       (
      hCtx, pRules, pData, User, ErrName, isAbortOnError
      )
ByVal
      hCtx
           As Long

      pRules
          As ESB_OBJDEF_T

      pData
           As ESB_OBJDEF_T

      User
           As ESB_MBRUSER_T
ByVal
      ErrName
         As String
ByVal
      isAbortOnError
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
pRules	ルール・ファイル・オブジェクト定義構造体へのポインタ。

パラメータ	説明
pData	データ・ファイル・オブジェクト定義構造体へのポインタ。
User	SQL ユーザー構造体へのポインタ(データ・ソースが SQL データベースの場合)。SQL ユーザー構造体が NULL の場合は、SQL 以外のデータ・ソースを示します。
ErrName	ローカルに作成されるエラー出力ファイルの名前。
isAbortOnError	TRUE の場合、最初のエラーでインポートは停止され、それ以外の場合は続行します。

備考

- 非 SQL ソースの場合で、pRules および pData に対する ESB_OBJDEF_T 構造体の AppName および DbName フィールドが空の文字列の場合は、hCtx がローカル・コンテキスト・ハンドルで、かつファイルへの完全修飾パスが ESB_OBJDEF_T の FileName フィールドに含まれている必要があります。
- ローカル・オブジェクトが使用されている場合、最初に **EsbCreateLocalContext()** を呼び出す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定されたデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbImport Lib "ESBAPIN" (ByVal hCtx As Long, Rules As
ESB_OBJDEF_T,
                   Data As ESB_OBJDEF_T, User As ESB_MBRUSER_T,
                   ByVal ErrName As String, ByVal AbortOnError As Integer)
                   As Long
Sub ESB_Import ()
 Dim sts      As Long
 Dim Rules     As ESB_OBJDEF_T
 Dim Data     As ESB_OBJDEF_T
 Dim User     As ESB_MBRUSER_T
 Dim ErrorName   As String
 Dim AbortOnError As Integer   '*********************************
 ' Rules file resides at the server
 '*********************************
 Rules.hCtx   = hCtx
 Rules.Type   = ESB_OBJTYPE_RULES
 Rules.AppName  = "Demo"
 Rules.DbName  = "Basic"
 Rules.FileName = "Test"

 '*********************************
 ' Data file resides at the server
 '*********************************
 Data.hCtx    = hCtx
 Data.Type    = ESB_OBJTYPE_TEXT
 Data.AppName  = "Demo"
```

```
Data.DbName   = "Basic"
Data.FileName  = "Data"        '*********************************
' Specify file to redirect errors
' to if any
'*********************************
ErrorName    = "IMPORT.ERR"  '************************
' Abort on the first error
'************************
AbortOnError  = ESB_YES  '*******
' Import
'*******
sts      = EsbImport (hCtx, Rules, Data, User, ErrorName, AbortOnError)
'*************************************************************************
'*
'* When a SQL data source is defined in the rules file, define
'* the variables in the ESB_OBJDEF_T Data structure as follows:
'*   Data.hCtx    = hCtx
'*   Data.AppName  = ""
'*   Data.DbName   = ""
'*   Data.ObjType  = ESB_OBJTYPE_NONE
'*   Data.FileName = ""
'*
'* Also, provide strings for the variables in the ESB_MBRUSER_T
'* User structure; for example:
'*   User.User    = "Dbusernm"
'*   User.Password = "Dbpasswd"
'*
'* Use a blank string for User and Password, if the SQL source
'* does not require user and password information; for example:
'*   User.User    = ""
'*   User.Password = ""
'*
'* Call the function as follows:
'*   sts      = EsbImport (hCtx, Rules, Data, User, AbortOnError)
'*
'************************************************************************* End Sub
```

関連トピック

- EsbExport
- EsbBuildDimension

EsbInit

VB API とメッセージ・データベースを初期化します。この関数に渡される ESB_INIT_T 構造体は、いくつかの初期化パラメータを含んでいます。含まれている初期化パラメータは、メッセージ・データベースの名前、カスタマイズされたエラー・ハンドラを使用するかどうかを示すフラグ、このエラー・ハンドラで使用されるメッセージ・スタックの最大サイズ、ヘルプ・ファイルの名前と場所およびバージョン番号などです。

構文

```
EsbInit
```

```
 (
pInit, phInst
)

pInit
 As ESB_INIT_T

phInst
 As Long
```

パラメータ 説明

pInit　　VB API 初期化構造体へのポインタ。

phInst　　VB API インスタンス・ハンドルへのポインタ。

備考

- この関数は、他のすべての VB API 関数よりも前に呼び出す必要があります。
- 初期化構造体のいずれかのフィールドが空の文字列またはゼロの場合(適切な場合)、API ではこれらのパラメータに対してデフォルトの値が使用されます。
- ESB_TRUE および ESB_FALSE はグローバル変数です。この例で紹介している整数値を割り当てます。

戻り値

この関数に渡される ESB_INIT_T 構造体は、いくつかの初期化パラメータを含んでいます。含まれている初期化パラメータは、メッセージ・データベースの名前、エラー・ハンドラ、ヘルプ・ファイルの名前と場所およびバージョン番号などです。

EsbInit()は、phInst にインスタンス・ハンドルを戻します。戻されたインスタンス・ハンドルによって、複数のアプリケーションが VB API に個別にアクセスできるようになります(DLL の場合のみ)。インスタンス・ハンドルを保持し、**EsbLogin()** および **EsbTerm()**関数に渡す必要があります。

アクセス

この関数には、特別なアクセス権は必要ありません。

例

```
    Declare Function EsbInit Lib "ESBAPIN.DLL" (Init As ESB_INIT_T, hInst As Long) As
Long

Sub ESB_Init ()
 Dim hInst As Long
 Dim Init As ESB_INIT_T
 Dim sts As Long  ESB_FALSE = 0
 ESB_TRUE  = 1  '**********************
 ' Define init structure
 '**********************
 Init.Version = ESB_API_VERSION
 Init.MaxHandles = 10
 Init.LocalPath = "C:\ESSBASE"
```

```
 ' Use default message file
 Init.MessageFile =""
 ' Use EsbGetMessage to retrieve
 ' messages
 Init.ClientError = ESB_TRUE
 Init.ErrorStack = 100   '*******************
 ' Initialize the API
 '*******************
 sts = EsbInit (Init, hInst)
End Sub
```

関連トピック

- EsbLogin
- EsbAutoLogin
- EsbTerm
- EsbGetMessage

EsbKillRequest

特定のユーザー・セッションまたは要求を終了します。

構文

```
      EsbKillRequest (hCtx, ReqInfo)
 ByVal hCtx   As Long
 ByVal pReqInfo  As ESB_REQUESTINFO_T
```

パラメータ 説明

hCtx　　コンテキスト・ハンドル

pReqInfo　要求情報構造体を指すポインタ。

備考

- **EsbKillRequest**()は ESB_REQUESTINFO_T 内の現行セッションに関する情報を使用して、特定のユーザー・セッションの終了を要求します。この関数は、ユーザー・セッション中に、アプリケーション、データベースまたはシステムに対して行われているアクティブな要求を終了(ユーザーをログアウトせずに)する場合にも使用できます。
- セッションとは、ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。
- 要求とは、ユーザーまたは他のプロセスが Essbase サーバーに対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。
- この関数は、ESB_REQUESTINFO_T 構造体の「UserName」、「AppName」、「DbName」で指定されたセッションおよび要求を終了します。これらのフィールドが NULL の場合、この関数はこのプロセス(ユーザー)によって起動された

すべてのセッションと要求を終了します。アプリケーション・プログラムは、ESB_REQUESTINFO_T によって使用されるメモリーの割当てと解放を行います。

戻り値

正常終了の場合は、ユーザー数が Items に戻され、指定したアプリケーションおよびデータベースに対してアクセス権を持っているユーザーのリストが生成されます。このリストにアクセスするには、**EsbGetNextItem()**を使用します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbKillRequest Lib "ESAPINW" (ByVal hCtx As Long, pReqInfo As
ESB_REQUESTINFO_T) As Long

 Sub ESB_KillRequest()
  Dim Items As Integer
  Dim ReqInfo As ESB_REQUESTINFO_T
  Dim sts As Long
  Dim pAccess As Integer

  '*********
  ' List Requests
  '*************
  'sts = EsbSetActive(hCtx, AppName, DbName, pAccess)
  'Debug.Print "EsbSetActive = " & sts
  'sts = EsbDefaultCalc(hCtx)
  'Debug.Print "EsbDefaultCalc = " & sts
   sts = EsbListRequests(hCtx, UserName, AppName, DbName, Items)
    Debug.Print "EsbListRequests = " & sts & " " & Items
  For n = 1 To Items
   '********************
   ' Get next Request Info
   ' from the list
   '********************
   sts = EsbGetNextItem(hCtx, ESB_REQUESTINFO_TYPE, ReqInfo)
   Debug.Print "EsbGetNextItem = " & sts & " " & ReqInfo.LoginId & " " &
ReqInfo.DbRequestCode
   sts = EsbKillRequest(hCtx, ReqInfo)
   Debug.Print "EsbKillRequest = " & sts
  Next
End Sub
```

関連トピック

- EsbListRequests

EsbListAliases

アクティブなデータベース内にある別名テーブルの名前をすべてリストします。

構文

```
      EsbListAliases
       (
      hCtx, pItems
      )
ByVal
      hCtx
        As Long

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pItems　　別名テーブルのアイテムを受け取る変数のアドレス。

戻り値

正常終了の場合は、別名テーブルのアイテムが pItems に戻され、**EsbGetNextItem()** を介してアクセス可能な別名テーブルの名前の配列が生成されます。

アクセス

この関数を使用するには、呼出し元がデータベースに対するアクセス権を持っており、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbListAliases Lib "ESBAPIN" (ByVal hCtx As Long, Items As
Integer) As Long

Sub ESB_ListAliases ()
 Dim Items As Integer
 Dim AliasName As String * ESB_ALIASNAMELEN
 Dim sts As Long
 '*************
 ' List Aliases
 '*************
 sts = EsbListAliases (hCtx, Items)  For n = 1 To Items
  '********************
  ' Get next Alias Name
  ' from the list
  '********************
  sts = EsbGetNextItem (hCtx,
     ESB_ALIASNAME_TYPE, ByVal AliasName)
 Next
End Sub
```

関連トピック

- EsbDisplayAlias

- EsbGetNextItem

EsbListApplications

呼出し元がアクセスできる、すべてのアプリケーションをリストします。

構文

```
      EsbListApplications
       (
      hCtx, pItems
      )
ByVal
      hCtx
        As Long

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pItems　　戻されたアプリケーションのアイテムを受け取る変数のアドレス。

戻り値

正常終了の場合、アクセス可能なアプリケーション数のアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なアプリケーション名の文字列のリストが生成されます。リスト内のアイテムの「アイテム」数があります。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアプリケーションにアクセスした場合、サーバー・アプリケーションのみがリストされることに注意してください。

例

```
    Declare Function EsbListApplications Lib "ESBAPIN" (ByVal hCtx As Long, Items As
Integer) As Long

Sub ESB_ListApplications ()
 Dim sts As Long
 Dim Items As Integer
 Dim AppName As String * ESB_APPNAMELEN  '*******************************
 ' Get List of Application names
 '*******************************
 sts = EsbListApplications (hCtx, Items)  For n = 1 To Items
  '*********************************
   ' Get next Application name string
   '*********************************
  sts = EsbGetNextItem (hCtx,
     ESB_APPNAME_TYPE, ByVal AppName)
```

```
 Next
End Sub
```

関連トピック

- EsbListDatabases
- EsbListObjects
- EsbGetNextItem

EsbListCalcFunctions

アクティブなアプリケーションで使用可能なすべての計算関数をリストします。これにはすべてのネイティブ関数と、カスタム定義関数(CDF)およびカスタム定義マクロ(CDM)が含まれます。

構文

```
     Declare Function EsbListCalcFunctions Lib "esbapin" (
ByVal hCtx As Long,
ByVal CalcString As String,
ByVal szString As Integer) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
CalcString	使用可能な計算関数を含む文字列。この文字列は XML 形式です。
szString	使用可能な計算関数を含む文字列のサイズ。

備考

EsbListCalcFunctions()の実行には、管理者権限が必要です。ユーザーがこのリストを入手するためには、データベースへのアクセス権も必要です。エラーを避けるため、データベースにアクセスして EsbListCalcFunctions()でプログラムを実行できるように、ユーザーは管理者の権限とデータベースへのアクセス権の両方を持っている必要があります。

EsbGetCalcList によって戻される文字列のコンテンツは XML でフォーマットされ、XML ユーティリティでレンダリング、または構文解析によって実際のテキストのみ表示する必要があります。すべての XML タグは山カッコで囲まれています(たとえば<xml_tag>)。

典型的な XML 出力ファイルを短縮した例:

```
     ESSBASE API v.62000
1051034: Logging in user admin
1051035: Last login on Tuesday, May 22, 2001 10:31:19 AM
<list>
<group name="Boolean">
<function>
<name><![CDATA[@ISACCTYPE]]
     ></name>
```

```
<syntax>
<![CDATA[@ISACCTYPE(tag)]]
      >
</syntax>
<comment>
<![CDATA[returns TRUE if the current member has the associated accounts tag]]
      >
</comment>
</function>
</group>
<group name="Relationship Functions">
<function>
<name><![CDATA[@ANCESTVAL]]
      ></name>
<syntax>
<![CDATA[@ANCESTVAL (dimName, genLevNum [, mbrName])]]
      >
</syntax>
<comment>
<![CDATA[returns the ancestor values of a specified member combination]]
      >
</comment>
</group>
<group name="Custom">
</group></list>
```

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

EsbListConnections

現在ログインしているサーバーまたはアプリケーションに接続されているユーザーをすべてリストします。

構文

```
      EsbListConnections
       (
      hCtx, pItems
      )
ByVal
      hCtx
        As Long

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pItems　　ユーザーのアイテムを受け取る変数。

備考

- hCtx が管理者の場合、pItems にはサーバーにログインしているユーザー数が入ります。hCtx がアプリケーション・マネージャの場合、pItems には hCtx がアプリケーション・マネージャになっているアプリケーションに接続しているユーザー数が入ります。
- **EsbGetNextItem()**を各ユーザーに対して 1 回呼び出します(pItems 変数で戻される)。**EsbGetNextItem()**への呼出しはそれぞれ、ESB_USERINFO_T 構造体にユーザー情報を戻します。

戻り値

成功の場合、0 が戻されます。

アクセス

この関数を使用するには、呼出し元が管理者またはアプリケーション・マネージャ権限を持っている必要があります。

例

```
    Declare Function EsbListConnections Lib "ESBAPIN" (ByVal hCtx As Long, Items As
Integer) As Long
Sub ESB_ListConnections()
 Dim Items As Integer
 Dim UserInfo As ESB_USERINFO_T
 Dim sts As Long
 '*****************
 ' List Connections
 '*****************
 sts = EsbListConnections(hCtx, Items)
 For n = 1 To Items
  '****************************
  ' Get next User Info structure
  ' from the list
  '****************************
  sts = EsbGetNextItem(hCtx,
     ESB_USERINFO_TYPE, UserInfo)
 Next
End Sub
```

関連トピック

- EsbListLocks
- EsbListUsers
- EsbGetNextItem

EsbListCurrencyDatabases

呼出し元がアクセス可能な、特定のアプリケーション内のすべての通貨データベースをリストします。

構文

```
      EsbListCurrencyDatabases
       (
      hCtx, AppName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

pItems　　通貨データベースのアイテムを受け取る変数のアドレス。

備考

この関数は、クライアントではなくサーバーのアプリケーション内部の通貨データベースをリストする場合にのみ使用できます。

戻り値

正常終了の場合は、アクセス可能な通貨データベース数のアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なアプリケーション名/通貨データベース名のリストが生成されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアクセスする場合、通貨データベースのみがリストされることに注意してください。

例

```
    Declare Function EsbListCurrencyDatabases Lib "ESBAPIN" (ByVal hCtx As Long,
ByVal AppName As String, Items As Integer) As Long

Sub ESB_ListCurrencyDatabases ()
 Dim Items As Integer
 Dim AppName As String
 Dim AppDb As ESB_APPDB_T
 Dim sts As Long  AppName = "Sample"  '************************
 ' List Currency Databases
 '************************
 sts = EsbListCurrencyDatabases (hCtx, AppName,
    Items)  For n = 1 to Items   '******************************
  ' Get next Application/Database
  ' item from the list
```

```
   '*****************************
   sts = EsbGetNextItem (hCtx,
      ESB_CAPPDB_TYPE, AppDb)
 Next
End Sub
```

関連トピック

- EsbGetDatabaseInfo
- EsbGetDatabaseState
- EsbListApplications
- EsbListDatabases
- EsbListObjects
- EsbGetNextItem

EsbListDatabases

呼出し元がアクセス可能な、特定のアプリケーション内またはサーバー全体の、すべてのデータベースをリストします。

構文

```
      EsbListDatabases
       (
      hCtx, AppName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
pItems	アプリケーションおよびデータベースのカウントを受け取る変数のアドレス。

備考

AppName の引数が空の文字列の場合、この関数はサーバー上のアクセス可能なアプリケーションおよびデータベースをすべてリストします。

戻り値

正常終了の場合、アクセス可能なデータベースの数が pCount に戻され、**EsbGetNextItem()**を介してアクセス可能なアプリケーション名およびデータベース名のリストが生成されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアクセスする場合、サーバー・データベースのみがリストされることに注意してください。

例

```
    Declare Function EsbListDatabases Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, Count As Integer) As Long

Sub ESB_ListDatabases ()
 Dim pItems As Integer
 Dim AppName As String
 Dim AppDb As ESB_APPDB_T
 Dim sts As Long  AppName = "Sample"  '***************
 ' List Databases
 '***************
 sts = EsbListDatabases (hCtx, AppName, pItems)  For n = 1 To pItems
  '*****************************
  ' Get next Application/Database
  ' item from the list
  '*****************************
  sts = EsbGetNextItem (hCtx,
     ESB_APPDB_TYPE, AppDb)
 Next
End Sub
```

関連トピック

- EsbGetDatabaseInfo
- EsbGetDatabaseState
- EsbListApplications
- EsbListCurrencyDatabases
- EsbListObjects
- EsbGetNextItem

EsbListDbFiles

指定したインデックスおよびデータ・ファイルに関する情報を取得します。

構文

```
EsbListDbFiles
 (
hCtx
,
AppName
,
DbName
,
FileType
,
Items
```

```
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FileType
       As Integer

      Items
        As Long
```

パラメータ 説明

hCtx　コンテキスト・ハンドル

AppName　アプリケーション名

DbName　データベース名

FileType　戻される次のファイル・タイプのいずれかになります:

- ESB_FILETYPE_INDEX
- ESB_FILETYPE_DATA
- ESB_FILETYPE_INDEX | ESB_FILETYPE_DATA

Items　戻されるインデックス・ファイルおよびデータ・ファイルの数

備考

EsbListDbFiles()を呼び出した後に、ESB_DBFILEINFO_TYPE を使用して**EsbGetNextItem()**を呼び出すと、必要なデータベース・ファイルの情報が入っている構造体を取得できます。

戻り値

- 成功の場合、
- **EsbListDbFiles()**によって 0 が戻されます
- Items には、戻されたインデックス・ファイルまたはデータ・ファイルの数が含まれます
- ESB_DBFILEINFO_T 構造体のリストが作成されます。各構造体にはインデックスの 1 つまたは戻されたデータ・ファイルについての情報が含まれます。

例

```
    Dim OutDbInfo As ESB_DBFILEINFO_T
 Dim FileType  As Integer
 Dim Count   As Long

 FileType = ESB_FILETYPE_INDEX + ESB_FILETYPE_DATA
```

```
sts = EsbListDbFiles(hCtx, "sample", "basic", FileType, Count)
MsgBox (sts)
If Not sts Then
 For Index = 1 To Count
  sts = EsbGetNextItem(hCtx, ESB_DBFILEINFO_TYPE, OutDbInfo)
 Next
End If
```

関連トピック

- 1271 ページの「ESB_DBFILEINFO_T」

EsbListDrillThruURLs

アクティブなデータベース・アウトライン内のドリルスルー URL をリストします。

1904 ページの「ドリルスルー URL の制限」.

構文

```
    Declare Function EsbListDrillThruURLs Lib "esbapin" (ByVal hCtx As Long, ByRef
URLNames As Variant) As Long
```

パラメータ	説明
hCtx	Visual Basic API のコンテキスト・ハンドル
URLNames	URL 名のリスト

戻り値

- 正常に処理されると、アクティブなデータベース・アウトライン内のドリルスルー URL 名がリストされます。
- 処理に失敗すると、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース読取り権限(ESB_PRIV_READ)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
   Sub ESB_ListGLDrillThru()
Dim intX    As Integer
Dim URLNames  As Variant

sts = EsbListDrillThruURLs(hCtx, URLNames)

If sts = 0 Then
 Debug.Print "EsbListDrillThruURLs sts: " & sts
```

```
   For intX = LBound(URLNames) To UBound(URLNames)

   Debug.Print "URL Name: " & URLNames(intX)

   Next
  End If
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbListFilters

データベースのすべてのフィルタをリストします。

構文

```
      EsbListFilters
       (
      hCtx, AppName, DbName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
pItems	フィルタ名のアイテムを受け取る変数のアドレス。

戻り値

正常終了の場合は、データベース内のフィルタのアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なフィルタ名の配列が生成されます。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・マネージャ権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbListFilters Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, Items As Integer) As Long

Sub ESB_ListFilters ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String * ESB_FTRNAMELEN
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"  '*************
 ' List Filters
 '*************
 sts = EsbListFilters (hCtx, AppName, DbName, Items)  For n = 1 To Items
  '****************************
  ' Get next Filter Name String
  ' from the list
  '****************************
  sts = EsbGetNextItem (hCtx,
     ESB_FTRNAME_TYPE, ByVal FilterName)
 Next
End Sub
```

関連トピック

- EsbGetFilter
- EsbSetFilter
- EsbGetNextItem

EsbListGroups

ある特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセス権を持つすべてのグループをリストします。

構文

```
      EsbListGroups
       (
      hCtx, AppName, DbName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。空の文字列の場合はすべてのグループがリストされます。
DbName	データベース名。空の文字列の場合は、アプリケーション内のすべてのデータベースのグループがリストされます。
pItems	グループのアイテムを受け取る変数のアドレス。

備考

AppName と DbName の両方が空の文字列でない場合、指定したアプリケーションとデータベースへのアクセス権を持つグループのみがリストされます。DbName が空の文字列の場合、指定したアプリケーションへのアクセス権を持つグループのみがリストされます。AppName が空の文字列の場合、ログオンしているサーバー上のすべてのグループがリストされます。

戻り値

正常終了の場合、グループの数のアイテムが pItems に戻され、**EsbGetNextItem()** を介してアクセス可能な指定されたアプリケーションおよびデータベースに対してアクセス権を持っているグループのリストが生成されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbListGroups Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, Items As Integer) As Long

Sub ESB_ListGroups ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim GroupInfo As ESB_USERINFO_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"  '************
 ' List Groups
 '************
 sts = EsbListGroups (hCtx, AppName, DbName,
    Items)  For n = 1 To Items   '************************
  ' Get next Group structure
  ' from the list
  '************************
  sts = EsbGetNextItem (hCtx,
     ESB_GROUPINFO_TYPE, GroupInfo)
 Next
End Sub
```

関連トピック

- EsbGetGroup
- EsbListUsers

- EsbGetNextItem

EsbListLocks

特定のアプリケーションおよびデータベースに接続されているユーザーをすべてリストします。その際、現在ロックされているデータ・ブロックのアイテムも一緒にリストします。

構文

```
      EsbListLocks
       (
      hCtx, AppName, DbName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
pItems	ユーザーのアイテムを受け取る変数のアドレス。

備考

この関数は、この関数が呼び出されたときにサーバーに接続していたユーザーのみがリストされる場合に「スナップショット」になります。

戻り値

正常終了の場合は、接続されているユーザー数のアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能なユーザー・ロック構造体のリストが生成されます。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbListLocks Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, Items As Integer) As Long

Sub ESB_ListLocks ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim LockInfo As ESB_LOCKINFO_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"  '***********
 ' List Locks
 '***********
 sts = EsbListLocks (hCtx, AppName, DbName,
    Items)  For n = 1 To Items   '****************************
  ' Get next user lock structure
  ' from the list
  '****************************
  sts = EsbGetNextItem (hCtx,
     ESB_LOCKINFO_TYPE, LockInfo)
 Next
End Sub
```

関連トピック

- EsbListConnections
- EsbListUsers
- EsbRemoveLocks
- EsbGetNextItem

EsbListLogins

現在のセッションのログイン・インスタンスのリストを戻します。

構文

```
     Declare Function EsbListLogins Lib "esbapin" (
ByVal hCtx As Long,
   pItems As Integer) As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

pItems　　サーバーから戻されるログイン・リストのログイン回数。

備考

同じユーザー名とサーバーに対して **EsbListLogins()**を複数回呼び出せます。API は
指定されたサーバーへの各ログインに一意のコンテキスト・ハンドルを戻します。

戻り値

正常終了の場合は、ログイン情報と現在のログイン・カウントが戻されます。

アクセス

この関数を呼び出す前に **EsbInit()**を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

EsbListObjects

サーバーまたはローカル・クライアント上にある、指定したタイプのオブジェクトをリストします。

構文

```
      EsbListObjects
       (
      hCtx, ObjType, AppName, DbName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。**EsbCreateLocalContext()**から戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(複数のタイプ可)。使用可能な値のリストはビットマスク・データ型を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーション・サブディレクトリのオブジェクトがリストされます。
pItems	該当するタイプのオブジェクトのアイテムを受け取る変数のアドレス。

戻り値

正常終了の場合は、該当するタイプのオブジェクト数のアイテムが pItems に戻され、**EsbGetNextItem()**を介してアクセス可能な一致するオブジェクト構造体の配列が生成されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。ただし、呼出し元がアプリケーションまたはデータベース(あるいはその両方)に対して(オブジェクト・タイプに応じて)適切なレベルのアクセス権限を持っている場合、サーバー・オブジェクトのみがリストされるので注意してください。

例

```
    Declare Function EsbListObjects Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, Items As Integer) As
Long

Sub ESB_ListObjects ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim ObjType As Integer
 Dim ObjInfo As ESB_OBJINFO_T
 Dim sts As Long  Appname = "Sample"
 DbName = "Basic"
 ObjType = ESB_OBJTYPE_OUTLINE  '**********************
 ' List Outline Objects
 '**********************
 sts = EsbListObjects (hCtx, ObjType,
    AppName, DbName, Items)  For n = 1 To Items   '**************************
  ' Get next Object Structure
  ' from the list
  '**************************
  sts = EsbGetNextItem (hCtx,
     ESB_OBJINFO_TYPE, ObjInfo)
 Next
End Sub
```

関連トピック

- EsbGetObject
- EsbGetObjectInfo
- EsbGetNextItem

EsbListRequests

アクティブなセッションおよび要求に関する情報を戻します。

構文

```
      EsbListRequests (hCtx, UserName, AppName, DbName, Items)
 ByVal hCtx   As Long
```

```
 ByVal UserName As String
 ByVal AppName  As String
 ByVal DbName   As String
    Items   As Long
```

パラメータ 説明

hCtx	コンテキスト・ハンドル
AppName	アプリケーション名
DbName	データベース名
UserName	ユーザー名
Items	戻されるインデックス・ファイルおよびデータ・ファイルの数

備考

- セッションとは、ユーザーがログインしてからログアウトするまでの時間を秒数で表したものです。
- 要求とは、ユーザーまたは他のプロセスが Essbase に対して送信するクエリーです。たとえば、アプリケーションの起動やデータベース・アウトラインの再構築に対する要求などがあります。各セッションは同時に複数の要求を処理できないため、セッションと要求は 1 対 1 の関係にあります。
- リストされた要求の中には終了済にもかかわらず、ネットワークの遅延によってアクティブとしてリストされたままのものもあります。
- この関数によって、UserName、AppName および DbName によって指定されたプロセスによって起動された要求およびセッションに関する情報が戻されます。これらのパラメータが NULL または空の場合、システム内のすべてのプロセスがリストされます。この関数は現在の要求数と、各要求に 1 つの ESB_REQUESTINFO_T 構造体を戻します。
- **EsbListRequests()**を呼び出した後、ESB_REQUESTINFO_TYPE を使用して **EsbGetNextItem()**を呼び出し、必要な要求情報構造体を取得します。

戻り値

正常終了の場合は、ユーザー数が Items に戻され、指定したアプリケーションおよびデータベースに対してアクセス権を持っているユーザーのリストが生成されます。このリストにアクセスするには、**EsbGetNextItem()**を使用します。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbListRequests Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
UserName As String, ByVal AppName As String, ByVal DbName As String, pItems As
Integer) As Long

Sub ESB_ListRequests()
  Dim Items As Integer
  Dim ReqInfo As ESB_REQUESTINFO_T
```

```
  Dim sts As Long
  Dim pAccess As Integer

  'sts = EsbSetActive(hCtx, AppName, DbName, pAccess)
  'sts = EsbDefaultCalc(hCtx)
  '*************
  ' List Requests
  '*************
  sts = EsbListRequests(hCtx, UserName, AppName, DbName, Items)
  Debug.Print "EsbListRequests = " & sts & " " & Items

  For n = 1 To Items
   '********************
   ' Get next Request Info
   ' from the list
   '********************
   sts = EsbGetNextItem(hCtx, ESB_REQUESTINFO_TYPE, ReqInfo)
   Debug.Print "EsbGetNextItem = " & sts & " " & ReqInfo.LoginId & " " &
ReqInfo.DbRequestCode

  Next
 End Sub
```

関連トピック

- EsbKillRequest

EsbListUsers

特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。

構文

```
      EsbListUsers
       (
      hCtx, AppName, DbName, pItems
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String

      pItems
       As Integer
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル
AppName	アプリケーション名。空の文字列の場合は、すべてのユーザーがリストされます
DbName	データベース名。
pItems	ユーザーのカウントを受け取る変数のアドレス

備考

- AppName および DbName の両方が空の文字列でない場合は、指定したアプリケーションとデータベースへのアクセス権のあるユーザーのみがリストされます。DbName が空の文字列の場合、指定したアプリケーションへのアクセス権のあるユーザーのみがリストされます。AppName が空の文字列の場合は、そのサーバー上のすべてのユーザーがリストされます。
- **EsbGetNextItem()**を使用して、指定したアプリケーションおよびデータベースにアクセス可能なユーザーのリストが、**ESB_USERINFO_T** 構造体のリストとして戻されます。戻されるユーザー情報構造体の「AppName」および「DbName」フィールドには NULL 値が含まれています。

戻り値

正常終了の場合、ユーザー数が pCount に戻され、**EsbGetNextItem()**を介してアクセス可能な、指定されたアプリケーションおよびデータベースに対してアクセス権を持っているユーザーのリストが生成されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbListUsers Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, Count As Integer) As Long

Sub ESB_ListUsers ()
 Dim Count As Integer
 Dim AppName As String
 Dim DbName As String
 Dim UserInfo As ESB_USERINFO_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"  '***********
 ' List Users
 '***********
 sts = EsbListUsers (hCtx, AppName, DbName,
    Count)  For n = 1 To Count   '*****************************
  ' Get next User Info structure
  ' from the list
  '*****************************
  sts = EsbGetNextItem (hCtx,
     ESB_USERINFO_TYPE, UserInfo)
 Next
End Sub
```

関連トピック

- EsbGetUser
- EsbListConnections
- EsbListGroups
- EsbListLocks
- EsbGetNextItem

EsbListUsersEx

特定の Essbase サーバー、アプリケーションまたはデータベースへのアクセスを持つすべてのユーザーをリストします。この関数は EsbListUsers に類似していますが、セキュリティ・プロトコル・パラメータが追加されています。

構文

```
     Declare Function EsbListUsersEx Lib "esbapin" (
ByVal hCtx As Long,
ByVal AppName As String,
ByVal DbName As String,
ByVal Protocol As String,
   pItems As Integer) As Long
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
AppName	アプリケーション名。NULL の場合は、すべてのユーザーがリストされます。
DbName	データベース名。NULL の場合は、アプリケーション内のすべてのデータベースのユーザーがリストされます。
Protocol	外部認証セキュリティ・プロトコルのメカニズム名。
pItems	ユーザーの数。

備考

- AppName および DbName の両方が NULL でない場合、指定したアプリケーションとデータベースへのアクセス権を持つユーザーのみがリストされます。DbName が NULL の場合、指定したアプリケーションへのアクセス権を持つユーザーのみがリストされます。AppName が NULL の場合、サーバー上のすべてのユーザーがリストされます。
- 指定したアプリケーションおよびデータベースへのアクセス権があるユーザーのリストが、1298 ページの「ESB_USERINFOEX_T」構造体のリストとして戻されます。戻されるユーザー情報構造体の「AppName」および「DbName」フィールドには、NULL 値が含まれています。EsbGetNextItem を呼び出して構造体のリストにアクセスできます。

戻り値

正常終了の場合は、ユーザー数のカウントが pCount に、指定したアプリケーションおよびデータベースへのアクセス権を持つユーザーのリストが ppUserList に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

EsbListVariables

入力条件に適合するすべての代替変数をリストします。

構文

```
      EsbListVariables
       (
      hCtx, pVariable, pItems
      )
ByVal
      hCtx
         As Long

      pVariable
       As ESB_PVARIABLE_T

      pItems
        As Integer
```

パラメータ 説明

hCtx　　API へのコンテキスト・ハンドル。

pVariable　リストされた代替変数の説明を含む構造体を指すポインタ。

- メンバー VarName および VarValue は無視されます。
- Server メンバーは指定されていますが、AppName および DbName が空の場合は、関数によってサーバー・レベルの代替変数のみがリストされます。
- Server および AppName メンバーは指定されていますが、DbName が空の場合は、指定したサーバーとアプリケーション・レベルの両方のすべての変数がリストされます。
- Server、AppName および DbName という 3 つのメンバーすべてが指定されている場合は、指定した 3 つのすべてのレベルの変数がリストされます。
- フィールドが空の場合は、そのフィールドは「無視」するよう処理されます。

pItems　　ppVarList パラメータに戻される変数の数を示す、符号なし long 値を指すポインタ。

戻り値

成功の場合はゼロ(0)が戻されます。

例

```
   Declare Function EsbListVariables Lib "esbapin" (ByVal hCtx As Long, pVariable As
ESB_VARIABLE_T, pItems As Integer) As Long
```

```
Public Sub ESB_ListVariables ()
 Dim i As Integer
 Dim nCount As Integer
 Dim sts As Long
 Dim oVariable As ESB_VARIABLE_T
 oVariable.AppName = "Sample"
 sts = EsbListVariables(hCtx, oVariable, nCount)
 If sts = 0 Then
  If nCount <> 0 Then
   For i = 1 To nCount
    sts = EsbGetNextItem(hCtx, ESB_VARIABLE_TYPE, oVariable)
    Debug.Print "Variable Name: " & oVariable.VarName
    Debug.Print "Value: " & oVariable.VarValue
    Debug.Print
   Next
  Else
   MsgBox "No substitution variables found."
  End If
 Else
  MsgBox "Error listing substitution variables."
 End If
End Sub
```

関連トピック

- 1300 ページの「ESB_VARIABLE_T」
- EsbCreateVariable
- EsbDeleteVariable
- EsbGetVariable

EsbLoadAlias

構造化テキスト・ファイルからアクティブなデータベースの別名テーブルを作成し、永続的にロードします。

構文

```
      EsbLoadAlias
       (
      hCtx, AltName, FileName
      )
ByVal
      hCtx
        As Long
ByVal
      AltName
       As String
ByVal
      FileName
       As String
```

パラメータ 説明

hCtx VB API コンテキスト・ハンドル。

AltName ロードする別名テーブル名。

FileName サーバー上の構造化された別名ファイルのフル・パス名。

備考

- この関数は、AliasName がすでに存在する場合は正常終了しません。既存のテーブルと同じ名前の別名テーブルをロードするには、既存の別名テーブルを先に削除する必要があります。
- 別名テーブル・ファイルのフォーマットは、『Oracle Essbase データベース管理者ガイド』に記載されています。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbLoadAlias Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AliasName
As String, ByVal FileName As String) As Long

Sub ESB_LoadAlias ()
 Dim sts As Long
 Dim AliasName As String
 Dim FileName As String  AliasName = "TestAlias"
 FileName = "c:\essbase\test.alt"  '***********
 ' Load Alias
 '***********
 sts = EsbLoadAlias (hCtx, AliasName, FileName)
End Sub
```

関連トピック

- EsbListAliases
- EsbRemoveAlias
- EsbSetActive

EsbLoadApplication

サーバー上のアプリケーションを開始します。

構文

EsbLoadApplication

```
      (hCtx, AppName)

ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　ロードされるアプリケーションの名前。

備考

アプリケーションをロードするには、接続されているユーザーがアプリケーションに対してロード権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したアプリケーションに対して、呼出し元がアプリケーションのロード/アンロード権限(ESB_PRIV_APPLOAD)を所有している必要があります。

例

```
    Declare Function EsbLoadApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String) As Long

Sub ESB_LoadApplication ()
 Dim sts As Long
 Dim AppName As String   AppName = "Sample"   '*****************
 ' Load Application
 '*****************
 sts = EsbLoadApplication (hCtx, AppName)
End Sub
```

関連トピック

- EsbLoadDatabase
- EsbUnloadApplication

EsbLoadDatabase

アプリケーション内のデータベースをサーバー上で開始します。

構文

```
      EsbLoadDatabase
```

```
      (
      hCtx, AppName, DbName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	ロードされるデータベースの名前。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースのロード/アンロード権限(ESB_PRIV_APPLOAD)を持っている必要があります。

例

```
    Declare Function EsbLoadDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String) As Long

Sub ESB_LoadDatabase ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String   AppName = "Sample"
 DbName = "Basic"   '***************
 ' Load Database
 '***************
 sts = EsbLoadDatabase (hCtx, AppName, DbName)
End Sub
```

関連トピック

- EsbLoadApplication
- EsbUnloadDatabase

EsbLockObject

サーバー上のオブジェクトまたはクライアントのオブジェクト・システムをロックし、他のユーザーによって更新されるのを防止します。

構文

```
      EsbLockObject
       (
      hCtx, ObjType, AppName, DbName, ObjName
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      ObjName
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されるローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	ロックされるオブジェクトの名前。

備考

- オブジェクトをロックするには、そのオブジェクトが存在している必要があり、かつ他のユーザーによってロックされていてはいけません。
- この関数はオブジェクトを取得しません。オブジェクトを取得するには、EsbGetObject()を使用します。

戻り値

なし。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベースに対して、呼出し元がアプリケーション・デザイン権限またはデータベース・デザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbLockObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String)
As Long

Sub ESB_LockObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
 ObjType = ESB_OBJTYPE_OUTLINE  '******************
 ' Lock Rules Object
 '******************
 sts = EsbLockObject (hCtx, ObjType, AppName,
    DbName, ObjName)
End Sub
```

関連トピック

- EsbGetObject
- EsbGetObjectInfo
- EsbListObjects
- EsbPutObject
- EsbUnlockObject

EsbLogin

ユーザーを Essbase サーバーにログインさせます。この関数は通常、 EsbInit が正しく実行された後で、かつコンテキスト・ハンドルの引数を必要とするその他すべての VB API が呼び出される前に呼び出す必要があります。

構文

```
      EsbLogin (
      hInst, Server, User, Password, pItems, hCtx
      )
ByVal
      hInst
        As Long
ByVal
      Server
       As String
ByVal
      User
         As String
ByVal
      Password
       As String
```

```
pItems
  As Integer

hCtx
   As Long
```

パラメータ	説明
hInst	VB API インスタンス・ハンドル。

Server	ネットワーク・サーバー名の文字列。必須フィールド。
	サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:
	`http://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1`
	保護モード(SSL)の場合、URL の構文は次のとおりです
	`http[s]://host:port/aps/Essbase?ClusterName=logicalName&SecureMODE=yesORno`
	たとえば、
	`https://myhost:13080/aps/Essbase?clustername=Essbase-Cluster1&SecureMODE=Yes`

User	ユーザー名の文字列。必須フィールド。
Password	パスワード文字列。必須フィールド。
pItems	アクセス可能なアプリケーションおよびデータベースのアイテムを受け取る変数のアドレス。
hCtx	Essbase サーバー・コンテキスト・ハンドルへのポインタ。

備考

- Microsoft Windows でプログラミングをする場合は、EsbLogin のかわりに EsbAutoLogin 関数を使用することを検討する必要があります。
- 同じユーザー名とサーバーに対して EsbLogin を複数回呼び出すことができます。指定したサーバーに対してログインするたびに、API から一意のコンテキスト・ハンドルが戻されます。

戻り値

成功の場合、Essbase サーバー・コンテキスト・ハンドルが phCtx に戻され、他の API 関数への後続の呼出しで引数として使用できます。また、指定したユーザーがアクセス可能なデータベースのアイテムが pItems に戻され、EsbGetNextItem を呼び出して読み取ることができる、アクセス可能なアプリケーションおよびデータベースのリストが生成されます。

アクセス

この関数を呼び出す前に、EsbInit 関数を呼び出して、最初に API を初期化し有効なインスタンス・ハンドルを取得する必要があります。

例

```
    Declare Function EsbLogin Lib "ESBAPIN" (ByVal hInst As Long, ByVal Server As
String, ByVal User As String, ByVal Password As String, Items As Integer, hCtx As
Long) As Long

Sub ESB_Login ()
 Dim hInst As Long
 Dim Server As String * ESB_SVRNAMELEN
 Dim User As String * ESB_USERNAMELEN
 Dim Password As String * ESB_PASSWORDLEN
 Dim Items As Integer
 Dim AppDb As ESB_APPDB_T
 Dim hCtx As Long  '*****************
 ' Login to Essbase Server
 '*****************
 sts = EsbLogin (hInst, Server, User, Password, Items, hCtx)  For n = 1 To Items
 '*******************************
 ' Get next Application/Database
 ' name combination from the list
 '*******************************
 sts = EsbGetNextItem (hCtx, ESB_LAPPDB_TYPE,AppDb)
 Next
End Sub
```

関連トピック

- EsbAutoLogin
- EsbInit
- EsbListDatabases
- EsbLogout
- EsbSetActive
- EsbGetNextItem

EsbLoginSetPassword

ユーザーをログインさせ、パスワードを変更します。パスワードが失効した場合、または次のログイン時に変更が必要な場合にこの関数を使用します。

構文

```
EsbLoginSetPassword(
hInstance
,
Server
,
UserName
,
Password
```

```
      ' _

      NewPassword
      ,
      Items
      ,
      hCtx
      )
ByVal
      hInstance
       As Long
ByVal
      Server
         As Long
ByVal
      UserName
        As String
ByVal
      Password
        As String
ByVal
      NewPassword
       As String

      Items
         As Integer

      hCtx
          As Long
```

パラメータ　説明

hInstance　API インスタンス・ハンドル。

Server　ネットワーク・サーバー名の文字列。

サーバー名は、hostname、hostname:port、または APS サーブレットのエンドポイントに Essbase フェイルオーバー・クラスタ名を付加した URL として表すことができます。次に例を示します:

```
        http://myhost:13080/aps/Essbase?clustername=Essbase-
Cluster1
```

保護モード(SSL)の場合、URL の構文は次のとおりです

```
        http[s]://host:port/aps/Essbase?
ClusterName=logicalName&SecureMODE=yesORno
```

たとえば、

```
        https://myhost:13080/aps/Essbase?clustername=Essbase-
Cluster1&SecureMODE=Yes
```

UserName　ユーザー名。

パラメータ	説明
Password	旧パスワード。
NewPassword	新パスワード。
Items	アクセス可能なデータベースの数。
hCtx	Essbase サーバー・コンテキスト・ハンドル。

備考

- EsbLoginSetPassword は、EsbLogin を呼び出し、ステータス・コード 1051090 (パスワードが期限切れ)または 1051093 (すぐにパスワードを変更)を受け取った後に呼び出します。
- Microsoft Windows では、EsbLoginSetPassword のかわりに EsbAutoLogin を使用することを検討してください。
- EsbFree を使用して、Items に割り当てられているメモリーを解放してください。

戻り値

成功の場合、EsbLoginSetPassword:

- hCtx に Essbase サーバー・コンテキスト・ハンドルを戻します。
- ユーザーがアクセス可能なデータベースの数が Items に戻されます。
- EsbGetNextItem の呼出しによって読み取ることのできる、アクセス可能なデータベースのリストが生成されます。

アクセス

EsbLoginSetPassword を呼び出す前に、EsbInit を呼び出して API を初期化し、有効なインスタンス・ハンドルを取得します。

例

```
    Declare Function EsbLoginSetPassword Lib "ESBAPIN" (ByVal hInst As Long, ByVal
Server As String, ByVal User As String, _
 ByVal Password As String, ByVal NewPassword As String, Items As Integer, hCtx As
Long) As Long

 Sub ESB_LoginSetPassword ()
 Dim hInst    As Long
 Dim Server    As String * ESB_SVRNAMELEN
 Dim User     As String * ESB_USERNAMELEN
 Dim Password   As String * ESB_PASSWORDLEN
 Dim NewPassword As String * ESB_PASSWORDLEN
 Dim Items    As Integer
 Dim AppDb    As ESB_APPDB_T
 Dim hCtx     As Long  sts = EsbLoginSetPassword (hInst, Server, User, Password,
NewPassword, Items, hCtx)  For n = 1 To Items  '*******************************
 ' Get next Application/Database
 ' name combination from the list
 '*******************************
 sts = EsbGetNextItem (hCtx, ESB_LAPPDB_TYPE, AppDb)
```

```
 Next
End Sub
```

関連トピック

- EsbAutoLogin
- EsbInit
- EsbListDatabases
- EsbLogout
- EsbSetActive

EsbLogout

ユーザーを Essbase サーバーからログアウトさせます。

構文

```
      EsbLogout
       (
      hCtx
      )
ByVal
      hCtx
       As Long
```

パラメータ 説明

hCtx	ログアウトする VB API コンテキスト・ハンドル。

備考

- この関数は指定されたコンテキスト・ハンドルが示すログインのみをログアウトさせます。その他のログインまたはコンテキストは、同じユーザー名を使用していても影響を受けません。
- この関数はログイン・コンテキストについてのみ使用してください。ローカル・コンテキストには、**EsbDeleteLocalContext**()関数を使用します。

戻り値

なし。

アクセス

この関数を呼び出すには、呼出し元が事前に **EsbLogin**()または **EsbAutoLogin**()関数を使用して正常にログインしている必要があります。

例

```
    Declare Function EsbLogout Lib "ESBAPIN" (ByVal hCtx As Long) As Long

Sub ESB_Logout ()
 Dim sts As Long  '*******
 ' Logout
```

```
 '*******
 sts = EsbLogout (hCtx)
End Sub
```

関連トピック

- EsbAutoLogin
- EsbDeleteLocalContext
- EsbGetActive
- EsbLogin
- EsbLogoutUser

EsbLogoutUser

管理者またはアプリケーション・マネージャが他のユーザーを Essbase サーバーから切断できるようにします。

構文

```
      EsbLogoutUser
       (
      hCtx, LoginId
      )
ByVal
      hCtx
        As Long
ByVal
      LoginId
       As Long
```

パラメータ 説明

hCtx	ログアウトを強制するユーザーの VB API コンテキスト・ハンドル。
LoginId	ログアウトされるユーザーのログイン ID。

備考

- LoginId は、**EsbListConnections()**関数によって戻されるユーザー情報構造体から取得できます。
- この関数は指定された LoginID が示すログインのみをログアウトさせます。その他のログインまたはコンテキストは影響を受けません。
- 管理者は、hCtx にログインしたサーバーにログインしたユーザーをログアウトできます。アプリケーション・マネージャは、hCtx をアプリケーション・マネージャとしてアプリケーションに接続しているユーザーのみをログアウトできます。
- 自分自身はログアウトできません。

戻り値

なし。

アクセス

この関数を呼び出すには、管理者またはアプリケーション・マネージャ権限を持っている必要があります。

例

```
    Declare Function EsbLogoutUser Lib "ESBAPIN" (ByVal hCtx As Long, ByVal LoginId
As Long) As Long

Sub ESB_LogoutUser()
 Dim Items As Integer
 Dim UserInfo As ESB_USERINFO_T
 Dim sts As Long
 '*****************
 ' List Connections
 '*****************
 sts = EsbListConnections(hCtx, Items)
 '*****************
 ' Log out all users
 '*****************
 For n = 1 To Items
  '****************************
  ' Get next User Info structure
  ' from the list
  '****************************
  sts = EsbGetNextItem(hCtx, ESB_USERINFO_TYPE, UserInfo)
  sts = EsbLogoutUser(hCtx, UserInfo.LoginId)
 Next
End Sub
```

関連トピック

- EsbListConnections
- EsbLogout

EsbLogSize

Essbase サーバー・ログ・ファイル(essbase.log)のサイズ、またはアプリケーション・ログ・ファイル(appname.log)のサイズを戻します。

構文

```
     Declare Function EsbLogSize Lib "esbapin" (
ByVal hCtx As Long,
ByVal isAgentLog As Integer,
ByVal AppName As String,
   pulLogSize As Long) As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

パラメータ 説明

isAgentLog TRUE の場合、Essbase サーバー・ログ・ファイル(essbase.log)のサイズが戻されます。FALSE の場合、アプリケーション・ログ・ファイル(appname.log)のサイズが戻されます。

AppName アプリケーション名。

pulLogSize 戻されるログ・ファイルのサイズ。

備考

- メッセージ・ログを表示するには、**EsbGetLogFile()**を使用します。
- essbase.log および appname.log の場所は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するのに、呼出し元がアクセス権限を持っている必要はありません。

EsbLROAddObject

レポート・オブジェクトを Essbase データベースのデータ・セルにリンクします。

構文

```
      EsbLROAddObject
       (
      hCtx, memCount, MemComb, usOption, pLRODesc
      )
ByVal
      hCtx
        As Long
ByVal
      memCount
       As Long
ByVal
      MemComb
       As String
ByVal
      usOption
       As Integer

      pLRODesc
       As ESB_LRODESC_API_T
```

パラメータ 説明

hCtx API コンテキスト・ハンドル。

パラメータ 説明

MemCount　pMemComb で指定したメンバー数。

MemComb　リンクするデータ・セルを定義するメンバー名(EOL、CR で区切られる)の文字列。

usOption　オブジェクトの保管先を指定するオプション。次のいずれかを使用します:

- ESB_STORE_OBJECT_API は、オブジェクトをサーバーに保管します
- ESB_NOSTORE_OBJECT_API は、サーバー上に保管しません

pLRODesc　オブジェクトの説明構造体 1256 ページの「ESB_LRODESC_API_T」へのポインタ。

備考

- リンク・オブジェクトは、次のいずれかのタイプです:
  - Word 文書、Excel スプレッドシートまたはビットマップ・イメージなどのフラット・ファイル。
  - 最大 599 文字のテキストが含まれるセル・ノート。
  - URL へのリンク。
  - その他のデータベースへのリンク(リンク・パーティション機能)。
- オブジェクトをサーバーに保管しない(usOption)を選択した場合、サーバーにはリンク情報のみが保管され、アプリケーションでオブジェクトに関するすべてのファイル管理タスクを行うことになります。
- セル・ノートは、常にサーバーに保管されるため、usOption パラメータは無視されます。
- URL リンク・オブジェクトに対する usOption パラメータは、常に ESB_NOSTORE_OBJECT_API である必要があります。
- EsbLROAddObject では、現在ログインしているユーザー名がオブジェクトに対する作成者ユーザー名として使用され、pLRODesc オブジェクト説明構造体で指定したユーザー名は無視されます。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbLROAddObject Lib "esbapin" _
(ByVal hCtx As Long, ByVal memCount As Long, _
ByVal memComb As String, ByVal usOption As Integer, _
pLRODesc As ESB_LRODESC_API_T) As Long

Public Sub ESB_LROAddObject()  Dim Desc As ESB_LRODESC_API_T
 Dim memCount As Long
 Dim memComb As String
```

```
 Dim opt As Integer
 Dim i As Integer

 Desc.userName = "Admin"

 Desc.ObjType = ESB_LROTYPE_CELLNOTE_API
 Desc.note = "Note from DFS"  memCount = 5
 memComb = "Jan" & vbLf & "Sales" & vbLf & _
      "Cola" & vbLf & "Utah" & vbLf & _
      "Actual"  opt = ESB_NOSTORE_OBJECT_API

 sts = EsbLROAddObject(hCtx, memCount, memComb, _
           opt, Desc)
End Sub
```

関連トピック

- 1257 ページの「ESB_LROINFO_API_T」
- EsbLROUpdateObject
- EsbLRODeleteObject

EsbLRODeleteCellObjects

データベース内の特定のデータ・セルのデータにリンクされたすべてのオブジェクトを削除します。セルにリンクされた特定のオブジェクトを削除するには、EsbLRODeleteObject を使用します。

構文

```
      EsbLRODeleteCellObjects
       (
      hCtx, memCount, memComb, PulCount
      )
ByVal
      hCtx
        As Long
ByVal
      memCount
       As Long
ByVal
      memComb
       As String

      PulCount
       As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

memCount　　MemComb に指定されているメンバー数。

MemComb　　メンバー名の文字列(EOL、CR で区切られる)。

パラメータ 説明

PulCount　削除されたLRO カタログ・エントリ数。

備考

- この関数では、指定されたセルにリンクされているすべてのオブジェクトがそれらのカタログ・エントリと一緒に削除されます。
- オブジェクトがサーバーに保管されていない場合は、セル・リンクのみ破棄されます。
- **EsbLRODeleteCellObjects()**では、削除するオブジェクトのリストが生成されます。この関数の呼出し後、EsbGetNextItem を使用して、削除した各オブジェクトの情報を取得します。コード例でこの方法を示します。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbLRODeleteCellObjects Lib "esbapin" _
(ByVal hCtx As Long, ByVal memCount As Long, _
ByVal memComb As String, PulCount As Long) As Long

Public Sub ESB_LRODeleteCellObjects()  Dim Desc As ESB_LRODESC_API_T
 Dim Items As Long
 Dim memCount As Long
 Dim memComb As String
 Dim i As Integer

 memCount = 5
 memComb = "Jan" & vbCrLf & "Sales" & _
      "Cola" & vbCrLf & "Utah" & _
      "Actual"

 sts = EsbLRODeleteCellObjects(hCtx, memCount, _
               memComb, Items)

 If sts = 0 Then
  For i = 1 To Items
    '*******************************
    '* Get the next LRO description
    '* item from the list
    '*******************************     sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE,
Desc)   Next i
 End IfEnd Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbGetNextItem
- EsbLROAddObject
- EsbLRODeleteObject
- EsbLROPurgeObjects

EsbLRODeleteObject

データベースのデータ・セルにリンクされた特定のオブジェクトを削除します。セルにリンクされたすべてのオブジェクトを削除するには、EsbLRODeleteCellObjects を使用します。

構文

```
      EsbLRODeleteObject
       (
      hCtx, pLinkId
      )
ByVal
      hCtx
        As Long

      pLinkId
       As ESB_LROHANDLE_API_T
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
pLinkId	オブジェクト識別構造体に対するポインタ。構造体は、EsbLROAddObject から 1256 ページの「ESB_LRODESC_API_T」構造体経由で戻されます。

備考

- 指定されたオブジェクトは削除され、カタログ・リストからも除外されます。
- オブジェクトがサーバーに保管されていない場合は、セル・リンクのみ破棄されます。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbLRODeleteObject Lib "esbapin" _
(ByVal hCtx As Long, pLinkID As ESB_LROHANDLE_API_T) _
```

```
As Long

Public Sub ESB_LRODeleteObject()  Dim LinkID As ESB_LROHANDLE_API_T

 LinkID.hObject = 1
 LinkID.cellKey.cellOffset = 0
 LinkID.cellKey.blkOffset = 198
 LinkID.cellKey.segment = 0

 sts = EsbLRODeleteObject(hCtx, LinkID)End Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbLROAddObject
- EsbLRODeleteCellObjects
- EsbLROPurgeObjects

EsbLROGetCatalog

データベースの指定されたデータ・セルに対してリンクされたオブジェクトのカタログ・エントリのリストを取得します。

構文

```
      EsbLROGetCatalog
       (
      hCtx, memCount, memComb, PulCount
      )
ByVal
      hCtx
        As Long
ByVal
      memCount
       As Long
ByVal
      memComb
       As String

      PulCount
       As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

memCount　　memComb に指定されているメンバーの数。

memComb　　メンバー名の配列。

PulCount　　呼出し元に戻される LRO カタログ・エントリの数。

備考

カタログ情報を取得するには、この関数を呼び出してから EsbGetNextItem を呼び出します。コード例でこの方法を示します。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbLROGetCatalog Lib "esbapin" _
(ByVal hCtx As Long, ByVal memCount As Long, _
ByVal memComb As String, PulCount As Long) As Long

Public Sub ESB_LROGetCatalog()

 Dim Desc As ESB_LRODESC_API_T
 Dim Items As Long
 Dim memCount As Long
 Dim memComb As String
 Dim i As Integer

 memCount = 5
 memComb = "Jan" & vbCrLf & "Sales" & _
      "Cola" & vbCrLf & "Utah" & _
      "Actual"

 sts = EsbLROGetCatalog(hCtx, memCount, _
            memComb, Items)

 If sts = 0 Then
  For i = 1 To Items
    '*******************************
    '* Get the next LRO description
    '* item from the list
    '*******************************     sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE,
Desc)    Next i
 End If

End Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbGetNextItem
- EsbLROAddObject
- EsbLROUpdateObject
- EsbLROGetObject
- EsbLRODeleteObject

EsbLROGetMemberCombo

現在の LRO のメンバー組合せリストから、n 番目のメンバーを取得します。

構文

```
      EsbLROGetMemberCombo
       (
      hCtx, memberIndex, memberName
      )
ByVal
      hCtx
      As ESB_HCTX_T
ByRef
      memberIndex
      As ESB_ULONG_T
ByVal
      memberName
      As ESB_MBRNAME_T
```

パラメータ　説明

hCtx　　VB API コンテキスト・ハンドル。

memberIndex　戻されるメンバーのメンバー・リスト内の位置。

memberName　戻されるメンバーの名前。

備考

- EsbLROGetMemberCombo()は、1256 ページの「ESB_LRODESC_API_T」の memComb で戻されなかった、(リンク・オブジェクトに関連付けられているデータ・セルを識別する)メンバーの組合せを戻します。
- hCtx に関連付けられているメモリー内のリストが LRO のリストであることを確認するため、EsbLROListObjects()または EsbLROGetObject()の後に、EsbLROGetMemberCombo()を呼び出してください。
- EsbLROListObjects()の呼出し後、EsbGetNextItem()を呼び出す必要があります。すると、EsbGetNextItem()によってフェッチされた現在の LRO に対して、ESbLROGetMemberCombo()が実行されます。
- 引き続き ESB_LRODESC_API_T の memCount を使用して、データ・セルを識別するメンバー組合せ内のメンバー名の数を調べることができます。例を参照してください。

戻り値

正常終了の場合は、memberName にメンバー名が戻されます。正常終了しなかった場合は、現在のオブジェクトが LRO タイプでないことを示す「-1」、または範囲外であることを示す「1」が戻されます。範囲外とは、memberIndex の位置にメンバーがないか、現在のアイテムに LRO が存在していないことを意味します。**「注意」**を参照してください。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
   Declare Function EsbLROGetMemberCombo Lib "ESBAPIN" (ByVal hCtx As Long, _
 ByVal MemberIndex As Long, ByVal MemberName As String * ESB_MBRNAMELEN) As Long

 Sub ESB_LROGetMemberCombo()
 Dim userName  As String * ESB_USERNAMELEN
 Dim listDate  As Long
 Dim Count    As Integer
 Dim Desc    As ESB_LRODESC_API_T
 Dim i     As Integer
 Dim j     As Integer
 Dim CutOffDate As Date
 Dim MemberName As String * ESB_MBRNAMELEN

 Const ESB_REFERENCE_DATE = #1/1/70#
 userName         = "admin"
 CutOffDate        = #8/1/97#
 listDate         = DateDiff("s", CutOffDate, ESB_REFERENCE_DATE)

 sts           = EsbLROListObjects(hCtx, userName, listDate, Count)

 If sts = 0 Then
  For i = 1 To Count

   '************************************
   '* Get the next LRO item from the list
   '************************************
   sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE, Desc)

   If sts = 0 Then
    For j = 1 To Desc.memCount

      '************************************************************
      '* Get the jth member from the member list of the current LRO
      '************************************************************
      sts = EsbLROGetMemberCombo(hCtx, j, MemberName)
    Next j
  Next i
 End If
End Sub
```

関連トピック

- EsbGetNextItem
- EsbLROGetObject
- EsbLROListObjects

EsbLROGetObject

データベースのデータ・セルにリンクされているオブジェクトを取得します。

構文

```
      EsbLROGetObject
       (
      hCtx, pLinkId, targetFile, usOption, pLRODesc
      )
ByVal
      hCtx
         As Long

      pLinkId
        As ESB_LROHANDLE_API_T
ByVal
      targetFile
       As String
ByVal
      usOption
        As Integer

      pLRODesc
        As ESB_LRODESC_API_T
```

パラメータ 説明

hCtx　API コンテキスト・ハンドル。

pLinkId　オブジェクト識別構造体に対するポインタ。リンク ID が、EsbLROAddObject によって、1256 ページの「ESB_LRODESC_API_T」構造体を介して戻されます。

targetFile　オブジェクトが取得されるターゲット・ファイル名。

usOption　オブジェクトとカタログ・エントリの一方またはその両方を取得するかどうかを指定するオプション。次のいずれかを使用します:

- ESB_LRO_OBJ_API はオブジェクトのみを取得します。
- ESB_LRO_CATALOG_API はカタログ・エントリのみを取得します。
- ESB_LRO_BOTH_API はオブジェクトとカタログ・エントリの両方を取得します。

pLRODesc　オブジェクトの説明構造体、1256 ページの「ESB_LRODESC_API_T」。

備考

セル・ノートはオブジェクトのカタログ・エントリの一部です。オブジェクトがセル・ノートかどうかを判断するには、1256 ページの「ESB_LRODESC_API_T」の ObjType フィールドを確認します。セル・ノートを取得するには、ESB_LRO_CATALOG_API を usOption パラメータに使用します。ノートのコンテンツは、1256 ページの「ESB_LRODESC_API_T」に含まれています。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbLROGetObject Lib "esbapin" _
(ByVal hCtx As Long, pLinkID As ESB_LROHANDLE_API_T, _
ByVal targetFile As String, ByVal usOption As Integer, _
pLRODesc As ESB_LRODESC_API_T) As Long

Public Sub ESB_LROGetObject()  Dim Desc As ESB_LRODESC_API_T
 Dim LinkID As ESB_LROHANDLE_API_T
 Dim TargetFile As String
 Dim opt As Integer
 Dim InputMsg As String

 LinkID.hObject = 1
 LinkID.cellKey.cellOffset = 0
 LinkID.cellKey.blkOffset = 198
 LinkID.cellKey.segment = 0

 TargetFile = "c:\docs\myfile.doc"

 InputMsg="Danger, Will Robinson"
 opt = InputBox(InputMsg, , ESB_LRO_BOTH_API)

 sts = EsbLROGetObject(hCtx, LinkID, TargetFile, _
           opt, Desc)
End Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbLROAddObject
- EsbLROGetMemberCombo
- EsbLROUpdateObject
- EsbLRODeleteObject

EsbLROListObjects

指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトのリストを取得します。

構文

```
      EsbLROListObjects
       (
      hCtx, userName, listDate, PulCount
      )
ByVal
      hCtx
        As Long
ByVal
      userName
       As String
ByVal
```

```
      listDate
       As Long

      PulCount
       As Long
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
userName	ユーザー名。指定した場合は、指定したユーザーが最後に変更したすべてのオブジェクトのリストが戻されます。
listDate	変更日。指定した場合は、特定の日付以前に変更されたすべてのオブジェクトのリストが戻されます。時刻は 1970 年 1 月 1 日以降に経過した秒数を Long 値で示します。
PulCount	戻される LRO カタログ・エントリの数。

備考

- userName および listDate パラメータの両方を指定した場合、両方の基準に合致するオブジェクトがリストされます。
- オブジェクトのリストを取得するには、この関数の呼出し後、EsbGetNextItem を呼び出します。コード例でこの方法を示します。
- EsbLROListObjects()を使用して EssLROListObjects()の機能を複製するには、EsbGetNextItem の呼び出し後に EsbLROGetMemberCombo を呼び出す必要があります。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すためには、日付セルまたはアクティブなデータベースに対して読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbLROListObjects Lib "esbapin" _
(ByVal hCtx As Long, ByVal userName As String, _
ByVal listDate As Long, PulCount As Integer) As Long

Public Sub ESB_LROListObjects()

 Dim userName As String * ESB_USERNAMELEN
 Dim listDate As Long
 Dim Items As Long
 Dim Desc As ESB_LRODESC_API_T
 Dim i As Integer
 Dim CutOffDate As Date

 Const ESB_REFERENCE_DATE = #1/1/70#
 userName = "admin"
 CutOffDate = #8/1/97#
```

```
 listDate = DateDiff("s", CutOffDate, _
         ESB_REFERENCE_DATE)

 sts = EsbLROListObjects(hCtx, userName, _
           listDate, Items)

 If sts = 0 Then
  For i = 1 To Items
    '*******************************
    '* Get the next LRO description
    '* item from the list
    '*******************************
   sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE, Desc)
   Next i
 End If
End Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbGetNextItem
- EsbLROGetCatalog
- EsbLROGetMemberCombo
- EsbLROPurgeObjects

EsbLROPurgeObjects

指定したユーザー名または変更日(あるいはその両方)の、アクティブ・データベースのセルにリンクされているすべてのオブジェクトを削除します。

構文

```
      EsbLROPurgeObjects
       (
      hCtx, userName, purgeDate, PulCount
      )
ByVal
      hCtx
         As Long
ByVal
      userName
       As String
ByVal
      purgeDate
       As Long

      PulCount
       As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

パラメータ 説明

userName　ユーザー名を指すポインタ。指定した場合、特定のユーザーが最後に変更したすべてのオブジェクトが削除されます。

purgeDate　変更日。指定した場合は、特定日以前に変更されたすべてのオブジェクトが削除されます。日付は 1970 年 1 月 1 日以降に経過した秒数を Long 値で表します。

PulCount　削除された LRO カタログ・エントリの数。

備考

- userName と purgeDate パラメータの両方を指定した場合、両方の基準に合致するオブジェクトが削除されます。
- **EsbLROPurgeObjects()**は削除するオブジェクトのリストを生成します。この関数の呼出し後、EsbGetNextItem を使用して、削除した各オブジェクトの情報を取得します。コード例でこの方法を示します。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対してデザイン権限(ESB_PRIV_DBDESIGN)持っている必要があります。

例

```
    Declare Function EsbLROPurgeObjects Lib "esbapin" _
(ByVal hCtx As Long, ByVal userName As String, _
ByVal purgeDate As Long, PulCount As Long) As Long

Public Sub ESB_LROPurgeObjects()  Dim userName As String * ESB_USERNAMELEN
 Dim purgeDate As Long
 Dim Items As Long
 Dim Desc As ESB_LRODESC_API_T
 Dim CutOffDate As Date
 Dim i As Integer  Const ESB_REFERENCE_DATE = #1/1/70#
 userName = "admin"

 CutOffDate = #8/1/97#
 purgeDate = DateDiff("s", ESB_REFERENCE_DATE, _
          CutOffDate)

 sts = EsbLROPurgeObjects(hCtx, userName, _
            purgeDate, Items)

 If sts = 0 Then
  For i = 1 To Items
    '*****************************
    '* Get the next LRO description
    '* item from the list
    '*****************************     sts = EsbGetNextItem(hCtx, ESB_LRO_TYPE,
Desc)   Next i
 End If
```

```
End Sub
```

関連トピック

- 1255 ページの「リンク・オブジェクトに対する定数および構造体の定義」
- EsbGetNextItem
- EsbLROGetCatalog
- EsbLRODeleteObject
- EsbLRODeleteCellObjects

EsbLROUpdateObject

リンク・オブジェクトの更新済バージョンをサーバーに保管します。

構文

```
      EsbLROUpdateObject
       (
      hCtx, pLinkId, usOption, pLRODesc
      )
ByVal
      hCtx
        As Long

      pLinkId
       As ESB_LROHANDLE_API_T
ByVal
      usOption
       As Integer

      pLRODesc
       As ESB_LRODESC_API_T
```

パラメータ 説明

hCtx　API コンテキスト・ハンドル。

pLinkId　オブジェクト識別構造体。

usOption　オブジェクトのどの部分を更新するかを指定するオプション。次のいずれかを使用します:

- ESB_LRO_BOTH_API は、オブジェクトのファイルおよびカタログ・エントリの両方を更新します
- ESB_LRO_OBJ_API は、オブジェクトのファイルのみを更新します
- ESB_LRO_CATALOG_API は、オブジェクトのカタログ・エントリのみを更新します

pLRODesc　オブジェクトの説明構造体、1256 ページの「ESB_LRODESC_API_T」。

備考

- オブジェクトを作成し、オブジェクトのカタログ・エントリにそれを保管すると、Essbase はリンク ID を割り当てます。EsbLROGetCatalog を使用して、説明構造体 1256 ページの「ESB_LRODESC_API_T」に含まれるカタログ・エ

ントリを取得します。次に、説明構造体を変更し、**EsbLROUpdateObject()**を呼び出して、サーバー上に変更を保存できます。

- 次のように usOption パラメータを指定します:
  - オブジェクトがセル・ノートである場合、ESB_LRO_CATALOG_API を使用します。セル・ノートはカタログ・エントリに保管されます。
  - オブジェクトがファイルの場合は、ESB_LRO_BOTH_API を使用してファイルのコンテンツおよびカタログの両方を更新します。
  - カタログ情報のみ(オブジェクトの説明またはユーザー名など)を更新する場合、ESB_LRO_CATALOG.API を使用します。この場合、ファイルのコンテンツは更新されません。
  - ファイル・コンテンツのみを更新しカタログを更新しない場合、ESB_LRO_OBJ_API を使用します。この場合、ファイルのコンテンツおよび変更日のみが更新されます。
- リンク・オブジェクトは、次のいずれかのタイプです:
  - Word 文書、Excel スプレッドシートまたはビットマップ・イメージなどのフラット・ファイル。
  - 最大 599 文字のテキストが含まれるセル・ノート。
  - 別の Essbase データベースへのリンク(リンク・パーティション機能)。

戻り値

正常終了の場合は、ESB_STS_NOERR が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を呼び出すには、データ・セルまたはアクティブなデータベースに対して書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbLROUpdateObject Lib "esbapin" _
(ByVal hCtx As Long, pLinkID As ESB_LROHANDLE_API_T, _
ByVal usOption As Integer, _
pLRODesc As ESB_LRODESC_API_T) As Long

Public Sub ESB_LROUpdateObject()  Dim LinkID As ESB_LROHANDLE_API_T
 Dim Desc As ESB_LRODESC_API_T
 Dim opt As Integer

 LinkID.hObject = 1
 LinkID.cellKey.cellOffset = 0
 LinkID.cellKey.blkOffset = 198
 LinkID.cellKey.segment = 0

 Desc.userName = "admin"
 Desc.ObjType = ESB_LROTYPE_CELLNOTE_API
 Desc.note = "New Note from DFS"
 opt = ESB_STORE_OBJECT_API
```

```
  sts = EsbLROUpdateObject(hCtx, LinkID, _
              opt, Desc)
End Sub
```

関連トピック

- 1257 ページの「ESB_LROINFO_API_T」
- EsbLROGetObject
- EsbLROAddObject
- EsbLRODeleteObject

EsbPartitionApplyOtlChangeFile

サーバーに対して、アウトライン変更ファイルのリストを適用するように要求します。

構文

```
      EsbPartitionApplyOtlChangeFile
       (
      hCtx
      ,
      usFileNum
      ,
      fileList
      )
ByVal
      hCtx
         As Long
ByVal
      usfilenum
       As Integer
ByVal
      filelist
       As String
```

パラメータ 説明

hCtx	API コンテキストへのハンドル。
usFileNum	アウトライン変更ファイルの数。
fileList	CR/LF で区切られたファイル名の文字列。配列のサイズは usFileNum で定義されます。

備考

EsbPartitionGetOtlChanges を呼び出して filename を取得します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

アクセス

データベース・マネージャ権限が必要です。

例

```
    Public Sub ESB_PartitionApplyOtlChangeFile()
 Dim FileItems As Integer
 Dim Filelist As String
 Dim ProcState As ESB_PROCSTATE_T

 FileItems = 1
 Filelist = "C:\ESSBASE\APP\SAMPPART\COMPANY\ESS00001.CHG"
  sts = EsbPartitionApplyOtlChangeFile(hCtx, FileItems, Filelist)
    If sts = 0 Then
   sts = EsbGetProcessState(hCtx, ProcState)
   Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState(hCtx, ProcState)
  Loop
  End If
End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionApplyOtlChangeRecs

ターゲット・アウトラインへのアウトライン変更に適用されます。この関数は、EsbPartitionReadOtlChangeFile を呼び出した後に、EsbPartitionGetOtlChanges と対話的に使用するように設計されています。

構文

```
     EsbPartitionApplyOtlChangeRecs
      (
     hCtx
     ,
     pszChgFileName
     ,
     MetaChangeReadHandle
     ,
     SourceTime
     )
ByVal
     hCtx
```

```
      As Long
ByVal
      pszChgFileName
      As String
ByVal
      MetaChangeReadHandle
      As Long
ByVal
      SourceTime
      As Long
```

パラメータ	説明
hCtx	API コンテキストへのハンドル。
pszChgFileName	変更ファイルの名前。
MetaChangeReadHandle	メタデータ変更ファイルへのハンドル。
SourceTime	メタデータ・ファイルの最終変更時刻。

備考

- 変更レコード間には依存関係が存在する場合があります。
- レコードを拒否すると、別のレコードを適用するときに失敗することがあります。たとえば、「A を追加」と「AA を A の子として追加」という 2 つのレコードがあるとします。最初のレコードを拒否して 2 つ目のレコードを受け入れると、適用時にエラーが発生します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャのアクセス権限が必要です。

関連トピック

- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionReadOtlChangeFile
- EsbPartitionResetOtlChangeTime

EsbPartitionGetAreaCellCount

指定したスライス文字列内のセルの数を戻します。

構文

```
      EsbPartitionGetAreaCellCount
       (
      hCtx
      ,
      pszSlice
      ,
      pdCount
      )
ByVal
      hCtx
        As Long
ByVal
      pszSlice
       As String

      pdCount
       As Double
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

pszSlice　　確認対象の入力スライス定義。

pdCount　　ここにセル数が戻されます。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    Public Sub ESB_PartitionGetAreaCellItems()
 Dim Slice As String
 Dim CellItems As Double

 Slice = "@Idescendants(Market)"

 sts = EsbPartitionGetAreaCellCount(hCtx, Slice, CellItems)

 If sts = 0 Then MsgBox Items
End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義 」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells

- EsbPartitionResetOtlChangeTime

EsbPartitionGetList

現在選択されているデータベースが関与しているパーティション定義のリストを戻します。

構文

```
      EsbPartitionGetList
       (
      hCtx
      ,
      SelectRegion
      ,
      pusCount
      )
ByVal
      hCtx
          As Long

      SelectRegion
       As ESB_PARTSLCT_T

      pusCount
        As Integer
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
SelectRegion	パーティションの選択条件。
pusCount	戻されるパーティションのアイテム。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    Public Sub ESB_PartitionGetList()  Dim SelectPartition As ESB_PARTSLCT_T
  Dim Partition As ESB_PART_INFO_T
  Dim Items As Integer
  Dim i As Integer

  SelectPartition.OperationTypes = ESB_PARTITION_OP_ALL
  SelectPartition.DirectionTypes = ESB_PARTITION_DATA_BOTH
  SelectPartition.MetaDirectionTypes = ESB_PARTITION_META_BOTH  sts =
EsbPartitionGetList(hCtx, SelectPartition, Items)

  If sts = 0 And Items > 0 Then
   For i = 1 To Items
    sts = EsbGetNextItem(hCtx, ESB_PART_INFO_TYPE, Partition)
```

```
    '****************************************
    '* Get information in ESB_PART_INFO_T here
     '****************************************    Next i
 End If

End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionGetOtlChanges

指定したソースからアウトラインの変更を取り込み、ファイルに保管します。

構文

```
      EsbPartitionGetOtlChanges
       (
      hCtx, MetaQuery, ChangeFile, szChangeFile
      )
ByVal
      hCtx
          As Long

      MetaQuery
        As ESB_PARTOTL_QUERY_T
ByVal
      ChangeFile
       As String
ByVal
      szChangeFile
       As Integer
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
MetaQuery	変更クエリー条件。
ChangeFile	呼出し元で割り当てた変更ファイルおよび情報構造体。
szChangeFile	変更ファイルのサイズ。

備考

複数のファイルを CR/LF で区切られたファイル・リストとして渡す必要があります。サーバー上のパス名を使用する必要があります(**EsbGetOtlChanges()**で参照される)。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャのアクセス権限が必要です。

例

```
    Public Sub ESB_PartitionGetOtlChanges()  Dim PartQuery As ESB_PARTOTL_QUERY_T
 Const SizeofChangeFile = 150
 Dim ChangeFile As String * SizeofChangeFile

 PartQuery.OperationType = ESB_PARTITION_OP_REPLICATED
 PartQuery.HostDatabase.HostName = "Dscharton2"  PartQuery.HostDatabase.AppName =
"Sampeast"
 PartQuery.HostDatabase.DbName = "East"
  PartQuery.MetaFilter.TimeStamp = _
  DateDiff("s", #1/1/70#, #6/18/97#)  PartQuery.MetaFilter.DimFilter =
ESB_PARTITION_OTLDIM_ALL
 PartQuery.MetaFilter.MbrFilter = ESB_PARTITION_OTLMBR_ALL
 PartQuery.MetaFilter.MbrAttrFilter = _
             ESB_PARTITION_OTLMBRATTR_ALL

 sts = EsbPartitionGetOtlChanges(hCtx, PartQuery, _
           ChangeFile, SizeofChangeFile)  If sts = 0 Then MsgBox ChangeFile

End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionGetReplCells

複製パーティションで識別されているすべてのデータ・セルを、ソース・データベースから選択したターゲット・データベースに複製します。

構文

```
      EsbPartitionGetReplCells
       (
      hCtx, ReplicatedRegion, HostAppDbList
      )
ByVal
      hCtx
            As Long

      ReplicatedRegion
       As ESB_PART_REPL_T
ByVal
      HostAppDbList
        As String
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
ReplicatePartition	パーティション情報。
HostAppDbList	CR/LF で区切られたサーバー、アプリケーションおよびデータベース・セットの文字列。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャのアクセス権限が必要です。

例

```
    Public Sub ESB_PartitionGetReplCells()  Dim ReplPartition As ESB_PART_REPL_T
 Dim HostAppDbList As String
 Dim ProcState   As ESB_PROCSTATE_T
 Dim ind, i    As Long


 ReplPartition.PartitionCount  = -1  'All areas
 ReplPartition.UpdatedOnly   = 0 'Updated only cells
 HostAppDbList         = "localhost" & vbCrLf & _
               "Sampeast" & vbCrLf & _
               "East"

 sts = EsbPartitionGetReplCells(hCtx, ReplPartition, HostAppDbList)

 If sts = 0 Then
  sts = EsbGetProcessState(hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
   sts = EsbGetProcessState(hCtx, ProcState)
  Loop
 End If
End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義 」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionPurgeOtlChangeFile

TimeStamp パラメータで指定した時刻より前に行われた変更を削除します。

構文

```
      EsbPartitionPurgeOtlChangeFile
       (
      hCtx
      ,
      pRegion
      ,
      TimeStamp
      )
ByVal
      hCtx
         As Long

      pRegion
        As ESB_PART_DEFINED_T
ByVal
      TimeStamp
       As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

pRegion　　パーティション定義。

TimeStamp　この時刻より前のすべての変更レコードを削除します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

例

```
    Public Sub Esb_PartitionPurgeOtlChangeFile()

 Dim PartitionInfo As ESB_PART_DEFINED_T
 Dim TimeStamp As Variant
```

```
 PartitionInfo.usType = ESB_PARTITION_OP_REPLICATED
 PartitionInfo.Direction = ESB_PARTITION_DATA_SOURCE

 PartitionInfo.HostDatabase.HostName = "Jsnider"
 PartitionInfo.HostDatabase.AppName = "Samppart"
 PartitionInfo.HostDatabase.DbName = "Company"

 TimeStamp = DateDiff("s", #1/1/70#, #7/7/97#)
 sts = EsbPartitionPurgeOtlChangeFile(hCtx, _
     PartitionInfo, TimeStamp)

End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionPutReplCells

複製パーティションで識別されているすべてのデータ・セルを、選択したソース・データベースからターゲット・データベースに複製します。

構文

```
      EsbPartitionPutReplCells
       (
      hCtx, ReplicatedRegion, HostAppDbList
      )
ByVal
      hCtx
            As Long

      ReplicatedRegion
       As ESB_PART_REPL_T
ByVal
      HostAppDbList
        As String
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
ReplicatedPartition	パーティション情報。
HostAppDbList	ホスト・サーバー上のデータベースまたはアプリケーション、またはその両方のリスト。

備考

このルーチンは、削除後ファイルが空になれば削除します。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャ権限が必要です。

例

```
    Public Sub ESB_PartitionPutReplCells()

 Dim ReplPartition As ESB_PART_REPL_T
 Dim HostAppDbList As String
 Dim ProcState   As ESB_PROCSTATE_T
 Dim ind, i     As Long


 ReplPartition.PartitionCount  = -1  'All areas
 ReplPartition.UpdatedOnly    = 0 'Updated only cells
 HostAppDbList          = "localhost" & vbCrLf & _
                "Sampeast" & vbCrLf & _
                "East"

 sts = EsbPartitionPutReplCells(hCtx, ReplPartition, HostAppDbList)

 If sts = 0 Then
  sts = EsbGetProcessState(hCtx, ProcState)
  Do Until ProcState.State = ESB_STATE_DONE
    sts = EsbGetProcessState(hCtx, ProcState)
  Loop
 End If

End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionResetOtlChangeTime

EsbPartitionReadOtlChangeFile

ターゲット・データベース上の変更ファイル(*.CHG)から、変更をメモリーに読み込みます。この関数は、**EsbPartitionGetOtlChanges()**を呼び出した後に、

EsbPartitionApplyOtlChangeRecs()と、対話的に使用するように設計されています。この関数はフィルタとともに使用できます。

構文

```
      EsbPartitionReadOtlChangeFile
       (
      hCtx
      ,
      pszChgFileName
      ,
      QueryFilter
      ,
      MetaChangeReadHandle
      ,
      SourceTime
      )
ByVal
      hCtx
       As Long
ByVal
      pszChgFileName
       As String
ByRef
      QueryFilter
       As ESB_PARTOTL_QRY_FILTER_T
ByRef
      MetaChangeReadHandle
       As Long
ByRef
      SourceTime
       As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
pszChgFileName	メタデータ変更ファイル名。
QueryFilter	クエリーのフィルタ式。
MetaChangeReadHandle	メタデータ変更ファイルのハンドル。
SourceTime	メタデータ・ファイルの最終変更時刻。

備考

このルーチンでは pMetaChangeRead に時刻が戻されます。これは、ターゲット・データベースのタイムスタンプを更新するために EsbPartitionApplyOtlChangeRecs に渡すタイム・スタンプと同一です。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャのアクセス権限が必要です。

関連トピック

- EsbPartitionApplyOtlChangeFile
- EsbPartitionApplyOtlChangeRecs
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells
- EsbPartitionResetOtlChangeTime

EsbPartitionResetOtlChangeTime

ソース・パーティションから最終変更時刻を取得し、宛先パーティションの最終メタ変更時刻として割り当てます。

構文

```
      EsbPartitionResetOtlChangeTime
       (
      hCtx
      ,
      pSourceRegion
      ,
      pDestRegion
      )
ByVal
      hCtx
           As Long

      pSourceRegion
       As ESB_PART_DEFINED_T

      pDestRegion
        As ESB_PART_DEFINED_T
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
pSourceRegion	新規時刻のパーティション。
pDestRegion	時刻がリセットされるパーティション。

備考

- ソース・パーティションとはタイム・スタンプを提供する側のパーティションを指し、ターゲット・パーティションとはタイム・スタンプを受け取る側のパーティションを指します。
- ソース・パーティションは、データ・ソース・パーティションまたはアウトライン・ソース・パーティションである必要はありません。

戻り値

正常終了の場合は 0 が戻され、失敗した場合はエラー・コードが戻されます。

アクセス

この関数を呼び出すには、データベース・マネージャのアクセス権限が必要です。

例

```
    Public Sub ESB_PartitionResetOtlChangeTime()

 Dim SourcePartition As ESB_PART_DEFINED_T
 Dim DestPartition As ESB_PART_DEFINED_T

 SourcePartition.usType = ESB_PARTITION_OP_REPLICATED
 DestPartition.usType = ESB_PARTITION_OP_REPLICATED

 SourcePartition.Direction = ESB_PARTITION_DATA_SOURCE
 DestPartition.Direction = ESB_PARTITION_DATA_TARGET

 SourcePartition.HostDatabase.HostName = "Dscharton2"
 DestPartition.HostDatabase.HostName = "Dscharton2"

 SourcePartition.HostDatabase.AppName = "Sampeast"
 DestPartition.HostDatabase.AppName = "East"

 SourcePartition.HostDatabase.DbName = "Samppart"
 DestPartition.HostDatabase.DbName = "Company"

 sts = EsbPartitionResetOtlChangeTime(hCtx, _
     SourcePartition, DestPartition)

End Sub
```

関連トピック

- 1257 ページの「パーティションの定数および構造体の定義」
- EsbPartitionApplyOtlChangeFile
- EsbPartitionGetAreaCellCount
- EsbPartitionGetList
- EsbPartitionGetOtlChanges
- EsbPartitionGetReplCells
- EsbPartitionPurgeOtlChangeFile
- EsbPartitionPutReplCells

EsbPutObject

ローカル・ファイルからサーバーまたはクライアントのオブジェクト・システムにオブジェクトをコピーし、オプションでロック解除します。

構文

```
      EsbPutObject
       (
      hCtx, ObjType, AppName, DbName, ObjName, LocalName, isUnlock
      )
ByVal
      hCtx
         As Long
ByVal
      ObjType
        As Long
ByVal
      AppName
        As String
ByVal
      DbName
         As String
ByVal
      ObjName
        As String
ByVal
      LocalName
       As String
ByVal
      isUnlock
       As Integer
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。

ObjType　オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。

AppName　アプリケーション名。

DbName　データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。

ObjName　配置するオブジェクト名。

LocalName　クライアント上のローカル・ソース・ファイルのフル・パス名。

isUnlock　オブジェクトのロック解除を制御するフラグ。TRUE の場合、サーバー・オブジェクトのロックが解除され、他のユーザーによる更新が許可されます。

備考

サーバー上に存在しているオブジェクトを配置するには、呼出し元によって事前にロックされている必要があります。オブジェクトがサーバー上になければ、新規作成されます。

戻り値

正常終了の場合は、オブジェクトが LocalName で指定したローカル・ファイルからサーバーにコピーされます。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベース(あるいはその両方)に対して、呼出し元が(オブジェクト・タイプに応じて)適切なレベルのアクセス権を持っている必要があります。オブジェクトのロックを解除するには(unlock フラグが TRUE)、呼出し元はオブジェクトを含む指定のアプリケーションまたはデータベースに対するアプリケーションまたはデータベースのデザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)が必要です。

例

```
    Declare Function EsbPutObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType As
Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String,
ByVal LocalName As String, ByVal Unlock As Integer) As Long

Sub ESB_PutObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer
 Dim LocalName As String
 Dim Unlock As Integer  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
 ObjType = ESB_OBJTYPE_TEXT
 LocalName = "C:\ESSBASE\CLIENT\BASIC.TXT"
 Unlock = ESB_YES  '***********
 ' Put Object
 '***********
 sts = EsbPutObject (hCtx, ObjType, AppName,
    DbName, ObjName, LocalName, Unlock)
End Sub
```

関連トピック

- EsbGetObject
- EsbLockObject
- EsbUnlockObject
- EsbListObjects

EsbQueryDatabaseMembers

レポートスタイルのクエリーを実行して、選択したデータベース・メンバーの情報をリストします。

構文

```
      EsbQueryDatabaseMembers
       (
      hCtx, mbrQuery
      )
ByVal
      hCtx
        As Long
ByVal
      mbrQuery
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
mbrQuery	メンバー・クエリー文字列。クエリー文字列は、レポート指定に類似のコマンドです。レポート指定に関する詳細は、『Oracle Essbase テクニカル・リファレンス』を参照してください。有効なクエリー文字列は下のノートにリストされています。文字列の長さは 64KB 未満である必要があります。

備考

- 空の文字列が戻されるまで **EsbGetString()**を呼び出して、このクエリーから戻されたメンバー情報を読み取る必要があります。
- この関数では、属性メンバーロング名がサポートされます。
- メンバー・クエリー文字列は、選択文字列、オプションのソート・コマンドおよび後続のオプション出力コマンドから構成されます。次の形式を使用します:

```
mbrQuery ==: <selectionstring> [<sortcommand> [<outputcommand>] ]
```

- メンバー<selectionstring>に有効な値は次のとおりです:

```
<CHILDRENOF  -- returns ICHILDRENOF
<ALLINSAMEDIM
<DIMTOP
<OFSAMEGENERATION
<ONSAMELEVELAS
<ANCESTORSOF -- returns IANCESTORSOF
<PARENTOF
<DESCENDANTSOF -- returns IDESCENDANTSOF
<ALLSIBLINGSOF
<LSIBLINGOF
```

- <sortcommand>に有効な値は次のとおりです:

```
        <SORTASCENDING
<SORTDESCENDING
<SORTNONE
<SORTMBRNAMES
<SORTALTNAMES
<SORTMBRNUMBERS
<SORTDIMNUMBERS
<SORTLEVELNUMBERS
<SORTGENERATION
```

- <outputcommand>の形式は次のとおりです:

```
        <outputcommand> ==: Item [separator] | FORMAT {<item> <separator> }
```

  - メンバー情報に関する 1 アイテムのリストを取得するには、次の出力コマンドを使用します:

```
          <outputcommand> ==: <MBRNAMES |
        <ALTNAMES |
        <MBRNUMBERS |
        <DIMNUMBERS |
        <LEVELNUMBERS |
        <GENERATIONS |
        <CALCSTRINGS |
        <UCALCS |
        <TABSEPARATED |
        <SPACESEPARATED |
        <COMMASEPARATED |
        <NEWLINESEPARATED |
        <ATTRIBUTES
```

  - メンバーの複数の情報アイテムのリストを取得するには、フォーマット指定句を使用します。リストしたいアイテム、順序、区切り文字を指定します。フォーマット指定句の構文は次のとおりです:

```
          <FORMAT <item> [<separator>] {<item> [<separator>]}
```

    <item>に有効な値は次のとおりです:

```
          MBRNAMES
ALTNAMES
MBRNUMBERS
DIMNUMBERS
LEVELNUMBERS
GENERATIONS
CALCSTRINGS
UCALCS
ATTRIBUTES
```

ATTRIBUTES は、属性の数と、それに続く属性名のタブ区切りリストとしてリストされます。

<separator>に有効な値は次のとおりです:

```
           TABSEPARATED
SPACESEPARATED
COMMASEPARATED
NEWLINESEPARATED
```

区切り文字を指定しない場合のデフォルトは TABSEPARATED です。

- 以下にスクリプト例を示します:

```
        login "local" "user1" "password" "" ""
select "attr" "attr"
GetMembers "<NEWLINESEPARATED
<FORMAT {
MBRNAMES    SPACESEPARATED ALTNAMES   TABSEPARATED
MBRNUMBERS   SPACESEPARATED DIMNUMBERS  TABSEPARATED
LEVELNUMBERS  SPACESEPARATED GENERATIONS TABSEPARATED
CALCSTRINGS   SPACESEPARATED UCALCS    TABSEPARATED
DIMTYPES    SPACESEPARATED STATUSES   TABSEPARATED
ATTRIBUTES
}
<DESCENDANTS Product "
```

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
   Declare Function EsbQueryDatabaseMembers Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
Query As String) As Long

Sub ESB_QueryDatabaseMembers ()
 Dim sts As Long
 Dim Query As String
 Const szMString = 256
 Dim MString As String * szMString
 Query = "<ALLINSAMEDIM"   '************************
 ' Query Database members
 '************************
 sts = EsbQueryDatabaseMembers (hCtx, Query)   '************************
 ' Print out all strings
 '************************
 If sts = 0 Then
  sts = EsbGetString (hCtx, MString,
```

```
     szMString)
  Do While Mid$(MString, 1, 1) <> Chr$(0)
   Print MString
   sts = EsbGetString (hCtx, MString,
      szMString)
  Loop
 End If
End Sub
```

関連トピック

- EsbCheckMemberName
- EsbGetMemberInfo
- EsbGetString
- EsbSetActive

EsbRemoveAlias

アクティブなデータベースから別名テーブルを完全に削除します。

構文

```
      EsbRemoveAlias
       (
      hCtx, AltName
      )
ByVal
      hCtx
        As Long
ByVal
      AltName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AltName	削除する別名テーブルの名前。

備考

- 「デフォルト」または現在アクティブな別名テーブルは、削除できません。
- **EsbListConnections()**を呼び出して、別名テーブルを削除しようとしているデータベースを誰も使用していないことを確認してください。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースに対してアクセス権を持っていて、**EsbSetActive()**を使用してこれをアクティブなデータベースとして選択している必要があります。

例

```
    Declare Function EsbRemoveAlias Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Name As
String) As Long

Sub ESB_RemoveAlias ()
 Dim sts As Long
 Dim Name As String  Name = "TestAlias"  '*************
 ' Remove Alias
 '*************
 sts = EsbRemoveAlias (hCtx, Name)
End Sub
```

関連トピック

- EsbClearAliases
- EsbListAliases
- EsbLoadAlias
- EsbListConnections
- EsbSetActive

EsbRemoveLocks

現在ユーザーがロックしているデータベースのデータ・ブロックのロックをすべて解除します。

構文

```
      EsbRemoveLocks
       (
      hCtx, AppName, DbName, LoginId
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      LoginId
       As Long
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　データベース名。

パラメータ 説明

LoginId　　ロックが解除されるユーザー・ログインの ID。

備考

- 必要とされる LoginId は、EsbListLocks 関数によって戻されるユーザー・ロック情報構造体から取得できます。
- LoginId で指定したユーザーが現在ログインしている場合、**EssRemoveLocks()** はそのユーザーとの接続を終了します。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbRemoveLocks Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, ByVal LoginId As Long) As Long

Sub ESB_RemoveLocks ()
 Dim Items As Integer
 Dim AppName As String
 Dim DbName As String
 Dim LockInfo As ESB_LOCKINFO_T
 Dim sts As Long  AppName = "Sample"
 DbName = "Basic"
 '***********
 ' List Locks
 '***********
 sts = EsbListLocks (hCtx, AppName, DbName,
    Items)
 '*****************
 ' Remove all locks
 '*****************
 For n = 1 To Items
  '******************************
  ' Get next user lock structure
  ' from the list and remove locks
  '******************************
  sts = EsbGetNextItem (hCtx,
     ESB_LOCKINFO_TYPE, LockInfo)
  sts = EsbRemoveLocks (hCtx, AppName,
     DbName, LockInfo.LoginId)
 Next
End Sub
```

関連トピック

- EsbListLocks

EsbRenameApplication

クライアントまたはサーバー上のいずれかの既存のアプリケーションの名前を変更します。アプリケーションがサーバー上で実行されている場合、最初に停止されます。

構文

```
      EsbRenameApplication
       (
      hCtx, AppName, nAppName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      nAppName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	名前を変更すべき既存のアプリケーションの名前。
nAppName	アプリケーションの新しい名前。1903 ページの「アプリケーション名の制限」を参照してください。

備考

クライアント・アプリケーションの名前を変更すると、ローカル・アプリケーションのディレクトリ名も変更されます。

戻り値

なし。

アクセス

サーバー・アプリケーションの場合、呼出し元はアプリケーションの作成/削除/編集権限(ESB_PRIV_APPCREATE)を持っている必要があります。

例

```
    Declare Function EsbRenameApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
OldName As String, ByVal NewName As String) As Long

Sub ESB_RenameApplication ()
 Dim sts As Long
 Dim OldName As String
 Dim NewName As String  OldName = "Sample"
 NewName = "NewSamp"  '********************
 ' Rename Application
```

```
  '*******************
  sts = EsbRenameApplication (hCtx, OldName,
     NewName)
End Sub
```

関連トピック

- EsbRenameDatabase
- EsbRenameObject

EsbRenameDatabase

クライアントまたはサーバー上で、アプリケーション内の既存のデータベースの名前を変更します。データベースがサーバー上で実行されている場合、最初に停止されます。

構文

```
      EsbRenameDatabase
       (
      hCtx, AppName, DbName, nDbName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      nDbName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()から戻されたローカル・コンテキスト・ハンドルの場合もあります。
AppName	アプリケーション名。
DbName	名前を変更する既存のデータベースの名前。
nDbName	データベースの新しい名前。1903 ページの「データベース名の制限」を参照してください。

備考

クライアント・データベースの名前を変更すると、ローカル・データベースのディレクトリ名も変更されます。

戻り値

なし。

アクセス

サーバー・データベースの場合は、呼出し元がデータベースの作成/削除/編集権限(ESB_PRIV_DBCREATE)を持っている必要があります。

例

```
    Declare Function EsbRenameDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal OldName As String, ByVal NewName As String) As Long

Sub ESB_RenameDatabase ()
 Dim sts As Long
 Dim AppName As String
 Dim OldName As String
 Dim NewName As String   AppName = "Sample"
 OldName = "Basic"
 NewName = "NewBasic"   '****************
 ' Rename database
 '****************
 sts = EsbRenameDatabase (hCtx, AppName,
    OldName, NewName)
End Sub
```

関連トピック

- EsbRenameApplication
- EsbRenameObject

EsbRenameFilter

既存のフィルタの名前を変更します。

構文

```
      EsbRenameFilter
       (
      hCtx, AppName, DbName, FltName, nFltName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FltName
       As String
ByVal
      nFltName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
FltName	名前を変更する既存のフィルタの古い名前。
nFltName	名前を変更したフィルタの新しい名前。1904 ページの「フィルタ名の制限」を参照してください。

備考

古いフィルタ名が存在し、新しいフィルタ名が存在していないことが必要です。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbRenameFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String, ByVal OldName As String, ByVal NewName As String)
As Long

Sub ESB_RenameFilter ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim OldName As String
 Dim NewName As String  AppName = "Sample"
 DbName = "Basic"
 OldName = "Filter"
 NewName = "NewFilter"  '**************
 ' Rename Filter
 '**************
 sts = EsbRenameFilter (hCtx, AppName,DbName,
    OldName, NewName)
End Sub
```

関連トピック

- EsbCopyFilter
- EsbDeleteFilter
- EsbListFilters
- EsbSetFilter

EsbRenameGroup

既存のグループの名前を変更します。

構文

```
      EsbRenameGroup
       (
      hCtx, GrpName, nGrpName
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
ByVal
      nGrpName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
GrpName	名前を変更する既存のグループの古い名前。
nGrpName	名前を変更したグループの新しい名前。1904 ページの「グループ名の制限」を参照してください。

備考

指定された新規グループ名が存在していることが必要です。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbRenameGroup Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String, ByVal nGrpName As String) As Long

Sub ESB_RenameGroup ()
 Dim sts As Long
 Dim OldName As String
 Dim NewName As String  OldName = "PowerUsers"
 NewName = "NewUsers"  '**************
 ' Rename Group
 '**************
 sts = EsbRenameGroup (hCtx, OldName, NewName)
End Sub
```

関連トピック

- EsbCreateGroup
- EsbDeleteGroup
- EsbListGroups

EsbRenameObject

サーバーまたはクライアント・オブジェクト・システム上の既存のオブジェクトの名前を変更します。

構文

```
      EsbRenameObject
       (
      hCtx, ObjType, AppName, DbName, ObjName, nObjName
      );
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      ObjName
       As String
ByVal
      nObjName
      As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
OldName	名前を変更するオブジェクトの古い名前。
NewName	名前を変更したオブジェクトの新しい名前。1904 ページの「オブジェクト名の制限」を参照してください。

備考

- オブジェクト名を変更するには、そのオブジェクトがロックされていないことと、新しいオブジェクトが存在していないことが必要です。
- アウトライン・オブジェクトおよびLROオブジェクトの名前は、変更できません。
- 関連付けられたアウトラインも含めてデータベースの名前を変更する場合は、**EsbRenameDatabase()**関数を使用します。
- 異なるアプリケーションやデータベースではオブジェクトの名前を変更できません。**EsbCopyObject()**関数を使ってオブジェクトを別のアプリケーションまたはデータベースにコピーしてください。

戻り値

なし。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベースに対して、呼出し元がアプリケーション・デザイン権限またはデータベース・デザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbRenameObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal OldName As String,
ByVal NewName As String) As Long

Sub ESB_RenameObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim OldName As String
 Dim NewName As String
 Dim ObjType As Integer  AppName = "Sample"
 DbName = "Basic"
 OldName = "Basic"
 NewName = "NewBasic"
 ObjType = ESB_OBJTYPE_OUTLINE  '********************
 ' Rename Rules Object
 '********************
 sts = EsbRenameObject (hCtx, ObjType, AppName,
    DbName, OldName, NewName)
End Sub
```

関連トピック

- EsbCopyObject
- EsbCreateObject
- EsbDeleteObject
- EsbListObjects
- EsbUnlockObject

EsbRenameUser

既存のユーザー名を変更します。

構文

```
      EsbRenameUser
       (
      hCtx, UserName, nUserName
      )
ByVal
      hCtx
         As Long
ByVal
      UserName
       As String
ByVal
      nUserName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
UserName	名前を変更する既存のユーザーの古い名前。
nUserName	名前を変更したユーザーの新しい名前。1904 ページの「ユーザー名の制限」を参照してください。

備考

指定された新規ユーザー名が存在していない必要があります。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbRenameUser Lib "ESBAPIN" (ByVal hCtx As Long, ByVal OldName
As String, ByVal NewName As String) As Long

Sub ESB_RenameUser ()
 Dim sts As Long
 Dim OldName As String
 Dim NewName As String  OldName = "Joseph"
 NewName = "Joe"  '************
 ' Rename user
 '************
 sts = EsbRenameUser (hCtx, OldName, NewName)
End Sub
```

関連トピック

- EsbCreateUser
- EsbDeleteUser
- EsbListUsers

EsbReport

アクティブなデータベースに対して単一の文字列としてレポート指定を送信します。

構文

```
      EsbReport
       (
      hCtx, isOutput, isLock, rptQuery
      )
ByVal
      hCtx
        As Long
ByVal
      isOutput
       As Integer
ByVal
      isLock
        As Integer
ByVal
      rptQuery
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。
isOutput	データの出力を制御します。TRUE の場合は、指定したレポートに従ってサーバーから出力されます。FALSE の場合は、データは出力されません。
isLock	ブロックのロックを制御します。TRUE の場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSE の場合は、ブロックのロックは行われません。
rptQuery	単一文字列としてのレポート指定(64KB 未満)。

備考

- この関数は、**EsbBeginReport()**を呼び出し、その後 **EsbSendString()**を呼び出して、最後に **EsbEndReport()**を呼び出すのと同じです。レポート・データは出力することも、レポート指定の確認のみ行い、エラーがあれば戻させることもできます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。
- レポート指定の文字列の長さは、64KB 未満である必要があります。

- この関数によってデータが出力される場合(Output フラグが TRUE)、空の文字列が戻されるまで、**EsbGetString()**を呼び出して戻されたデータを読み取ることができます。
- この関数によってブロックがロックされる場合(Lock フラグが TRUE)、呼出し元がロックされたブロックのロック解除を行う必要があります(たとえば、Unlock フラグを TRUE に設定して **EsbUpdate()**を呼び出します)。
- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESB_PRIV_READ)を持っている必要があります。呼出し元がアクセス権を持っていないすべてのメンバーは、不明として戻されます。

例

```
    Declare Function EsbReport Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Output As
Integer, ByVal Lock As Integer, ByVal Query As String) As Long

Sub ESB_Report ()
 Dim sts As Long
 Dim pOutput As Integer
 Dim pLock As Integer
 Dim Query As String
 Const szRString = 256
 Dim RString As String * szRString  Query = "<Desc Year !"
 pOutput = ESB_YES
 pLock = ESB_NO
 '***********
 ' Run Report
 '***********
 sts = EsbReport (hCtx, pOutput, pLock, Query)   '**********************
 ' Print out all strings
 '**********************
 If sts = 0 Then
  sts = EsbGetString (hCtx, RString,
     szRString)
  Do While Mid$(RString, 1, 1) <> Chr$(0)
    Print RString
    sts = EsbGetString (hCtx, RString,
       szRString)
  Loop
 End If
End Sub
```

関連トピック

- EsbBeginReport

- EsbEndReport
- EsbGetString
- EsbReportFile
- EsbUpdate
- EsbSendString

EsbReportFile

ファイルからアクティブなデータベースへレポート指定を送信します。レポート・データを出力できます。または、レポート指定の確認のみも可能です。エラーがあれば戻されます。また、この呼出しでは、オプションでデータベース内の対応するデータ・ブロックをロックすることもできます(更新用のロック)。

構文

```
      EsbReportFile
       (
      hDestCtx, hSrcCtx, AppName, DbName, FileName, isOutput, isLock
      )
ByVal
      hDestCtx
       As Long
ByVal
      hSrcCtx
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FileName
       As String
ByVal
      isOutput
       As Integer
ByVal
      isLock
        As Integer
```

パラメータ	説明
hDestCtx	サーバー上のターゲット・データベースの VB API コンテキスト・ハンドル。
hSrcCtx	レポート・ファイルの場所の VB API コンテキスト・ハンドル。レポート・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。レポート・ファイルがクライアント(ローカル)にある場合、ローカルのコンテキストは EsbCreateLocalContext で作成する必要があります。
AppName	レポート・ファイルの場所に対するアプリケーション名。
DbName	レポート・ファイルの場所に対するデータベース名。

パラメータ	説明
FileName	レポート指定ファイル名。拡張子は.rep であることがわかっているため、ファイル拡張子を指定する必要はありません。
isOutput	データの出力を制御します。TRUE の場合は、指定したレポートに従ってサーバーから出力されます。FALSE の場合は、データは出力されません。
isLock	ブロックのロックを制御します。TRUE の場合は、レポート指定でアクセスされるすべてのブロックが更新用にロックされます。FALSE の場合は、ブロックのロックは行われません。

備考

- この関数によってデータが出力される場合(Output フラグが TRUE)、**EsbGetString()**を呼び出して戻されたデータを読み取ることができます。
- この関数によってブロックがロックされる場合(Lock フラグが TRUE)、呼出し元がロックされたブロックのロック解除を行う必要があります(たとえば、Unlock フラグを TRUE に設定して **EsbUpdate()**を呼び出します)。
- Output および Lock の両方のフラグが FALSE に設定されている場合、データベースはレポート指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESB_PRIV_READ)を持っている必要があります。

例

```
    Declare Function EsbReportFile Lib "ESBAPIN" (ByVal hDestCtx As Long, ByVal
hSrcCtx As Long, ByVal AppName As String, ByVal DbName As String, ByVal FileName As
String, ByVal Output As Integer, ByVal Lock As Integer) As Long

Sub ESB_ReportFile ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FileName As String
 Dim pOutput As Integer
 Dim pLock As Integer
 Dim hSrcCtx As Long Const szRString = 256
 Dim RString As String * szRString AppName = "Sample"
 DbName = "Basic"
 hSrcCtx = hCtx
 FileName = "test"
 pOutput = ESB_YES
 pLock = ESB_NO sts = EsbReportFile(hCtx, hSrcCtx, AppName,
 DbName, FileName, pOutput, pLock)
 If sts = 0 Then
  sts = EsbGetSTring(hCtx, RString, szRString)
 Do While Mid$(RString, 1, 1) <> Chr$(0)
  Print RString
```

```
   sts = EsbGetSTring(hCtx, RString, szRString)
  Loop
 End If
End Sub
```

関連トピック

- EsbBeginReport
- EsbGetString
- EsbReport
- EsbUpdateFile

EsbResetUser

ユーザーのセキュリティ構造体を最初の状態にリセットします。

構文

```
      EsbResetUser
      (
      hCtx
      ,
      UserName
      )
ByVal
      hCtx
        As Long
ByVal
      UserName
       As String
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル
UserName	ユーザー名

備考

次のユーザー・セキュリティ・パラメータは、初期状態にリセットされます:

- LockedOut
- PwdChgNow
- Failcount
- LastLogin
- LastPwdChg
- Expiration

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbResetUser Lib "esbapin" (ByVal hCtx As Long, _
                    ByVal UserName As String) As Long

Sub ESB_ResetUser ()
 Dim UserName As String
 Dim sts    As Long  UserName = "William"
 sts    = EsbResetUser (hCtx, UserName)
End Sub
```

関連トピック

- EsbInit
- EsbLogin
- EsbLogout

EsbRestore

使用されなくなりました。

この関数は、Essbase の以前のリリースとの互換性のためにのみ保持されています。現在の Essbase アーカイブには、**EsbArchiveBegin()**および **EsbArchiveEnd()**を参照してください。この関数は、エラー・メッセージ **ESB_STS_OBSOLETE** を戻します。

関連トピック

- EsbArchive
- EsbGetProcessState
- EsbSetActive
- EsbArchiveBegin
- EsbArchiveEnd

EsbSendString

アクティブ・データベースにデータの文字列を送信します。文字列の長さは 32KB 未満にする必要があります。この関数は、**EsbBeginReport()**、**EsbBeginUpdate()**または **EsbBeginCalc()**を呼び出した後に呼び出す必要があります。

構文

```
    EsbSendString
```

```
      (
      hCtx, sndString
      )
ByVal
      hCtx
         As Long
ByVal
      sndString
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

sndString　データ文字列の長さは 32KB 未満にする必要があります。

備考

- レポートの開始、更新または関数の計算が正常に実行されないうちにこの関数を呼び出すと、エラーが発生します。
- 送信する文字列の長さは、32KB 未満とする必要があります。
- この関数を **EsbBeginUpdate()** とともに使用する場合、更新した文字列の最後に改行記号を付ける必要があります。

戻り値

なし。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
   Declare Function EsbSendString Lib "ESBAPIN" (ByVal hCtx As Long, ByVal sndString
As String) As Long
```

EsbBeginReport および EsbBeginUpdate の例を参照してください。

関連トピック

- EsbBeginCalc
- EsbBeginReport
- EsbBeginUpdate
- EsbGetString

EsbSetActive

呼出し元のアクティブなアプリケーションとデータベースを設定します。

構文

```
      EsbSetActive
       (
```

```
      hCtx, AppName, DbName, pAccess
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      pAccess
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。
pAccess	選択したデータベースへのユーザーのアクセス・レベルを受け取る変数のアドレス。このフィールドに使用できる値のリストは、表 15 についての説明を参照してください。

備考

- アプリケーションおよびデータベースがロードされていない場合、これらはこの関数によってロードされます。
- また、ユーザーが **EsbAutoLogin()**関数を使用してログインし、アクティブなアプリケーションおよびデータベースを設定することもできます。

戻り値

正常終了の場合は、選択したアプリケーションおよびデータベースに対するユーザーのアクセス・レベルが pAccess に戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbSetActive Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, Access As Integer) As Long

Sub ESB_SetActive ()
 Dim AppName As String
 Dim DbName As String
 Dim pAccess As Integer
 Dim sts As Long  AppName = "Demo"
 DbName = "Basic"
 '*********************************
 ' Set active Application & Database
 '*********************************
 sts = EsbSetActive (hCtx, AppName, DbName, pAccess)
```

```
End Sub
```

関連トピック

- EsbClearActive
- EsbGetActive
- EsbListApplications
- EsbListDatabases
- EsbLogin
- EsbSetActive

EsbSetAlias

1 人のユーザーについて、アクティブなデータベースにアクティブな別名テーブルを設定します。

構文

```
      EsbSetAlias
       (
      hCtx, AltName
      )
ByVal
      hCtx
        As Long
ByVal
      AltName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AltName	アクティブに設定する別名テーブルの名前。

戻り値

なし。

例

```
    Declare Function EsbSetAlias Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Name As
String) As Long

Sub ESB_SetAlias ()
 Dim sts As Long
 Dim pName As String pName = "TestAlias"
 '**********
 ' Set Alias
 '**********
 sts = EsbSetAlias (hCtx, pName)
End Sub
```

関連トピック

- EsbGetAlias
- EsbListAliases

EsbSetApplicationAccess

アプリケーションへのユーザー・アクセスに関する情報が含まれているユーザーのアプリケーション・アクセス構造体を設定します。

構文

```
      EsbSetApplicationAccess
       (
      hCtx, Items, pUserApp
      )
ByVal
      hCtx
        As Long
ByVal
      Items
        As Long
ByVal
      pUserApp
       As ESB_USERAPP_T
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
Items	今後使用するために予約されています。
pUserApp	ユーザー・アプリケーション構造体へのポインタ。

備考

ユーザー・アプリケーション構造体の Access フィールドを使用して、ユーザーに付与されたアプリケーションへのアクセスを設定します。この呼出しでは MaxAccess フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、指定されたアプリケーションに対して、呼出し元がアプリケーション・デザイン権限(ESB_PRIV_APPDESIGN)を持っている必要があります。

例

```
   Declare Function EsbSetApplicationAccess Lib "esbapin" (ByVal hCtx As Long, ByVal
Items As Integer, UserApp As ESB_USERAPP_T) As Long
```

```
Sub Esb_SetApplicationAccess ()
 Dim sts As Long
 Dim Items As Integer
 Dim UserApp As ESB_USERAPP_T
 '****************************
 ' Initialize UserApp structure
 '****************************
 UserApp.UserName = "Joseph"
 UserApp.AppName = "Sample"
 UserApp.Access = ESB_ACCESS_SUPER
 UserApp.MaxAccess = ESB_ACCESS_SUPER
 '**************************
 ' Set Administrator access level
 '**************************
 sts = EsbSetApplicationAccess (hCtx, Items,
    UserApp)
End Sub
```

関連トピック

- EsbGetApplicationAccess
- EsbListUsers
- EsbSetDatabaseAccess
- EsbSetUser

EsbSetApplicationState

アプリケーションの状態構造体を使用して、ユーザーが構成可能なアプリケーションのパラメータを設定します。

構文

```
      EsbSetApplicationState
       (
      hCtx, AppName, pAppState
      )
ByVal
      hCtx
         As Long
ByVal
      AppName
        As String
ByVal
      pAppState
       As ESB_PAPPSTATE_T
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。

AppName　アプリケーション名。

pAppState　アプリケーション状態構造体へのポインタ。

備考

- パラメータ値を変更するときは、最初に **EsbGetApplicationState()**を呼び出して、変更しないパラメータの正しい値を取得することをお薦めします。たとえば、次のようにして接続を使用不可にします:

```
        Function ESB_DisableConnects(AppName As String) As Long
  Dim sts As Long
  Dim AppState As ESB_APPSTATE_T

  sts = EsbGetApplicationState(phCtx, AppName, AppState)

  If sts = 0 Then
    AppState.Connects = ESB_FALSE
    sts = EsbSetApplicationState(phCtx, AppName, AppState)
  End If
  ESB_SetApplicationState = sts
 End Function
```

- 次のパラメータは集約ストレージ・データベースに適用されません:LockTimeout および lroSizeLimit。

戻り値

なし。

アクセス

この関数を使用するには、指定したアプリケーションに対して、呼出し元がアプリケーション・マネージャ権限(ESB_PRIV_APPDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetApplicationState Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, AppState As ESB_APPSTATE_T) As Long

Sub ESB_SetApplicationState ()
  Dim sts As long
  Dim AppName As String
  Dim AppState As ESB_APPSTATE_T  AppName = "Sample"
  AppState.Description = "This is a test application"
  AppState.Loadable = ESB_TRUE
  AppState.Autoload = ESB_TRUE
  AppState.Access = ESB_PRIV_APPCREATE
  AppState.Connects = ESB_TRUE
  AppState.Commands = ESB_TRUE
  AppState.Updates = ESB_TRUE
  AppState.Security = ESB_TRUE
  AppState.LockTimeout = 1000   '*******************************
  ' Set Application State structure
  '*******************************
  sts = EsbSetApplicationState (hCtx, AppName, AppState)
End Sub
```

関連トピック

- EsbGetApplicationState
- EsbSetDatabaseState

EsbSetCalcList

ユーザーがアクセス可能な計算スクリプト・オブジェクトのリストを設定します。

構文

```
      EsbSetCalcList
       (
      hCtx, User, AppName, DbName, isAllCalcs, CalcList, Items
      )
ByVal
      hCtx
         As Long
ByVal
      User
         As String
ByVal
      AppName
        As String
ByVal
      DbName
          As String
ByVal
      isAllCalcs
       As Integer
ByVal
      CalcList
        As String
ByVal
      Items
         As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
User	ユーザー名。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます
isAllCalcs	すべての計算を許可するフラグ。TRUE の場合、ユーザーはすべての計算スクリプトにアクセスできます。それ以外の場合は、CalcList 引数で指定されている計算スクリプトにのみアクセスできます。
CalcList	計算スクリプト・オブジェクト名の文字列(CR、EOL で区切られる)。長さは、64KB 未満である必要があります
Items	CalcList 文字列内のアクセス可能な計算スクリプト・オブジェクト数のアイテム。

備考

- AllCalcs フラグが TRUE に設定されている場合、Items および pCalcList 引数は無視されます。
- 計算スクリプト・オブジェクトにアクセスするには、ユーザーは少なくとも該当のデータベースに対して計算アクセス権限を持っている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetCalcList Lib "ESBAPIN" (ByVal hCtx As Long, ByVal User As
String, ByVal AppName As String, ByVal DbName As String, ByVal isAllCalcs As Integer,
ByVal CalcList As String, ByVal Items As Integer) As Long

Sub EsbSetCalcList ()
 Dim sts As Long
 Dim Items As Integer
 Dim User As String
 Dim AppName As String
 Dim DbName As String
 Dim AllCalcs As Integer
 Dim CalcList As String  User = "Joseph"
 AppName = "Sample"
 DbName = "Basic"
 AllCalcs = ESB_NO  '******************************
 ' Initialize CalcList and Items
 '******************************
.
.   '**************
 ' Set Calc list
 '**************
 sts = EsbSetCalcList (hCtx, User, AppName, DbName, AllCalcs, CalcList,
    Items)
End Sub
```

関連トピック

- EsbGetCalcList
- EsbListObjects
- EsbListUsers

EsbSetDatabaseAccess

データベースへのユーザー・アクセスに関する情報が含まれる、ユーザーのデータベース・アクセス構造体を設定します。

構文

```
      EsbSetDatabaseAccess
       (
      hCtx, Items, pUserDb
      )
ByVal
      hCtx
        As Long
ByVal
      Items
       As Integer
ByVal
      pUserDb
       As ESB_USERDB_T
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
Items	今後使用するために予約されています。
pUserDb	ユーザー・データベース構造体へのポインタ。

備考

ユーザー・データベース構造体の Access フィールドを使用して、ユーザーに付与されたデータベースへのアクセスを設定します。この呼出しでは MaxAccess と FilterName フィールドは無視されます。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・マネージャ権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetDatabaseAccess Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
Items As Integer, UserDb As ESB_USERDB_T) As Long

Sub EsbSetDatabaseAccess ()
 Dim sts As Long
 Dim hCtx As Long
 Dim Items As Integer
 Dim UserDb As ESB_USERDB_T  '*****************************
 ' Initialize UserDb structure
 '*****************************
 UserDb.UserName = "Joseph"
 UserDb.AppName = "Sample"
 UserDb.DbName = "Basic"
 UserDb.Access = ESB_ACCESS_SUPER
 UserDb.MaxAccess = ESB_ACCESS_SUPER
```

```
 UserDb.FilterName = ""  '**************************
 ' Set Administrator access level
 '**************************
 sts = EsbSetDatabaseAccess (hCtx, Items, UserDb)
End Sub
```

関連トピック

- EsbGetDatabaseAccess
- EsbListUsers
- EsbSetApplicationAccess
- EsbSetUser

EsbSetDatabaseNote

データベースの最新情報に関するメッセージを設定します。このメッセージを使用して、ユーザーがデータベースに接続する前に、データベースに関する有用な情報(データがロードされているかどうか、データが最後に計算されたのはいつかなど)を表示できます。

構文

```
      EsbSetDatabaseNote
       (
      hCtx, AppName, DbName, DbNote
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      DbNote
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　データベース名。

DbNote　　データベース・ノート文字列へのポインタ。

備考

データベース・ノート文字列の長さは、64KB 未満である必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetDatabaseNote Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, ByVal DbNote As String) As Long

Sub ESB_SetDatabaseNote ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim DbNote As String  AppName = "Sample"
 DbName = "Basic"
 DbNote = "This is a test"
 '******************
 ' Set Database note
 '******************
 sts = EsbSetDatabaseNote (hCtx, AppName,
    DbName, DbNote)
End Sub
```

関連トピック

- EsbGetDatabaseNote

EsbSetDatabaseState

データベースの状態構造体を使用して、ユーザーが構成可能なデータベースのパラメータを設定します。

構文

```
      EsbSetDatabaseState
       (
      hCtx, AppName, DbName, pDbState
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      pDbState
```

```
      As ESB_PDBSTATE_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　データベース名。

pDbState　　データベース状態構造体へのポインタ。

備考

- **EsbGetDatabaseState()**を呼び出して、ESB_DBSTATE_T 構造体を初期化した後、**EsbSetDatabaseState()**を呼び出す必要があります。
- この関数は、ユーザーが構成可能なサーバー・データベースのパラメータのみ設定できます。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetDatabaseState Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, DbState As ESB_DBSTATE_T) As Long

Sub ESB_SetDatabaseState ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim DbState As ESB_DBSTATE_T  AppName = "Sample"
 DbName = "Basic"
 ************************************
 ' Initialize DbState structure fields
 '************************************

 '
    DbState.Description = "This is Sample/Basic"
 DbState.Loadable = "1"
 '*****************************
 ' Set Database state structure
 '*****************************
 sts = EsbSetDatabaseState (hCtx, AppName,
    DbName, DbState)
End Sub
```

関連トピック

- EsbGetDatabaseState

- EsbSetApplicationState

EsbSetDefaultCalc

アクティブ・データベースに対してデフォルト計算スクリプトを設定します。

構文

```
      EsbSetDefaultCalc
       (
      hCtx, cscString
      )
ByVal
      hCtx
         As Long
ByVal
      cscString
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

cscString　デフォルト計算スクリプト文字列。

備考

計算スクリプトの文字列の長さは、64KB 未満である必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbSetDefaultCalc Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
Script As String) As Long

Sub ESB_SetDefaultCalc ()
 Dim sts As Long
 Dim Script As String  Script = "CALC ALL;"  '*****************
 ' Set default calc
 '*****************
 sts = EsbSetDefaultCalc (hCtx, Script)
End Sub
```

関連トピック

- EsbDefaultCalc
- EsbGetDefaultCalc

- EsbSetDefaultCalcFile
- EsbSetActive

EsbSetDefaultCalcFile

計算スクリプト・ファイルからアクティブ・データベースに対してデフォルト計算スクリプトを設定します。

構文

```
      EsbSetDefaultCalcFile
       (
      hDestCtx, hSrcCtx, AppName, DbName, FileName
      )
ByVal
      hDestCtx
       As Long
ByVal
      hSrcCtx
       As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FileName
       As String
```

パラメータ 説明

hDestCtx	サーバー上のターゲット・データベースの VB API コンテキスト・ハンドル。
hSrcCtx	計算スクリプト・ファイルの場所の VB API コンテキスト・ハンドル。計算スクリプト・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。
AppName	計算スクリプト・ファイルの場所に対するアプリケーション名。
DbName	計算スクリプト・ファイルの場所に対するデータベース名。
FileName	デフォルト計算スクリプト・ファイルの名前。

備考

デフォルト計算スクリプトの長さは、64KB 以下である必要があります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元はアクティブなデータベースに対して計算権限(ESB_PRIV_CALC)を持っている必要があります。

例

```
    Declare Function EsbSetDefaultCalcFile Lib "ESBAPIN" (ByVal hDestCtx As Long,
ByVal hSrcCtx As Long, ByVal AppName As String, ByVal DbName As String, ByVal
FileName As String) As Long

Sub ESB_SetDefaultCalcFile ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FileName As String
 Dim hSrcCtx As Long  AppName = "Sample"
 DbName = "Basic"  '************************************
 ' Calc script is an object at the server *
 '************************************
 hSrcCtx = hCtx
 FileName = "calc"   '**********
 ' Calc File
 '**********
 sts = EsbSetDefaultCalcFile (hCtx, hSrcCtx, AppName, DbName, FileName)
End Sub
```

関連トピック

- EsbDefaultCalc
- EsbGetDefaultCalc
- EsbSetDefaultCalc

EsbSetFilter

フィルタのコンテンツの設定を開始します。

構文

```
     EsbSetFilter
      (
     hCtx, AppName, DbName, FltName, isActive, pAccess
     )
ByVal
     hCtx
       As Long
ByVal
     AppName
      As String
ByVal
     DbName
       As String
ByVal
     FltName
      As String
ByVal
     isActive
      As Integer
ByVal
```

```
    pAccess
     As Integer
```

パラメータ 説明

hCtx　VB API コンテキスト・ハンドル。

AppName　アプリケーション名。

DbName　データベース名。

FltName　フィルタ名。1904 ページの「フィルタ名の制限」を参照してください。

isActive　フィルタのアクティブ・フラグ。TRUE の場合はフィルタがアクティブに設定され、TRUE でない場合は非アクティブに設定されます。

pAccess　デフォルトのフィルタ・アクセス・レベル。

備考

- フィルタが存在しない場合は、この呼出しによって最初にそのフィルタが作成されます。
- この呼出しの後に **EsbSetFilterRow()**を続けて呼び出して、フィルタのすべての行を設定する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName As
String, ByVal DbName As String, ByVal FltName As String, ByVal is Active As Integer,
ByVal pAccess As Integer) As Long

Sub ESB_SetFilter()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Dim Active As Integer
 Dim pAccess As Integer
 Dim Row As String AppName = "Demo"
 DbName = "Basic"
 FilterName = "Filter"
 Active = ESB_YES
 pAccess = ESB_ACCESS_DBCREATE '***********
 ' Set Filter
 '***********
 sts = EsbSetFilter(hCtx, AppName, DbName,
 FilterName, Active, pAccess)
 pAccess = ESB_ACCESS_READ
```

```
 Row = "@IDESCENDANTS(Scenario)"
 sts = EsbSetFilterRow(hCtx, Row, pAccess)
 pAccess = ESB_ACCESS_WRITE
 Row = "@IDESCENDANTS(Scenario), East"
 sts = EsbSetFilterRow(hCtx, Row, pAccess)
 sts = EsbSetFilterRow(hCtx, ByVal 0&, pAccess)
End Sub
```

関連トピック

- EsbGetFilter
- EsbListFilters
- EsbSetFilterRow

EsbSetFilterList

フィルタに割り当てられたグループまたはユーザーのリストを設定します。Items パラメータはフィルタに割り当てられたグループまたはユーザーの数を制御します。Items がゼロの場合、グループまたはユーザーのすべてがリストから削除されます。

構文

```
      EsbSetFilterList
       (
      hCtx, AppName, DbName, FltName, UserList, Items
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FltName
       As String
ByVal
      UserList
       As String
ByVal
      Items
        As Integer
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　データベース名。

パラメータ 説明

FltName　フィルタ名。

UserList　ユーザー名の文字列(CR、EOL で区切られる)の長さは、64KB 未満である必要があります。

Items　文字列内のユーザー名の数。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetFilterList Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String, ByVal FltName As String, ByVal UserList As
String, ByVal Items As Integer) As Long

Sub ESB_SetFilterList ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FilterName As String
 Dim UserList As String
 Dim Items As Integer  AppName = "Sample"
 DbName = "Basic"
 FilterName = "Filter"
 '******************************
 ' Initialize UserList and Items
 '******************************
 Items = 2
 UserList = "Admin"+Chr$(13)+Chr$(10)+"Truc"
 '*****************
 ' Set Filter List
 '*****************
 sts = EsbSetFilterList (hCtx, AppName, DbName,
 FilterName, UserList, Items)
End Sub
```

関連トピック

- EsbGetFilterList
- EsbListFilters
- EsbSetFilter

EsbSetFilterRow

フィルタの次の行を設定します。

構文

```
      EsbSetFilterRow
       (
      hCtx, FltRow, pAccess
      )
ByVal
      hCtx
        As Long
ByVal
      FltRow
       As Any
ByVal
      pAccess
       As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
FltRow	フィルタの次の行へのポインタ。
pAccess	フィルタ行のアクセス・レベル。

備考

この関数は、**EsbSetFilter()**を呼び出した後、行リストが NULL で終了するまで、フィルタの各行に対して 1 回ずつ繰り返し呼び出す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbSetFilterRow Lib "ESBAPIN" (ByVal hCtx As Long, ByVal FltRow
As Any, ByVal pAccess As Integer) As Long
```

EsbSetFilter の例を参照してください。

関連トピック

- EsbListFilters
- EsbSetFilter

EsbSetGlobalState

システム管理用のパラメータが含まれている、サーバーのグローバルな状態構造体を設定します。

構文

```
      EsbSetGlobalState
       (
      hCtx, pGlobal
      )
ByVal
      hCtx
        As Long

      pGlobal
       As ESB_GLOBAL_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pGlobal　　グローバル状態構造体へのポインタ。

備考

パラメータ値を変更するときは、最初に **EsbGetGlobalState()**を呼び出して、変更しないパラメータの正しい値を取得することをお薦めします。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が管理者である必要があります。

例

```
    Declare Function EsbSetGlobalState Lib "ESBAPIN" (ByVal hCtx As Long, Global As
ESB_GLOBAL_T) As Long

Sub ESB_SetGlobalState ()
 Dim sts As Long
 Dim pGlobal As ESB_GLOBAL_T  pGlobal.Security = ESB_TRUE
 pGlobal.Logins = ESB_TRUE
 pGlobal.Access = ESB_ACCESS_READ
 pGlobal.Validity = 14
 pGlobal.Currency = ESB_FALSE
 pGlobal.PwMin = 6
 pGlobal.InactivityTime = 300
 pGlobal.InactivityCheck = 40  '*****************
 ' Set Global State
 '*****************
 sts = EsbSetGlobalState (hCtx, pGlobal)
End Sub
```

関連トピック

- EsbGetGlobalState

EsbSetGroup

グループのセキュリティ情報を含むグループ情報構造体を設定します。

構文

```
      EsbSetGroup
       (
      hCtx, pUserInfo
      )
ByVal
      hCtx
         As Long

      pUserInfo
       As ESB_USERINFO_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pUserInfo　グループ情報構造体へのポインタ。

備考

- 設定するグループ名は、グループ情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更できるグループ情報構造体のフィールドは、Access フィールドのみです(その他のフィールドはユーザーに情報を提供する目的のみに使用されます)。詳細は ESB_GROUPINFO_T 構造体の説明を参照してください。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbSetGroup Lib "ESBAPIN" (ByVal hCtx As Long, GroupInfo As
ESB_USERINFO_T) As Long

Sub ESB_SetGroup ()
 Dim sts As Long
 Dim GroupInfo As ESB_USERINFO_T  '*******************************
 ' Initialize GroupInfo structure
 '*******************************
 GroupInfo.Name = "PowerUsers"
 GroupInfo.Type = ESB_TYPE_GROUP
 GroupInfo.Access = ESB_PRIV_APPCREATE
 GroupInfo.MaxAccess = ESB_PRIV_APPCREATE '************************
```

```
  ' Set GroupInfo structure
  '************************
  sts = EsbSetGroup (hCtx, GroupInfo)
End Sub
```

関連トピック

- EsbGetGroup
- EsbListGroups

EsbSetGroupList

グループのメンバーであるユーザーのリストを設定します。

構文

```
      EsbSetGroupList
       (
      hCtx, GrpName, GrpList, Items
      )
ByVal
      hCtx
        As Long
ByVal
      GrpName
       As String
ByVal
      GrpList
       As String
ByVal
      Items
        As Integer
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
GrpName	グループまたはユーザー名。
GrpList	ユーザー名の文字列(EOL、CR で区切られる)の長さは、64KB 未満である必要があります。
Items	ユーザー名のアイテム。

備考

ユーザーが属するグループのリストを設定するには、ユーザー名を GroupName 引数として入力し、グループのリストを UserList 引数として渡します。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbSetGroupList Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
GroupName As String, ByVal UserList As String, ByVal Items As Integer) As Long

Sub ESB_SetGroupList ()
 Dim sts As Long
 Dim GroupName As String
 Dim UserList As String
 Dim Items As Integer
 Dim CRLF As String
 CRLF = Chr$(13) + Chr$(10)
 GroupName = "PowerUsers"  '*****************************
 ' Initialize UserList and Items
 '*****************************
 Items = 2
 UserList = "Admin" + CRLF + "Bob" '***************
 ' Set Group List
 '***************
 sts = EsbSetGroupList (hCtx, GroupName, UserList, Items)
End Sub
```

関連トピック

- EsbAddToGroup
- EsbDeleteFromGroup
- EsbListGroups
- EsbSetGroupList

EsbSetPassword

既存のパスワードを消去して、ユーザーのパスワードを設定します。

構文

```
      EsbSetPassword
       (
      hCtx, userName, Password
      )
ByVal
      hCtx
        As Long
ByVal
      userName
       As String
ByVal
      Password
       As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

userName　ユーザー名。

Password　ユーザーの新パスワード。

備考

- パスワードを変更するには、呼出し元は、管理者アクセス権を持っているか、または変更するパスワードが呼出し元のものである必要があります。
- 新パスワードは、ユーザーが次回ログインする際から有効になります。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元が呼出し元のパスワードを設定していないかぎり、ログインしたサーバーに対して呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbSetPassword Lib "ESBAPIN" (ByVal hCtx As Long, ByVal UserName
As String, ByVal Password As String) As Long

Sub ESB_SetPassword ()
 Dim sts As long
 Dim UserName As String
 Dim Password As String  UserName = "Joseph"
 Password = "NewPassword"
 '*****************
 ' Set New password
 '*****************
 sts = EsbSetPassword (hCtx, UserName, Password)
End Sub
```

関連トピック

- EsbListUsers
- EsbSetUser

EsbSetPath

実行時プロセスのための ESSBASEPATH 環境変数を設定します。

構文

```
EsbSetPath
(
Path
)
```

```
ByVal
      Path
       As String
```

パラメータ 説明

Path　　ESSBASEPATH 環境変数を示す文字列

備考

- EsbSetPath()を呼び出してから EsbInit()を呼び出してください。
- ESB_PATHLEN で定義されているように、Path は 120 文字以内で指定します。
- Path は現在のプロセスにのみ適用されます。
- Essbase DLL はシステム・パスからアクセスできる必要があります。**EsbSetPath()** は Essbase DLL のパスを解決しません。

戻り値

- 正常終了の場合は、ESB_STS_NOERR が戻されます。
- Path が長すぎる場合は、API_NAME_TOO_LONG(1030009)が戻されます。

例

```
    Sub ESB_SetPath
 Dim sts  As Long
 Dim Path As String

 Path = "C:\Hyperion\products\Essbase"
 sts  = EsbSetPath(Path)
End Sub
```

EsbSetSpanRelationalSource

Essbase に関係するデータが接続したリレーショナル・ストアに存在することを通知する、ブール bSpanRelPart フィールドを設定します。EsbQueryDatabaseMembers などのその他の API 関数の一部は、bSpanRelPart を読み込み、bSpanRelPart が設定されている場合はリレーショナル・ストアにアクセスします。

構文

```
     Declare Function EsbSetSpanRelationalSource Lib "esbapin" (ByVal hCtx As Long)
As Long
```

パラメータ 説明

hCtx　　API コンテキスト・ハンドル。

備考

一部の API 関数は、リレーショナル・ストアから情報を取得するように拡張されています。

- EsbQueryDatabaseMembers- リレーショナル・ストアからメンバー名を戻します。
- EsbGetMemberInfo - リレーショナル・ストア内のメンバーに関する情報を戻します。
- EsbCheckMemberName - リレーショナル・ストアで有効なメンバー名を確認します。
- EsbGetMemberCalc - 入力として渡されたリレーショナル・メンバーを認識し、すべてのリレーショナル・メンバーに対して NULL 文字列を戻します。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベース内の 1 つ以上のメンバーに対して、呼出し元が読取り権限(ESS_PRIV_READ)を持っている必要があります。

例

```
     Declare Function EsbSetRelationalSource Lib "ESBAPIN" (ByVal hCtx As Long) As
Long

 Sub ESB_SetRelationalSource ()
   Dim sts As Long

   '************************
   ' Set the bSpanRelPart  field
   '************************
   sts = EsbSetRelationalSource (hCtx)

 End Sub
```

関連トピック

- EsbClrSpanRelationalSource

EsbSetUser

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。

構文

```
      EsbSetUser
       (
      hCtx, pUserInfo
      )
ByVal
```

```
    hCtx
       As Long

    pUserInfo
     As ESB_USERINFO_T
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

pUserInfo　ユーザー情報構造体へのポインタ。

備考

- 設定するユーザー名は、ユーザー情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更できるユーザー情報構造体のフィールドは、Access、Expiration、PwdChgNow フィールドです(その他のフィールドは情報を提供する目的のみに使用されます)。詳細は 1296 ページの「ESB_USERINFO_T, ESB_GROUPINFO_T」を参照してください。
- 呼出し元は、指定したユーザーに対して、呼出し元ユーザーが所有していないアクセス権限は付与できません。
- 新規のユーザー設定は、次にユーザーがログインする際に有効になります。

戻り値

なし。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESB_PRIV_USERCREATE)を持っている必要があります。

例

```
    Declare Function EsbSetUser Lib "ESBAPIN" (ByVal hCtx As Long, UserInfo As
ESB_USERINFO_T) As Long

Sub ESB_SetUser ()
 Dim sts As long
 Dim UserInfo As ESB_USERINFO_T  '*******************************
 ' Initialize fields for UserInfo
 '*******************************
 UserInfo.Name = "Joseph"
 UserInfo.Type = ESB_TYPE_USER
 UserInfo.Access = ESB_ACCESS_SUPER
 UserInfo.MaxAccess = ESB_ACCESS_SUPER
 '************************
 ' Set User Info structure
 '************************
 sts = EsbSetUser (hCtx, UserInfo)
End Sub
```

関連トピック

- EsbGetUser
- EsbListUsers
- EsbSetApplicationAccess
- EsbSetPassword

EsbSetUserEx

ユーザーのセキュリティ情報が含まれているユーザー情報構造体を設定します。

構文

```
Declare Function EsbSetUserEx Lib "esbapin" (ByVal hCtx As Long, pUserInfo As
ESB_USERINFOEX_T) As Long
```

パラメータ　説明

hCtx　　API コンテキスト・ハンドル。

pUserInfoEx 外部認証ユーザーの情報構造体へのポインタ。

備考

- この関数は EsbSetUser に似た動作をします。違いはこの関数は拡張ユーザー情報構造体である ESB_USERINFOEX_T を設定するという点です。
- 設定するユーザー名は、ユーザー情報構造体のフィールドであり、必ず指定する必要があります。
- この関数を使用して変更できるユーザー構造体内のフィールドは、「アクセス」、「有効期限」、および「PwdChgNow」フィールドのみです(他のフィールドは情報提供専用)。詳細は 1296 ページの「ESB_USERINFO_T, ESB_GROUPINFO_T」を参照してください。
- 呼出し元は、指定したユーザーに対して、呼出し元ユーザーが所有していないアクセス権限は付与できません。
- 新規のユーザー設定は、次にユーザーがログインする際に有効になります。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

この関数を使用するには、ログインしたサーバーに対して、呼出し元がユーザーの作成/削除権限(ESS_PRIV_USERCREATE)を持っている必要があります。

EsbShutdownServer

エージェントを停止します。

構文

```
      EsbShutdownServer
       (
      hInst, Server, User, Password
      )
ByVal
      hInst
        As Long
ByVal
      Server
       As String
ByVal
      User
        As String
ByVal
      Password
       As String
```

パラメータ 説明

hInst	VB API インスタンス・ハンドル。
Server	ネットワーク・サーバー名の文字列。シャットダウンするサーバーの名前を指定します。
User	ユーザー名の文字列。シャットダウンを要求しているユーザーを指定します。
Password	パスワード文字列。シャットダウンを要求しているユーザーのパスワードを指定します。

備考

- この関数はエージェント(ESSBASE.EXE)に対して、この関数をシャットダウンするように要求を送信します。エージェントは、データのコミット、すべてのアプリケーションとデータベースの終了、ユーザーのログオフ後の停止など、通常のシャットダウン手順を実行します。
- 管理者権限を持っているユーザーのみが、エージェントをシャットダウンできます。
- この関数はいつでも呼び出せますが、通常はバックグラウンドで起動されたエージェントのシャットダウンのために呼び出します。詳細は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

なし。

この関数の結果として考えられるエラー条件には、次のものがあります:

- AD_AMSG_IPO は、この操作のための権限が不足していることを示します
- AD_AMSG_IPW は、パスワードが正しくないことを示します
- AD_AMSG_UNE は、ユーザーが存在しないことを示します
- AD_MSGAR_NOSHUTDOWN は、アプリケーションをシャットダウンできないことを示します

- ネットワーク・エラー: NET_TCP_HOSTS は、ホスト・ファイルで検索できないことを示します
- ネットワーク・エラー: NET_NP_NOSERVER は、サーバーを検索できないことを示します

アクセス

この関数を使用するには、管理者権限を持っている必要があります。

例

```
    Declare Function EsbShutdownServer Lib "ESBAPIN" (ByVal hInst As Long, ByVal
Server As String, ByVal User As String, ByVal Password As String) As Long

Sub ESB_ShutdownServer()
 Dim sts As Long
 Dim Server As String
 Dim UserName As String
 Dim Password As String  Server = "Rainbow"
 UserName = "Admin"
 Password = "password"  '******************
 ' Shut down Server
 '******************
 sts = EsbShutdownServer(hInst, Server, UserName, Password)
End Sub
```

関連トピック

- EsbSetPassword
- EsbUnloadApplication
- EsbUnloadDatabase

EsbTerm

VB API を終了して、VB API で使用しているすべてのシステム・リソースをリリースします。

構文

```
      EsbTerm
       (
      hInst
      )
ByVal
      hInst
       As Long
```

パラメータ 説明

hInst　　VB API インスタンス・ハンドル。

備考

- この関数は、通常は他のすべての VB API 呼出しが完了した後、つまり使用しているプログラムを終了する直前に呼び出す必要があります。
- この関数によって VB API の使用は終了するため、この関数の実行後に(**EsbInit()** 以外の)VB API 関数を呼び出すと、エラーが戻されます。

戻り値

なし。

アクセス

この関数には、特別なアクセス権は必要ありません。

例

```
    Declare Function EsbTerm Lib "ESBAPIN" (ByVal hInst As Long) As Long
Sub ESB_Term ()
 Dim sts As Long  '*****************************
 ' Terminate the VB API
 '*****************************
 sts = EsbTerm (hInst)
End Sub
```

関連トピック

- EsbInit

EsbUnloadApplication

サーバー上のアプリケーションを停止します。

構文

```
     EsbUnloadApplication
      (
     hCtx, AppName
     )
ByVal
     hCtx
       As Long
ByVal
     AppName
      As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アンロードするアプリケーション名。

備考

- アプリケーションをアンロードするには、接続しているユーザーがアプリケーションに対してロード権限を持っている必要があります。
- アプリケーションに関連付けられているデータベースが Essbase サーバーによって再構築されている場合は、そのアプリケーションをアンロードできません。

戻り値

なし。

アクセス

この関数を使用するには、指定したアプリケーションに対して、呼出し元がアプリケーションのロード/アンロード権限(ESB_PRIV_APPLOAD)を所有している必要があります。

例

```
    Declare Function EsbUnloadApplication Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String) As Long

Sub ESB_UnloadApplication ()
 Dim sts As Long
 Dim AppName As String  AppName = "Sample"  '*****************
 ' Unload Application
 '*****************
 sts = EsbUnloadApplication (hCtx, AppName)
End Sub
```

関連トピック

- EsbLoadApplication
- EsbUnloadDatabase

EsbUnloadDatabase

サーバー上でアプリケーション内のデータベースを停止します。

構文

```
      EsbUnloadDatabase
       (
      hCtx, AppName, DbName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
```

```
      As String
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

AppName　　アプリケーション名。

DbName　　アンロードするデータベース名。

戻り値

なし。

アクセス

この関数を使用するには、呼出し元がデータベースのロード/アンロード権限(ESB_PRIV_APPLOAD)を持っている必要があります。

例

```
    Declare Function EsbUnloadDatabase Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
AppName As String, ByVal DbName As String) As Long

Sub ESB_UnloadDatabase ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String  AppName = "Sample"
 DbName = "Basic"  '****************
 ' Unload Database
 '****************
 sts = EsbUnloadDatabase (hCtx, AppName, DbName)
End Sub
```

関連トピック

- EsbLoadDatabase

EsbUnlockObject

サーバーまたはクライアント・オブジェクト・システム上でロックされているオブジェクトをロック解除します。

構文

```
      EsbUnlockObject
       (
      hCtx, ObjType, AppName, DbName, ObjName
      )
ByVal
      hCtx
        As Long
ByVal
      ObjType
       As Long
```

```
ByVal
      AppName
       As String
ByVal
      DbName
       As String
ByVal
      objName
       As String
```

パラメータ	説明
hCtx	VB API コンテキスト・ハンドル。EsbCreateLocalContext()によって戻されたローカル・コンテキスト・ハンドルの場合もあります。
ObjType	オブジェクト・タイプ(単一のタイプのみ)。使用可能な値のリストは、表 15 を参照してください。
AppName	アプリケーション名。
DbName	データベース名。空の文字列の場合は、アプリケーションのサブディレクトリが使用されます。
ObjName	ロック解除されるオブジェクト名。

備考

オブジェクトをロック解除するには、そのオブジェクトが存在し、呼出し元によってロックされている必要があります。

戻り値

なし。

アクセス

この関数を使用するには、オブジェクトが含まれている指定されたアプリケーションまたはデータベースに対して、呼出し元がアプリケーション・デザイン権限またはデータベース・デザイン権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbUnlockObject Lib "ESBAPIN" (ByVal hCtx As Long, ByVal ObjType
As Integer, ByVal AppName As String, ByVal DbName As String, ByVal ObjName As String)
As Long

Sub ESB_UnlockObject ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim ObjName As String
 Dim ObjType As Integer  AppName = "Sample"
 DbName = "Basic"
 ObjName = "Basic"
 ObjType = ESB_OBJTYPE_OUTLINE  '******************
 ' UnLock Rules Object
```

```
 '******************
 sts = EsbUnlockObject (hCtx, ObjType, AppName,
    DbName, ObjName)
End Sub
```

関連トピック

- EsbGetObject
- EsbGetObjectInfo
- EsbListObjects
- EsbLockObject
- EsbPutObject

EsbUpdate

アクティブなデータベースに更新指定を単一文字列として送信します。

構文

```
      EsbUpdate
       (
      hCtx, isStore, isUnlock, updQuery
      )
ByVal
      hCtx
        As Long
ByVal
      isStore
       As Integer
ByVal
      isUnlock
       As Integer
ByVal
      updQuery
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
isStore	データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。
isUnlock	データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。
updQuery	単一文字列としての更新指定(32KB 未満である必要があります)。

備考

- この関数は、**EsbBeginUpdate()**を呼び出し、その後 **EsbSendString()**を呼び出し、最後に **EsbEndUpdate()**を呼び出すのと同じです。更新されたデータをデータベースに保管することも、確認のみ行ってエラーがあれば戻させることもで

きます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

- 更新指定文字列の長さは 32KB 未満にする必要があります。
- この関数によってデータが保管される場合(Store フラグが TRUE)は、関連データ・ブロックが更新のためにロックされている必要があります(たとえば、Lock フラグを TRUE に設定して **EsbReport**()を呼び出します)。
- 呼出し元がメンバーにデータを書き込もうとした場合に、書込み権限がないと、警告が生成され、メンバーは更新されません。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベースに対して、呼出し元が書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbUpdate Lib "ESBAPIN" (ByVal hCtx As Long, ByVal Store As
Integer, ByVal Unlock As Integer, ByVal Query As String) As Long

Sub ESB_Update ()
 Dim sts As Long
 Dim Store As Integer
 Dim pUnlock As Integer
 Dim Query As String  Query = "Year Market Scenario AcItemss Product 12345"  Store =
ESB_YES
 pUnlock = ESB_NO  '*******
 ' Update
 '*******
 sts = EsbUpdate (hCtx, Store, pUnlock, Query)
End Sub
```

関連トピック

- EsbBeginUpdate
- EsbEndUpdate
- EsbReport
- EsbSendString
- EsbUpdateFile

EsbUpdateDrillThruURL

アクティブなデータベース・アウトライン内で、指定された名前のドリルスルー URL を更新します。

1904 ページの「ドリルスルー URL の制限」.

構文

```
     Declare Function EsbUpdateDrillThruURL Lib "esbapin" (ByVal hCtx As Long, ByRef
symRegions() As String, ByRef pUrl As ESB_DURLINFO_T, ByVal bMerge As Integer) As
Long
```

パラメータ	説明
hCtx	Visual Basic API のコンテキスト・ハンドル
symRegions()	対称領域の指定を含む配列
pUrl	URL 定義
bMerge	● TRUE の場合、pUrl 内のドリルスルー領域定義を、指定された URL 定義内にある既存のドリルスルー領域のリストに追加します ● FALSE の場合、既存のドリルスルー領域定義のリストを、pUrl 内のリストで置き換えます

戻り値

- 正常に処理されると、URL XML の置換と、pUrl 内の対応するフィールドによるドリルスルー領域のリストの更新または置換によって、アクティブなデータベース内の指定されたドリルスルー URL が更新されます。
- 指定された名前の URL が存在しない場合は、エラー・コードが戻されます。

アクセス

- 呼出し側は、指定したデータベースに対してデータベース設計権限(ESB_PRIV_DBDESIGN)を持っている必要があります。
- 呼出し側は EsbSetActive()を使用して、指定したデータベースを呼出し側のアクティブなデータベースとして選択しておく必要があります。

例

```
   Sub ESB_UpdateGLDrillThru()
 Dim sts           As Long
 Dim url           As ESB_DURLINFO_T
 Dim cppDrillRegions(0 To 1)  As String
 Dim bMerge          As Integer

 '*************************************************************
 ' Need to create a local context, if files are not on the server
 '*************************************************************
 url.bIsLevel0 = 0
 bMerge = ESB_TRUE

 cppDrillRegions(0) = "qtr1"
 url.cpURLXML = "<?xml version="1.0" encoding="UTF-8"?>
<foldercontents path="/">
 <resource name="Assets Drill through GL" description="" type="application/x-hyperion-
applicationbuilder-report">
  <name xml:lang="fr">Rapport de ventes</name>
  <name xml:lang="es">Informe de ventas</name>
  <action name="Display HTML" description="Launch HTML display of Content"
```

```
shortdesc="HTML">
   <url>/fusionapp/Assetsdrill.jsp?$SSO_TOKEN$&$CONTEXT$&$ATTR(ds,pos,gen,level.edge)
$
   </url>
  </action>
 </resource>
</foldercontents>"
 url.cpURLName = "VB URL7"
 url.iURLXMLSize = 512

 sts = EsbUpdateDrillThruURL(hCtx, cppDrillRegions, url, bMerge)

 Debug.Print "EsbUpdateDrillThruURL sts: " & sts
End Sub
```

1232 ページの「ドリルスルー Visual Basic API の例」に記載されている拡張の例も参照してください。

EsbUpdateFile

ファイルからアクティブ・データベースに対して更新指定を送信します。更新データはデータベースに保管することも、確認のみ行ってエラーがあれば戻すこともできます。また、この呼出しによって、更新用にロックされていたデータ・ブロックもロック解除できます。

構文

```
      EsbUpdateFile
       (
      hDestCtx, hSrcCtx, AppName, DbName, FileName, isStore, isUnlock
      )
ByVal
      hDestCtx
       As Long
ByVal
      hSrcCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
        As String
ByVal
      FileName
       As String
ByVal
      isStore
       As Integer
ByVal
      isUnlock
       As Integer
```

パラメータ 説明

hDestCtx　サーバー上のターゲット・データベースの VB API コンテキスト・ハンドル。

hSrcCtx　更新ファイルの場所に対する VB API コンテキスト・ハンドル。レポート・ファイルは、クライアント、またはターゲット・データベースと同一のサーバー上に配置できます。

AppName　更新ファイルの場所のアプリケーション名。

DbName　更新ファイルの場所のデータベース名。

FileName　更新指定ファイル名。

isStore　データの保管を制御します。TRUE の場合は、データがサーバーに保管されます。FALSE の場合はデータは保管されません。

isUnlock　データ・ブロックのロック解除を制御します。TRUE の場合、ロックされているすべての関連ブロックのロックが解除されます(必要に応じてデータの保管後)。FALSE の場合、ブロックのロックは解除されません。

備考

- この関数によってデータが保管される場合(Store フラグが TRUE)は、関連データ・ブロックが更新のためにロックされている必要があります(たとえば、Lock フラグを TRUE に設定して **EsbReport()**を呼び出します)。
- Store および Unlock の両方のフラグが FALSE に設定されている場合、データベースは更新指定の構文確認のみを行います。

戻り値

なし。

アクセス

この関数を使用するには、アクティブなデータベースに対して、呼出し元が書込み権限(ESB_PRIV_WRITE)を持っている必要があります。

例

```
    Declare Function EsbUpdateFile Lib "ESBAPIN" (ByVal hDestCtx As Long, ByVal
hSrcCtx As Long, ByVal AppName As String, ByVal DbName As String, ByVal FileName As
String, ByVal Store As Integer, ByVal Unlock As Integer) As Long

Sub ESB_UpdateFile ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim FileName As String
 Dim Store As Integer
 Dim pUnlock As Integer
 Dim hSrcCtx As Long  AppName = "Sample"
 DbName = "Basic"  '************************************
 ' Update file is an object at the server *
 '************************************
 hSrcCtx = hCtx
 FileName = "update"   Store = ESB_YES
 pUnlock = ESB_NO  '************
 ' Update File
```

```
 '*************
 sts = EsbUpdateFile (hCtx, hSrcCtx, AppName,
    DbName, FileName, Store, pUnlock)
End Sub
```

関連トピック

- EsbBeginUpdate
- EsbReportFile
- EsbUpdate

EsbValidateDB

データベースの整合性を検証します。

構文

```
      EsbValidateDB
       (
      hCtx, DbName, FileName
      )
ByVal
      hCtx
        As Long
ByVal
      DbName
       As String
ByVal
      FileName
      As String
```

パラメータ 説明

hCtx	API コンテキスト・ハンドル。
DbName	データベース名。必須で、NULL は指定できません。
FileName	エラー・ログ・ファイル。サーバー上の app¥db に保存されます。必須。

備考

- この関数は、検証チェックを実行してデータベースの整合性を検証します。
- このコマンドによって現在のデータベースが検証されます。**EsbValidateDB()**コマンドの発行前に、データベースを選択する必要があります。
- **EsbValidateDB()**は、各ブロックのデータの整合性を確認します。最上位から最下位まで読み取り、検証プロセスでデータベース全体を検証し、ブロック、セクション、ブロック・タイプ、ブロック長および浮動小数点数の有効性を確認します。
- このコマンドによってブロックと不正ブロックに関する情報がログ・ファイルに書き込まれます。

- このコマンドによって整合性エラーが検出されると、検証プロセス・エラー・メッセージがテキスト・フォーマットのログ・ファイルに書き込まれます。ファイルのデフォルト場所は、たとえば ESSBASE\APP\DB\VALIDATE.LST のような application\database ディレクトリです
- この呼出しの前に、**EsbSetActive()**を呼び出してください。
- この関数は非同期であるため、検証プロセスが終了するまで**EsbGetProcessState()**を呼び出し続ける必要があります。
- インデックスには各データ・ブロックのインデックスが含まれています。すべての読取り操作に関して、このコマンドは自動的にインデックス・ページ内のインデックス・キーをそれに対応するデータ・ブロックのインデックス・キーと比較し、ブロックのその他のヘッダー情報を確認します。不一致がある場合、**EsbValidateDB()**はエラー・メッセージを表示し、データベース全体を確認するまで、処理を続行します。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・マネージャ権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbValidateDB Lib "ESBAPIN" (ByVal hCtx As Long, ByVal DbName As
String, ByVal FileName As String) As Long

Sub ESB_ValidateDB()
 Dim sts As Long
 Dim DbName As String
 Dim FileName As String
 Dim ProcState As ESB_PROCSTATE_T  DbName = "Basic"
 FileName =
 "D:\Essbase\App\Sample\Basic\Validate.lst"  sts = EsbValidateDB(hCtx, DbName,
FileName)
 If Not sts Then
  '************************************
  'Check process state until it is done
  '************************************
  sts = EsbGetProcessState(hCtx, ProcState)
  Do While sts = 0 And ProcState.State =
     ESB_STATE_INPROGRESS
   sts = EsbGetProcessState(hCtx, ProcState)
  Loop
 End If
End Sub
```

関連トピック

- EsbSetActive
- EsbGetProcessState

EsbValidateHCtx

特定のコンテキスト・ハンドル(hCtx)を検証します。

構文

```
EssValidateHCtx
 (
hCtx
 )
```

パラメータ　データ型　　説明

hCtx	ESB_HCTX_T	検証する API コンテキスト・ハンドル

備考

この関数を待機期間延長後に使用すると、プログラムのコンテキスト・ハンドルがサーバーによって認識される状態を確保できます。

戻り値

この関数はコンテキスト・ハンドルが有効な場合は 0 を戻し、それ以外の場合は無効なコンテキスト・ハンドルを示すエラー・コードを戻します。無効なコンテキスト・ハンドルに対して考えられる理由には、ログインがタイムアウトした、またはユーザーが管理者によって明示的にログアウトされたなどがあります。

アクセス

この関数には、特別なアクセス権は必要ありません。

例

```
        Dim sts As Long
    Dim Count As Integer
    Dim pAccess As Integer  sts = EsbLogin(hInst, "localhost", "test", "testing",
Count, hCtx)
    sts = EsbSetActive(hCtx, "sample", "Basic", pAccess)

    ' Do something else not related to Essbase Server

    sts = EsbValidateHCtx(hCtx)
    If (sts <> 0) Then
      'if Context no longer valid, re-login
      sts = EsbLogin(hInst, "localhost", "test", "testing", Count, hCtx)
      sts = EsbSetActive(hCtx, "Sample", "Basic", pAccess)
    End If

    'Proceed
```

関連トピック

- EsbLogin
- EsbAutoLogin
- EsbTerm

EsbVerifyFilter

指定したデータベースに照らしあわせて、一連のフィルタ行の文字列の構文を確認します。

構文

```
      EsbVerifyFilter
       (
      hCtx, AppName, DbName
      )
ByVal
      hCtx
        As Long
ByVal
      AppName
       As String
ByVal
      DbName
       As String
```

パラメータ 説明

hCtx	VB API コンテキスト・ハンドル。
AppName	アプリケーション名。
DbName	データベース名。

備考

この呼出しの後に続いて **EsbVerifyFilterRow()**を呼び出し、フィルタのすべての行を確認する必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・デザイン権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
   Declare Function EsbVerifyFilter Lib "ESBAPIN" (ByVal hCtx As Long, ByVal AppName
As String, ByVal DbName As String) As Long

Sub ESB_VerifyFilter ()
 Dim sts As Long
 Dim AppName As String
 Dim DbName As String
 Dim Row As String AppName = "Sample"
 DbName = "Basic"
 '***************
 ' Verify Filter
```

```
  '***************
  sts = EsbVerifyFilter(hCtx, AppName, DbName) ' Initialize Filter Row
  Row = "@IDESCENDANTS(Scenario)"
  sts = EsbVerifyFilterRow(hCtx, Row) ' Initialize Filter Row
  Row = "@IDESCENDANTS(Product)"
  sts = EsbVerifyFilterRow(hCtx, Row) sts = EsbVerifyFilterRow(hCtx,
     ByVal 0&)
End Sub
```

関連トピック

- EsbGetFilter
- EsbVerifyFilterRow

EsbVerifyFilterRow

指定したデータベースに照らしあわせて、単一のフィルタ行の文字列の構文を確認します。

構文

```
      EsbVerifyFilterRow
       (
      hCtx, FltRow
      )
ByVal
      hCtx
        As Long
ByVal
      FltRow
       As Any
```

パラメータ 説明

hCtx　　VB API コンテキスト・ハンドル。

FltRow　　フィルタ行文字列。

備考

EsbVerifyFilter()を呼び出した後、行リストが NULL で終了するまで、フィルタの各行に対して 1 回ずつ、この関数を繰り返し呼び出す必要があります。

戻り値

なし。

アクセス

この関数を使用するには、指定したデータベースに対して、呼出し元がデータベース・マネージャ権限(ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbVerifyFilterRow Lib "ESBAPIN" (ByVal hCtx As Long, ByVal
FltRow As Any) As Long
```

EsbVerifyFilter の例を参照してください。

関連トピック

- EsbGetFilter
- EsbVerifyFilter

EsbWriteToLogFile

Essbase サーバー・ログ・ファイル(essbase.log)またはアプリケーション・ログ・ファイル(appname.log)にメッセージを書き込みます。

構文

```
     Declare Function EsbWriteToLogFile Lib "esbapin" (
ByVal hCtx As Long,
ByVal isAgentLog As Boolean,
ByVal Message As String) As Long
```

パラメータ	説明
hCtx	API コンテキスト・ハンドル。
isAgentLog	TRUE の場合、メッセージは Essbase サーバー・ログ・ファイル essbase.log に書き込まれます。FALSE の場合、メッセージはアプリケーション・ログ・ファイル appname.log に書き込まれます。
Message	Essbase サーバー・ログ・ファイル(essbase.log)またはアプリケーション・ログ・ファイル(appname.log)に記録されるメッセージ。

備考

- メッセージ・ログを表示するには、**EsbGetLogFile()**を使用します。
- essbase.log および appname.log の場所は、『Oracle Essbase データベース管理者ガイド』を参照してください。

戻り値

正常終了の場合は 0 が戻されます。

アクセス

呼出し元は、指定したアプリケーションに対して管理者権限(ESS_ACCESS_SUPER)を持っている必要があります。

第 V I 部

Visual BasicのアウトラインAPI

Visual Basic のアウトライン API の内容：

21 Visual Basicのアウトライン APIの使用

この章の内容

Visual Basic のアウトライン API について

アウトライン API は Essbase アウトラインをカスタム・アプリケーション内から作成、維持、操作する一連の関数です。アウトライン API を使用して、Administration Services のアウトライン・エディタを使用する場合と同様にしてコード内からデータベース・アウトラインを操作できます。

アウトライン API は Essbase API の重要な部分で、C および Visual Basic のインタフェースを備えています。アウトライン API は Essbase API とともに使用され、サーバー接続を必要とします。

Visual Basic のアウトライン API エラー処理

アウトライン API 関数が正常終了の場合は 0 が戻されます。失敗すると C の場合は `esserror.h` で定義されたエラー・ステータスの値が、Visual Basic の場合は `esberror.bas` で定義されたエラー・ステータスの値が戻されます。メイン API の関数はエラー・メッセージ・コールバック・ルーチンを使用して、メッセージ・ハンドラにエラー番号を戻します。ハンドラは `essbase.mdb` メッセージ・データベースを使用してエラー・メッセージを特定し、ユーザーにエラー・メッセージを表示します。

アウトライン API 関数は通常、エラー・ステータスを戻す際にエラー・メッセージ・コールバック・ルーチンを使用しません。エラー・コールバック・ルーチンは次の場合に呼び出されます:

- ネットワークを使用する関数(**EsxOtlOpenOutline**()、**EsxOtlWriteOutline**()および **EsxOtlRestructure**())を呼び出して、アウトラインに関係のないアクションでエラーが発生した場合。

- アウトライン API に渡されたときルーチン確認で NULL が検出され、API_NULL_ARG が戻された場合。
- 不良なアウトライン・ハンドル(HOUTLINE)が、アウトライン・ハンドルを必要とする呼出しに渡され、OTLAPI_BAD_HOUTLINE が戻された場合。

Visual Basic サーバー・アウトライン・クエリー

いくつかの関数はアウトライン API へのクエリー・インタフェースをサポートしているため、アウトラインをサーバーからダウンロードしてメモリーに完全に読み込む必要がありません。この種のアウトライン API 関数はサーバー・アウトラインのみをサポートしています。アウトラインを開く前に、ユーザーはサーバーにログインして、有効な Essbase ログイン・コンテキストを設定する必要があります。

これらの関数のエラー処理は、標準的な API エラー処理メカニズムで行われます。したがって、エラーの際は、呼出し元が **EsxInit()**から指定したメッセージ・コールバックが呼び出されます。

このメカニズムを次に説明します:

1. プログラマは **EsxInit()**と **EsxLogin()**を呼び出して、通常どおり API を初期化します。
2. プログラマは **EsxOtlOpenOutlineQuery()**を呼び出してサーバーからアウトラインを開き、ある程度の初期情報を取得します。その際、**ESX_OUTLINEINFO_T** 構造体内のすべての情報と、**ESX_MBRINFO_T** 構造体を含む各次元における **ESX_OTLMBR_T** 内部構造体内のすべての関連情報が、サーバーから取得されます。
3. 呼出し元はメンバーに関する情報を入手する必要があるため、該当するフラグを設定して **EsxOtlQueryMembers()**を呼び出し、メンバー・ハンドルの配列を取り戻します。**EsxOtlQueryMembers()**の呼出しによって内部構造体である **ESX_OTLMBR_T** に関連するすべての情報が戻されます。その結果、ユーザーは戻されたメンバー・ハンドルのいずれかを渡して、特定のメンバーに関連するいずれかの **EsxOtlGetXxxx()**を呼び出すことができます。アウトラインをクエリー・モードで開いている場合にサポートされている呼出しの詳細は、**EsxOtlQueryMembers()**呼出しのコメント部分を参照してください。
4. **EsxOtlQueryMembers()**からデータが戻され、呼出しが完了したら、**EssOtlFreeMembers()**または **EsbOtlFreeMember()**を呼び出してメンバーの配列を解放します。
5. 完了したら、呼出し元は **EsxOtlCloseOutline()**を呼び出して、内部のデータ構造体をクリーン・アップします。
6. 呼出し元は **EsxLogout()**と **EsxTerm()**を呼び出して、通常どおり API を終了します。

Visual Basic のアウトライン API のアウトライン確認

アウトライン API によって、呼出し元が不正なアウトラインを作成するのを防止できます。アウトラインを確認するには、**EsxOtlVerifyOutline()**関数を使用して、サーバーに保存する前に確認します。アウトライン API は、アウトラインがサーバーに書き込まれる際、**EsxOtlVerifyOutline()**がまだ呼び出されていない場合には、自動的に呼び出します。

アウトライン API の各関数は、呼出し元による処理によって不正なアウトラインが生成されないか検証します。たとえば **EsxOtlRenameMember()**は新しいメンバー名を確認して、有効で、アウトラインに既存のものでないことを確認します。この種の自動検証にはいくつかの例外があります:

- **EsxOtlOpenOutline()**では、呼出し元が前に作成された不正なアウトラインを読み取ることができます。アプリケーション・マネージャのアウトライン・エディタで不正なアウトラインをローカル・ファイルに保存できるため、このアウトラインは不正な状態です。**EsxOtlVerifyOutline()**を呼び出すと、既存のエラーが検出されます。また、アウトラインが不正なものとして起動した場合は、処理中に個別の操作が不正になります。
- **EsxOtlDeleteMember()**および **EsxOtlDeleteDimension()**は、削除されたメンバーを含む別名の組合せを確認しません。この状態は **EsxOtlVerifyOutline()**によって検出されます。
- **EsxOtlSetMemberFormula()**では不正な式を入力でき、**EsxOtlVerifyOutline()**ではメンバー式を確認しません。不正なメンバー式があると、再構築中に障害が発生します。**EsxGetProcessState()**は、サーバーから戻されたエラー・メッセージを表示します。

Visual Basic のアウトライン API のメモリー割当て

Essbase API では、**EsxAlloc()**、**EsxRealloc()**、**EsxFree()**という、一連のメモリー管理関数を備えています。これらの関数とすべての内部 API メモリー割当ては、**ESX_INIT_T** 初期化構造体の `AllocFunc`、`ReallocFunc`、および `FreeFunc` フィールドが指すメモリー割当てルーチンを呼び出します。

ユーザー独自のメモリー割当てルーチンを使用している場合、多数の小規模なメモリー・バッファの割当てを処理できるようなメモリー割当ての仕組みを使用していることを確認してください。

Visual Basic のアウトライン API のセキュリティ要件

アウトライン API を使用して、アウトラインの作成、編集および削除ができるので、アウトライン API を使用するアプリケーションを作成するときは、セキュリ

ティの問題に注意する必要があります。これらの問題は、セッションの間にアウトラインを作成、編集または保存するプログラムにのみ影響します。

アプリケーション・マネージャのアウトライン・エディタによってアウトラインを操作するには、アプリケーション・デザイナ以上の権限を持っている必要があります。また、実行の間にアウトライン API を使用するプログラムを使用するためにも、これらの権限を必要とします。これらの権限を持っていない場合、サーバーからアウトラインを読取りまたは書込みするアウトライン API 呼出しは機能しません。セキュリティと権限のレベルに関する詳細情報は、『Oracle Essbase データベース管理者ガイド』を参照してください。

たとえば、ユーザーがセッションの間に複数の「仮定」状況を調査できる、新しい EIS エンドユーザー・アプリケーションを書いているとします。これを行うために、プログラムは、セッションの間に複数の Essbase データベースを動的に作成します。これらのデータベース(およびそのアウトライン)は、一時的なものであり、セッションが終了した後に保存されません。この状況にアプローチできる方法はいくつかあります:

- セッションの間にアプリケーションおよび複数のデータベースをユーザーが作成できるようにする場合、ユーザーに**アプリケーションの作成/削除**権限を与えます。プログラムを実行する前に、Essbase スーパーバイザが、この権限を割り当てる必要があります。これは Essbase では比較的高い権限レベルです。しかし、ユーザーがアプリケーション・マネージャなど、他のプログラムにアクセスできない場合、システム・セキュリティ全体への影響はほとんどありません。
- 同時に利用可能な複数のデータベースを必要としない場合、プログラムのインストールの間に、Essbase スーパーバイザに一時的なアプリケーションおよびデータベースを作成してもらうことができます。プログラム自身は、一時的なデータベースを操作するので、各「what-if」状況に新しいデータベースを作成する必要がありません。

2 番目の方法では、ユーザーは、より低く制限された**データベース・デザイナ**権限のみを必要とします。Essbase スーパーバイザに、データベース・デザイナ権限を持つ特別なグループを一時的なアプリケーションおよびデータベースのためにのみ設定してもらうことができます。ユーザーをそのグループに割り当てることができます。ユーザーは、システムへの他のアクセスでは、通常のユーザー権限に戻ります。この方法は、セキュリティ露出がより少なくなりますが、プログラムを実行する前に、より多くの設定が必要です。

Visual Basic のアウトライン API 関数の呼出し順序

アウトライン API を使用する場合、一部の API 関数は他の関数より先に呼び出す必要があります。基本的な呼出し順序は次のとおりです:

1. **EsxInit()**は、他の API 関数よりも先に呼び出します。

   この API はインスタンス・ハンドルを戻します。

2. サーバーにログ・オンするには、**EsxLogin()**または **EsxAutoLogin()**を呼び出します。

   この API はコンテキスト・ハンドルを戻します。

3. アウトラインを開く、または作成するには、**EsxOtlOpenOutline()**または **EsxOtlNewOutline()**を呼び出します。

   この API はアウトライン・ハンドルを戻します。

4. 現在のアウトラインをサーバーに書き込むには、**EsxOtlWriteOutline()**を呼び出します。**EsxOtlVerifyOutline()**は、この関数より先に呼び出していないかぎり、アウトラインの保存前に API によって自動的に呼び出されます。

5. アウトラインに対して行った変更に基づいてデータベースを再構築するには、**EsxOtlRestructure()**を呼び出します。

6. アウトラインを開いたときにロックされたアウトライン・オブジェクトのロックを解除するには、**EsxUnlockObject()**を呼び出します。

7. アウトラインに関連付けられているリソースを解放するには、**EsxOtlCloseOutline()**を呼び出します。

8. サーバーからログアウトするには、**EsxLogout()**を呼び出します。

   これでコンテキスト・ハンドルが無効になります。

9. セッションを終了するには、**EsxTerm()**を呼び出します。

   これでインスタンス・ハンドルが無効になります。

一般的な Visual Basic のアウトライン API タスクの順序

単純なアウトライン API アプリケーションの、一般的な操作順序を次に示します。

1. **ESX_INIT_T** 構造体を作成し、初期化します。
2. **EsxInit()**を呼び出してアウトライン API を初期化します。
3. ローカルの静的な構造体またはグローバル構造体を割り当てます。
4. **EsxLogin()**または **EsxAutoLogin()**を呼び出して、必要なサーバーにログインします。
5. **ESX_OUTLINEINFO_T** 構造体を作成し、初期化します(新規のアウトラインに対してのみ)。
6. **EsxOtlOpenOutline()**または **EsxOtlNewOutline()**を呼び出して、既存のアウトラインを開くか、新規のアウトラインを作成します。
7. アウトラインの処理を行います。
8. **EsxOtlVerifyOutline()**を呼び出してアウトラインを確認します。
9. **EsxOtlWriteOutline()**を呼び出して、確認済のアウトラインをサーバーに書き込みます。

   .OTN という拡張子でアウトラインが保存されます。

10. **EsxOtlRestructure()**を呼び出して、データベースを再構築します。

    .OTN ファイルが.OTL ファイルに変更されます。これは非同期の関数呼出しなので、プロセスが完了するまで **EsxGetProcessState()**を呼び出す必要があります。

11. **EsxUnlockObject()**を呼び出して、アウトラインのロックを解除します(オープン時にロックされた場合)。
12. **EsxOtlCloseOutline()**を呼び出して、アウトラインに関連付けられているすべての情報を解放します。
13. **EsxLogout()**を呼び出して、サーバーからログアウトします。
14. ローカルの静的な構造体またはグローバル構造体を解放します。
15. **EsxTerm()**を呼び出して、API を終了します。

22 Visual Basicのアウトライン APIの宣言

この章の内容

VB のアウトライン・エラー戻り値

次の表は、アウトライン API 呼出しが失敗したときに戻されるエラー・ステータス定数を説明しています。これらの値は、Essbase アウトライン API Visual Basic グローバル・テキスト・ファイル `esberror.bas` に定義されています。

表 21　エラーの戻り値

値	説明
OTLAPI_BAD_ALIASTABLE	別名テーブルが正しくありません
OTLAPI_BAD_CONSOL	集計のタイプが正しくありません(+、-、など)
OTLAPI_BAD_GENLEVELNAME	世代名またはレベル名が正しくありません
OTLAPI_BAD_HOUTLINE	EsbOtl...関数に渡されたアウトライン・ハンドルが無効です
OTLAPI_BAD_MBRNAME	メンバー名が無効です
OTLAPI_BAD_MEMBER	メンバーのハンドルが無効です
OTLAPI_BAD_MOVE	メンバーの移動が不正です; たとえば、メンバーを子孫に移動できません
OTLAPI_BAD_OBJTYPE	オブジェクト・タイプが不正です
OTLAPI_BAD_OUTLINETYPE	アウトライン型が無効です
OTLAPI_BAD_RENAMESHARE	共有メンバー名は変更できません
OTLAPI_BAD_RESTRUCTTYPE	再構築のタイプが正しくありません

値	説明
OTLAPI_BAD_SORTCOMPAREFUNC	ソート比較関数が無効です
OTLAPI_BAD_SORTTYPE	ソート型が無効です
OTLAPI_BAD_TRANSTYPE	トランザクション作成時のトランザクションのタイプが不明です(内部エラー)
OTLAPI_BAD_USERATTR	ユーザー属性が無効です
OTLAPI_CUR_NOACCOUNTS	会計次元がありません。通貨データベースを作成するには、会計次元が必要です
OTLAPI_CUR_NOCOUNTRY	国次元がありません。通貨データベースを作成するには、国次元が必要です
OTLAPI_CUR_NOTIME	時間次元がありません。通貨データベースを作成するには、時間次元が必要です。
OTLAPI_ERR_ADDNAMEUSED	メンバー名はすでに使用されています(追加操作)
OTLAPI_ERR_ALIASSHARED	共有メンバーに別名は指定できません
OTLAPI_ERR_ALIASTABLEEXISTS	別名テーブル名はすでに存在します
OTLAPI_ERR_ALIASTABLENAME	別名テーブル名が正しくありません
OTLAPI_ERR_ALREADYCURRENCY	このアウトラインは通貨アウトラインです。通貨アウトラインを作成しようとしていますが、初期アウトラインがすでに通貨アウトラインです。
OTLAPI_ERR_BADSHARE	共有値が正しくありません
OTLAPI_ERR_BADSKIP	タイム・バランス・スキップ値が正しくありません
OTLAPI_ERR_BADSTORAGE	次元ストレージ値が正しくありません
OTLAPI_ERR_ BADSTORAGECATEGORY	ストレージ・カテゴリが正しくありません
OTLAPI_ERR_BADTIMEBAL	タイム・バランス値が正しくありません
OTLAPI_ERR_CONFIGTOOMANYDIMS	次元が多すぎるため、自動的に構成できません
OTLAPI_ERR_COPYALIASTABLE	ソースと宛先のテーブルが同じです
OTLAPI_ERR_CREATETEMP	一時ファイル名を作成できません。読取り専用ドライブに作成しようとしている可能性があります。アウトラインをサーバーから開くかサーバーへ書き込むたびに、クライアントに一時ファイルが作成されます。
OTLAPI_ERR_CURTOOMANYDIMS	通貨アウトライン内の次元が多すぎます。1 つの通貨アウトラインに指定できる次元は、最大 4 つです。
OTLAPI_ERR_DELETEDEFALIAS	デフォルトの別名テーブルを削除できません
OTLAPI_ERR_DUP_LANGCODE	言語コードが、同じデータベース内の異なる別名テーブルに割り当てられています
OTLAPI_ERR_DUPGENLEVNAME	世代名またはレベル名と重複するメンバー名または別名の追加、変更または設定はできません

値	説明
OTLAPI_ERR_DUPLICATEALIAS	別名が重複しています
OTLAPI_ERR_DUPLICATENAME	メンバー名が重複しています
OTLAPI_ERR_FILEIO	ファイルの読取りまたはファイルへの書込みができませんでした
OTLAPI_ERR_FILEOPEN	ファイルを開けませんでした
OTLAPI_ERR_GENLEVELEXISTS	世代またはレベルには、すでに名前が付いています
OTLAPI_ERR_GENLEVELNAMEEXISTS	世代名またはレベル名がすでに存在しています
OTLAPI_ERR_GENLEVNAMEMBR	メンバー名または別名と重複する世代名またはレベル名は追加できません
OTLAPI_ERR_GENLEVELVALUE	世代値またはレベル値が正しくありません
OTLAPI_ERR_ILLEGALALIASSTRING	別名の組合せメンバーが正しくありません
OTLAPI_ERR_ILLEGALCOMBOALIAS	別名の組合せが正しくありません
OTLAPI_ERR_ILLEGALCURRENCY	通貨メンバーが正しくありません
OTLAPI_ERR_ILLEGALDEFALIAS	デフォルトの別名が正しくありません
OTLAPI_ERR_ILLEGALNAME	メンバー名が正しくありません
OTLAPI_ERR_ILLEGALTAG	次元タグ(カテゴリ)が正しくありません
OTLAPI_ERR_INVALIDOPTION	ユーザーが無効なオプションを **EssOtlGetGenNames()**または **EssOtlGetLevelNames()**に対して渡した場合に発生します
OTLAPI_ERR_LEAFLABEL	リーフ・メンバーがラベル・メンバーとして定義されています
OTLAPI_ERR_MAXALIASTABLES	別名テーブルの数が最大に達しました
OTLAPI_ERR_MEMBERCALC	メンバー式が正しくありません
OTLAPI_ERR_NOALIAS	このメンバーには、別名がありません
OTLAPI_ERR_NOALIASCOMBO	別名の組合せがありません
OTLAPI_ERR_ NODTSMBRANDGENMATCH	この世代の DTS メンバーは有効にできません
OTLAPI_ERR_NOFORMULA	このメンバーには、式がありません
OTLAPI_ERR_NOSHAREPROTO	共有メンバーに実際のメンバーが付加されていません
OTLAPI_ERR_NOTADIM	次元名が必要です
OTLAPI_ERR_NOTIMEDIM	時間次元が定義されていません(時間次元がない場合、タイム・バランス操作が実行できません)
OTLAPI_ERR_NOTVERIFIED	アウトラインにエラーがあります(サーバーへの保存時)
OTLAPI_ERR_OPENMODE	この呼出しを行うために、ファイルが不適切なモードで開かれました。アウトラインを開くために **EssOtlOpenOutlineQuery()**を呼び出す場合、呼出しのすべてが機能するとはかぎりません。

値	説明
OTLAPI_ERR_RENAMEDEFALIAS	デフォルトの別名テーブル名を変更できません
OTLAPI_ERR_RENAMENAMEUSED	メンバー名はすでに使用されています(名前変更操作)
OTLAPI_ERR_SHAREDMEMBERFORMULA	共有メンバーは式を持つことができません
OTLAPI_ERR_SHARENOTLEVEL0	共有メンバーのレベルが 0 になっていません(共有メンバーを別のメンバーの親にすることはできません)
OTLAPI_ERR_SHAREUDA	共有メンバーに対してユーザー属性は設定できません
OTLAPI_ERR_TIMESPARSE	会計次元が密で、時間次元が疎、つまり未使用の状態になっています
OTLAPI_NULL_ARG	NULL 引数が EsbOtl...関数に渡されました
OTLAPI_NO_GENLEVELNAME	世代名またはレベル名が見つかりません
OTLAPI_NO_USERATTR	ユーザー属性が見つかりません
OTLAPI_SORT_TOOMANY	ソートの対象となるメンバーの数が多すぎます(最大のソート容量は、64K/4 メンバーです)

VB アウトライン・シンボリック定数の定義

次の表は、特にアウトライン API によって使用されるシンボリック定数を説明しています。これらの定数は、Essbase Visual Basic グローバル・テキスト・ファイル `esb32.bas` で定義されています。

表 22　再構築値

値	説明
ESB_DOR_ALLDATA	すべてのデータを保持します
ESB_DOR_NODATA	すべてのデータを破棄します

値	説明
ESB_DOR_LOWDATA	レベル 0 のデータのみを保持します
ESB_DOR_INDATA	入力データのみを保持します

表 23　会計メンバーの通貨換算カテゴリ値

値	説明
ESB_CONV_NONE	デフォルトの変換カテゴリ。メンバーは、親からカテゴリを継承します。
ESB_CONV_CATEGORY	このメンバーに対する通貨換算カテゴリを定義します
ESB_CONV_NOCONV	このメンバーには換算がありません

表 24　会計メンバーのタイム・バランス値

値	説明
ESB_TIMEBAL_NONE	タイム・バランスはありません
ESB_TIMEBAL_FIRST	最初のタイム・バランス・メンバー
ESB_TIMEBAL_LAST	最後のタイム・バランス・メンバー
ESB_TIMEBAL_AVG	平均のタイム・バランス・メンバー

表 25　会計メンバーのタイム・バランス・スキップ値

値	説明
ESB_SKIP_NONE	何もスキップしません
ESB_SKIP_MISSING	データが#missing の場合、値をスキップします
ESB_SKIP_ZEROS	データが 0 の場合、値をスキップします
ESB_SKIP_BOTH	データが#missing または 0 の場合、値をスキップします

注： 会計メンバーのタイム・バランス・スキップ値は、タイム・バランスが **ESB_TIMEBAL_NONE** と等しくない場合のみ有効です。

表 26　共有定数

値	説明
ESB_SHARE_DATA	通常メンバー(デフォルト値)
ESB_SHARE_NEVER	たとえ暗黙的に共有になるような場合でも、このメンバーを共有しません。
ESB_SHARE_LABEL	ラベル・メンバー。このメンバーに対してはデータを保管しません。
ESB_SHARE_SHARE	共有メンバー。このメンバーは子を持つことができず、同じ次元に同じ名前の実メンバーが存在している必要があります。

表 27　次元カテゴリ(タグ)

値	説明
ESB_CAT_NONE	カテゴリはありません
ESB_CAT_ACCOUNTS	会計次元
ESB_CAT_TIME	時間次元
ESB_CAT_COUNTRY	国次元
ESB_CAT_TYPE	タイプ次元。この次元は、通貨データベースでのみ有効です
ESB_CAT_CURPARTITION	通貨パーティション次元。非通貨データベースでのみ有効です。

表 28　自動構成ストレージの最適化で使用される次元カテゴリ

値	説明
ESB_STORECAT_OTHER	ストレージ・カテゴリがないか、または不明です
ESB_STORECAT_TIME	時間ストレージ・カテゴリ
ESB_STORECAT_UNITS	単位ストレージ・カテゴリ
ESB_STORECAT_SCENARIO	シナリオ・ストレージ・カテゴリ
ESB_STORECAT_ACCOUNTS	会計ストレージ・カテゴリ
ESB_STORECAT_PRODUCT	製品ストレージ・カテゴリ
ESB_STORECAT_ORGAN	組織ストレージ・カテゴリ
ESB_STORECAT_MARKET	マーケット・ストレージ・カテゴリ
ESB_STORECAT_CUSTOMER	顧客ストレージ・カテゴリ
ESB_STORECAT_DIST	流通チャネル・ストレージ・カテゴリ
ESB_STORECAT_BUSUNIT	ビジネス・ユニット・ストレージ・カテゴリ
ESB_STORECAT_GEOG	地域ストレージ・カテゴリ

注：　ストレージ自動構成の使用時に、ストレージの最適化のために使用されます

表 29　ソート・オプション

値	説明
ESB_SORT_ASCENDING	昇順でのソート
ESB_SORT_DESCENDING	降順でのソート

クエリー・タイプ

ESB_PREDICATE_T で実行する操作を定義するために使用されます。

- ESB_CHILDREN
- ESB_DESCENDANTS
- ESB_BOTTOMLEVEL
- ESB_SIBLINGS
- ESB_SAMELEVEL
- ESB_SAMEGENERATION
- ESB_PARENT
- ESB_DIMENSION
- ESB_NAMEDGENERATION
- ESB_NAMEDLEVEL
- ESB_SEARCH
- ESB_WILDSEARCH
- ESB_USERATTRIBUTE
- ESB_ANCESTORS

表 30　クエリー・オプション

値	説明
ESB_MEMBERSONLY	ESB_SEARCH、ESB_WILDSEARCH に対して有効です
ESB_ALIASESONLY	ESB_SEARCH、ESB_WILDSEARCH に対して有効です
ESB_MEMBERSANDALIASES	ESB_SEARCH、ESB_WILDSEARCH に対して有効です
ESB_COUNTONLY	任意のクエリー・タイプに対して有効です。データを戻さずにアウトラインにクエリーを行います。ESB_PMBRCOUNTS_T の ulTotalCount フィールドに記入することにより、そのクエリー・タイプを満たすメンバーの人数を戻します。

注： **ESB_PREDICATE_T** で特定のクエリーのタイプについて指定できます。

表 31　世代/レベル・オプション

値	説明
ESB_GENLEV_ALL	デフォルト名およびユーザー定義名を戻します
ESB_GENLEV_ACTUAL	ユーザー定義名のみを戻します
ESB_GENLEV_DEFAULT	ユーザー定義名も持つ世代およびレベルのデフォルト名を含むすべてのデフォルト名を戻します
ESB_GENLEV_NOACTUAL	ユーザー定義名も持つ世代およびレベルのデフォルト名を除くすべてのデフォルト名を戻します

注： **EsbOtlGetGenNames** および **EsbOtlGetLevelNames** と使用できます。

ESB_ATTRIBUTEQUERY_T

指定した属性メンバーまたは次元の属性情報を含んでいます。
EsbGetAttributeInfo が使用します。フィールドは次のとおりです:

```
  Type ESB_ATTRIBUTEQUERY_T
 InputMember    As Variant
 InputMemberType  As Integer
 OutputMemberType As Integer
 Operation     As Integer
 Attribute     As Variant
End Type
```

VB データ型	フィールド	説明
As Variant	InputMember	属性メンバーまたは次元
As Integer	InputMemberType	次のメンバー・タイプのいずれかになります: ● ESB_ATTRIBUTE_DIMENSION ● ESB_ATTRIBUTE_MEMBER ● ESB_STANDARD_DIMENSION ● ESB_STANDARD_MEMBER ● ESB_BASE_DIMENSION ● ESB_BASE_MEMBER ● ESB_ATTRIBUTED_MEMBER 表 20 を参照してください。
As Integer	OutputMemberType	次のメンバー・タイプのいずれかになります: ● ESB_ATTRIBUTE_DIMENSION ● ESB_ATTRIBUTE_MEMBER ● ESB_STANDARD_DIMENSION ● ESB_STANDARD_MEMBER ● ESB_BASE_DIMENSION ● ESB_BASE_MEMBER ● ESB_ATTRIBUTED_MEMBER ● ESB_INVALID_MEMBER

VB データ型	フィールド	説明
As Integer	Operation	次の操作のいずれかになります: ● ESB_EQ: 等しい ● ESB_NEQ: 等しくない ● ESB_GT: より大きい ● ESB_LT: より小さい ● ESB_GTE: より大きいまたは等しい ● ESB_LTE: より小さいまたは等しい ● ESB_TYPEOF ● ESB_ALL
As Variant	Attribute	属性値

ESB_GENLEVELNAME_T

世代名およびレベル名に関する情報を含んでいます。

```
   Type ESB_GENLEVELNAME_T

  usNumber  As Integer
  szName   As String * ESB_MBRNAMELEN
End Type
```

データ型	フィールド	説明
Integer	usNumber	世代番号またはレベル番号。
STRING * ESB_MBRNAMELEN	szName	世代名またはレベル名。

ESB_MBRCOUNTS_T

```
   Type ESB_MBRCOUNTS_T

  ulStart     As Long
  ulMaxCount   As Long
  ulTotalCount  As Long
  ulReturnCount  As Long
End Type
```

データ型	フィールド	説明
Long	ulStart	情報の取得のための開始メンバー。
Long	ulMaxCount	取得するメンバーの最大数。

データ型	フィールド	説明
Long	ulTotalCount	クエリーの結果存在するメンバーの総数を戻します。
Long	ulReturnCount	戻されたメンバーのハンドルの数を戻します。

ESB_MBRINFO_T

アウトライン・メンバーに関する情報を含みます。

```
  Type ESB_MBRINFO_T

 szMember     As String * ESB_MBRNAMELEN
 usLevel      As Integer
 usGen       As Integer
 usConsolidation  As Integer
 fTwoPass     As Integer
 fExpense     As Integer
 usConversion    As Integer
 szCurMember    As String * ESB_MBRNAMELEN
 usTimeBalance   As Integer
 usSkip       As Integer
 usShare      As Integer
 usStorage     As Integer
 usCategory    As Integer
 usStorageCategory As Integer
 ulChildCount    As Long
 szComment     As String * ESB_MBRCOMMENTLEN
 szDimName     As String * ESB_MBRNAMELEN
 Attribute     As Variant
 IsAttributed    As Integer
End Type
```

データ型	フィールド	説明
STRING * ESB_MBRNAMELEN	szMember	メンバー名。このフィールドは、メンバー作成時に呼出し元でのみ設定できます。
Integer	usLevel	アウトラインのメンバーのレベル。このフィールドは変更できません。
Integer	usGen	アウトラインのメンバーの世代。このフィールドは変更できません。
Integer	usConsolidation	単項集計タイプ。次のいずれかになります: ● ESB_UCALC_ADD ● ESB_UCALC_SUB ● ESB_UCALC_MULT ● ESB_UCALC_DIV ● ESB_UCALC_PERCENT ● ESB_UCALC_NOOP
Integer	fTwoPass	2 パス計算メンバーの場合 ESB_TRUE

データ型	フィールド	説明
Integer	fExpense	支出メンバーの場合、ESB_TRUE
Integer	usConversion	通貨換算タイプ。このフィールドは会計次元のメンバーに対してのみ有効です。次のいずれかになります: ● ESB_CONV_NONE ● ESB_CONV_CATEGORY ● ESB_CONV_NOCONV
STRING * ESB_ MBRNAMELEN	szCurMember	メンバーが会計次元で、usConversion が ESB_CONV_CATEGORY である場合、このフィールドは通貨カテゴリを定義します。メンバーが国次元である場合、このフィールドは通貨名を定義します。このフィールドは、他のすべての状況においては定義されません。
Integer	usTimeBalance	タイム・バランス・オプション。このフィールドは会計次元のメンバーに対してのみ有効です。次のいずれかになります: ● ESB_TIMEBAL_NONE ● ESB_TIMEBAL_FIRST ● ESB_TIMEBAL_LAST ● ESB_TIMEBAL_AVG
Integer	usSkip	タイム・バランス・スキップ・オプション。これは、usTimeBalance が ESB_TIMEBAL_NONE と等しくない場合に会計次元のメンバーにのみ有効です。次のいずれかになります: ● ESB_SKIP_NONE ● ESB_SKIP_MISSING ● ESB_SKIP_ZEROS ● ESB_SKIP_BOTH
Integer	usShare	共有オプション。次のいずれかになります: ● ESB_SHARE_DATA (デフォルト値) ● ESB_SHARE_DYNCALCSTORE ● ESB_SHARE_DYNCALCNOSTORE ● ESB_SHARE_NEVER ● ESB_SHARE_LABEL ● ESB_SHARE_SHARE (レベル 0 メンバーにのみ有効)
Integer	usStorage	次元ストレージ・タイプ。このフィールドは、次元メンバーに対してのみ有効です。次のいずれかの値になります: ● ESB_DIMTYPE_DENSE ● ESB_DIMTYPE_SPARSE

データ型	フィールド	説明
Integer	usCategory	次元カテゴリ。このフィールドは、次元メンバーおよび属性メンバーに対してのみ有効です。次のいずれかになります: ● ESB_CAT_ACCOUNTS ● ESB_CAT_ATTRCALC (内部での使用専用) ● ESB_CAT_ATTRIBUTE ● ESB_CAT_COUNTRY ● ESB_CAT_CURPARTITION (非通貨データベースのみ) ● ESB_CAT_NONE ● ESB_CAT_TIME (通貨データベースのみ) ● ESB_CAT_TYPE
Integer	usStorageCategory	次元ストレージ・カテゴリ。このフィールドは、次元メンバーおよび属性メンバーに対してのみ有効です。アウトラインが自動最適化用に構成されているとき、次元のストレージ・タイプを最適化するために使用されます。次のいずれかになります: ● ESB_STORECAT_ACCOUNTS ● ESB_STORECAT_ATTRCALC (内部での使用専用) ● ESB_STORECAT_ATTRIBUTE ● ESB_STORECAT_BUSUNIT ● ESB_STORECAT_CUSTOMER ● ESB_STORECAT_DIST ● ESB_STORECAT_GEOG ● ESB_STORECAT_MARKET ● ESB_STORECAT_ORGAN ● ESB_STORECAT_OTHER ● ESB_STORECAT_PRODUCT ● ESB_STORECAT_SCENARIO ● ESB_STORECAT_TIME ● ESB_STORECAT_UNITS
Long	ulChildCount	このフィールドは、ESB_MBRNAME_T に指定されているメンバーの子の合計数を含みます。
STRING * ESB_ MBRCOMMENTLEN	szComment	メンバー・コメント配列。
STRING * ESB_ MBRNAMELEN	szDimName	次元名。

データ型	フィールド	説明
VARIANT	Attribute	属性値: 属性次元またはゼロレベル(リーフ・ノード)の属性メンバーについては、次のデータ型のいずれかになります: ● ESB_ATTRMBRDT_BOOL ● ESB_ATTRMBRDT_DATETIME ● ESB_ATTRMBRDT_DOUBLE ● ESB_ATTRMBRDT_STRING 属性次元ではなく、属性メンバーの場合: ● ESB_ATTRMBRDT_NONE ● ESB_ATTRMBRDT_AUTO
Integer	IsAttributed	メンバーに属性が関連付けられているかどうかを示します。

ESB_OUTERROR_T

アウトラインを確認するときに、各メンバーのエラーを戻します。エラーは、32ビットのステータス・ワードで戻される、ビット・フィールド値です。各エラー値は、表 21 で説明する関数呼出しエラーの戻り値に対応します。

```
   Type ESB_OUTERROR_T

  hMember   As Long
  ulErrors  As Long
End Type
```

データ型	フィールド	説明
Long	hMember	エラーのあるメンバーへのハンドル。

データ型	フィールド	説明
Long	ulErrors	メンバーに対するエラーのビットマスク。次の値の任意の組合せです: ● ESB_OUTERROR_ALIASSHARED ● ESB_OUTERROR_BADCATEGORY ● ESB_OUTERROR_BADSHARE ● ESB_OUTERROR_BADSKIP ● ESB_OUTERROR_BADSTORAGE ● ESB_OUTERROR_BADSTORAGECATEGORY ● ESB_OUTERROR_BADTIMEBAL ● ESB_OUTERROR_CURTOOMANYDIMS ● ESB_OUTERROR_DUPGENLEVNAME ● ESB_OUTERROR_DUPLICATEALIAS ● ESB_OUTERROR_DUPLICATENAME ● ESB_OUTERROR_ILLEGALALIASSTRING ● ESB_OUTERROR_ILLEGALCOMBOALIAS ● ESB_OUTERROR_ILLEGALCURRENCY ● ESB_OUTERROR_ILLEGALDEFALIAS ● ESB_OUTERROR_ILLEGALNAME ● ESB_OUTERROR_ILLEGALTAG ● ESB_OUTERROR_LEAFLABEL ● ESB_OUTERROR_MEMBERCALC ● ESB_OUTERROR_NOSHAREPROTO ● ESB_OUTERROR_NOTIMEDIM ● ESB_OUTERROR_SHAREDMEMBERFORMULA ● ESB_OUTERROR_SHARENOTLEVEL0 ● ESB_OUTERROR_SHAREUDA ● ESB_OUTERROR_TIMESPARSE

ESB_OUTLINEINFO_T

```
   Type ESB_OUTLINEINFO_T

  fCaseSensitive As String * 1
  usOutlineType  As Integer
  fAutoConfigure As String * 1
End Type
```

データ型	フィールド	説明
String * 1	fCaseSensitive	大文字と小文字を区別するメンバー名の場合は、ESB_TRUE。

データ型	フィールド	説明
Integer	usOutlineType	アウトラインのタイプ。次のいずれかになります: ● ESB_DBTYPE_NORMAL ● ESB_DBTYPE_CURRENCY
String * 1	fAutoConfigure	ESB_TRUE の場合、アウトライン保存時の次元のストレージ(密/疎)が自動的に構成されます。

ESB_PREDICATE_T

クエリー記述に関する情報を含んでいます。

```
   Type ESB_PREDICATE_T

  ulQuery     As Long
  ulOptions    As Long
  pszDimension  As String * ESB_MBRNAMELEN
  pszString1   As String * 256
  pszString2   As String * 256
End Type
```

データ型	フィールド	説明
Long	ulQuery	クエリーのタイプ。詳細は、EsbOtlQueryMembers を参照してください。
Long	ulOptions	クエリーのタイプに依存するオプション。詳細は、EsbOtlQueryMembers を参照してください。
STRING * ESB_MBRNAMELEN	pszDimension	次元名。詳細は、EsbOtlQueryMembers を参照してください。
String * 256	pszString1	入力文字列の値。詳細は、EsbOtlQueryMembers を参照してください。
String * 256	pszString2	入力文字列の値。詳細は、EsbOtlQueryMembers を参照してください。

23 Visual Basicのアウトライン API関数

この章の内容

Visual Basic のアウトライン API 関数のカテゴリ

VB のアウトライン API 別名テーブル関数

次の関数は、別名テーブルに対する操作を実行します。

関数	説明
EsbOtlCreateAliasTable	アウトラインに空の別名テーブルを作成します
EsbOtlCopyAliasTable	別名テーブルを他の別名テーブルにコピーします
EsbOtlRenameAliasTable	既存の別名テーブル名を変更します

関数	説明
EsbOtlClearAliasTable	既存の別名テーブルを削除せずに、そのエントリをすべて消去します
EsbOtlDeleteAliasTable	別名テーブルをアウトラインから削除し、そのエントリをすべて消去します
EsbOtlSetAliasTableLanguage	別名テーブルの言語コードを設定します。1 つの別名テーブルに複数の言語コードを設定できます。
EsbOtlGetAliasTableLanguages	別名テーブルと関連付けられている言語コードのセットを取得します。
EsbOtlClearAliasTableLanguages	別名テーブルと関連付けられている言語コードのセットを消去します。

VB アウトライン API 属性の関数

次の Visual Basic のアウトライン関数は、属性に関するものです。

関数	説明
EsbOtlAssociateAttributeDimension	属性次元を基本次元に関連付けます
EsbOtlAssociateAttributeMember	属性メンバーを基本メンバーに関連付けます
EsbOtlDisassociateAttributeDimension	属性次元と基本次元との関連付けを解除します
EsbOtlDisassociateAttributeMember	属性メンバーと基本メンバーとの関連付けを解除します
EsbOtlFindAttributeMembers	属性メンバーに関連付けられているすべての基本メンバーを戻します
EsbOtlGetAssociatedAttributes	基本メンバーまたは基本次元に関連付けられているすべての属性メンバーを戻します
EsbOtlGetAttributeInfo	指定した属性メンバーまたは属性次元に関する属性情報を戻します
EsbOtlGetAttributeSpecifications	アウトラインの属性指定を取得します
EsbOtlQueryAttributes	属性に関する複雑なクエリーを実行します
EsbOtlSetAttributeSpecifications	アウトラインの属性指定を設定します

VB のメイン APIVB のメイン API 属性関数項も参照してください。

VB のアウトライン API 動的時系列関数

次の関数は、動的時系列メンバーおよび別名を使用可能にして処理します。

関数	説明
EsbOtlDeleteDTSMemberAlias	動的時系列メンバーの別名を削除します。

関数	説明
EsbOtlEnableDTSMember	アウトラインの新規動的時系列メンバーを使用可能にします。
EsbOtlGetEnabledDTSMembers	アウトラインに対して定義されている新規動的時系列メンバーを取得します。
EsbOtlGetDTSMemberAlias	動的時系列メンバーの別名を取得します。
EsbOtlSetDTSMemberAlias	動的時系列メンバーの別名を設定します。

VB のアウトライン API 世代名関数

次の関数は、世代名に対する操作を実行します。

関数	説明
EsbOtlGetGenName	指定された次元および世代番号の世代名を取得します
EsbOtlGetGenNames	特定の次元に対して指定されたすべての世代名を取得します
EsbOtlSetGenName	指定された次元および世代番号の世代名を設定します
EsbOtlDeleteGenName	指定された次元およびレベル番号の世代名を削除します

VB のアウトライン API レベル名関数

次の関数は、レベル名に対する操作を実行します。

関数	説明
EsbOtlGetLevelName	指定された次元のレベル名を取得します
EsbOtlGetLevelNames	特定の次元に対して指定されたすべてのレベル名を取得します
EsbOtlSetLevelName	指定された次元のレベル名を設定します
EsbOtlDeleteLevelName	指定された次元のレベル名を削除します

VB のアウトライン API メンバー管理関数

次の関数は、アウトラインのメンバーの管理を支援します。

関数	説明
EsbOtlAddMember	メンバーを追加します
EsbOtlDeleteMember	メンバーを削除します
EsbOtlAddDimension	次元を追加します
EsbOtlDeleteDimension	次元を削除します

関数	説明
EsbOtlRenameMember	メンバー名を変更します
EsbOtlMoveMember	メンバーを移動します
EsbOtlFindMember	メンバーを検索します
EsbOtlGetMemberInfo	メンバー情報を取得します
EsbOtlSetMemberInfo	メンバー情報を設定します

VB のアウトライン API メンバー別名関数

次の関数は、メンバー別名に対する操作を実行します。

関数	説明
EsbOtlFindAlias	指定された別名を持つメンバーを検索します
EsbOtlGetMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を取得します
EsbOtlSetMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を設定します
EsbOtlDeleteMemberAlias	特定の別名テーブルの特定のメンバーに対する、デフォルトのメンバー別名を削除します
EsbOtlAddAliasCombination	特定の別名テーブルに対するメンバーに別名の組合せを追加します
EsbOtlDeleteAliasCombination	特定の別名テーブルに対するメンバーから別名の組合せを削除します
EsbOtlGetNextAliasCombination	指定された別名テーブルの指定されたメンバーに対する、別名の組合せを戻します

VB のアウトライン API メンバー式関数

次の関数は、メンバー式に対する操作を実行します。

関数	説明
EsbOtlGetMemberFormula	指定されたメンバーの式を取得します
EsbOtlGetMemberLastFormula	メンバーの計算に使用された最後の式を戻します
EsbOtlSetMemberFormula	指定されたメンバーに対して式を設定します
EsbOtlDeleteMemberFormula	指定されたメンバーの式を削除します

VB のアウトライン API メンバー走査関数

次の関数は、アウトライン・ツリーの走査に使用されます。

関数	説明
EsbOtlGetFirstMember	アウトラインの最初のメンバー、すなわちアウトラインで最初に定義されている次元へメンバーのハンドルを戻します
EsbOtlGetChild	メンバーの子へメンバーのハンドルを戻します
EsbOtlGetParent	メンバーの親へメンバーのハンドルを戻します
EsbOtlGetNextSibling	メンバーの次の兄弟へメンバーのハンドルを戻します
EsbOtlGetPrevSibling	メンバーの前の兄弟へメンバーのハンドルを戻します
EsbOtlGetNextSharedMember	実メンバーの次の共有メンバーへメンバーのハンドルを戻します

VB のアウトライン API アウトライン管理関数

次の関数は、アウトラインの管理を支援します。

関数	説明
EsbOtlGetOutlineInfo	アウトライン・ファイルに関する情報を戻します
EsbOtlGetUpdateTime	指定されたアウトラインに対するタイムスタンプを戻します
EsbOtlSetOutlineInfo	アウトライン情報を設定します
EsbOtlVerifyOutline	アウトラインが正しいことを確認します
EsbOtlSortChildren	アウトライン・メンバーの子をソートします
EsbOtlGenerateCurrencyOutline	既存のアウトラインを基に通貨アウトラインを生成します

VB のアウトライン API アウトライン・クエリー関数

次の関数は、アウトラインのクエリー実行を支援します。

関数	説明
EsbOtlOpenOutlineQuery	既存のアウトラインを開きます
EsbOtlQueryMembers	メンバーのハンドルを使用して、アウトラインにクエリーを行います
EsbOtlQueryMembersByName	メンバー名文字列を使用して、アウトラインにクエリーを行います
EsbOtlFreeMember	**EsbOtlQueryMembers()**から戻されたメンバー配列を解放します

VB のアウトライン API 設定およびクリーンアップ関数

次の関数は、アウトラインの編集操作を開始および終了します。

関数	説明
EsbOtlNewOutline	アウトラインを新規に作成します
EsbOtlOpenOutline	既存のアウトラインを開きます
EsbOtlWriteOutline	アウトラインをサーバーに書き込みます
EsbOtlRestructure	新規に保存されたアウトラインに基づいて、データベースを再構築します
EsbOtlCloseOutline	アウトラインに関連付けられているリソースを解放します

VB のアウトライン API ユーザー属性関数

次の関数は、ユーザー属性に対する操作を実行します。

関数	説明
EsbOtlGetDimensionUserAttributes	指定された次元のユーザー属性を取得します
EsbOtlGetUserAttributes	指定されたメンバーのユーザー属性を取得します
EsbOtlSetUserAttribute	指定されたメンバーに対してユーザー属性を設定します
EsbOtlDeleteUserAttribute	指定されたメンバーのユーザー属性を削除します

Visual Basic のアウトライン API 関数のリファレンス

「コンテンツ」ペインで、EsbOtl が前に付いた Visual Basic のアウトライン API 関数のアルファベット順リストを参照してください。

EsbOtlAddAliasCombination

単一の別名テーブルに対するメンバーに別名の組合せを追加します。

構文

```
      EsbOtlAddAliasCombination
       (
      hOutline, hMember, pszAliasTable, pszAlias, pszCombination
      )
ByVal
      hOutline
         As Long
```

```
ByVal
     hMember
        As Long
ByVal
     pszAliasTable
      As String
ByVal
     pszAlias
        As String
ByVal
     pszCombination
      As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名の組合せを作成するメンバーのハンドル。
PszAliasTable	組合せの追加先の別名テーブル。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszAlias	別名。
PszCombination	別名に関連付けられたメンバーの組合せ。これは次元間メンバー・リストの場合もあります。

備考

メンバー・ハンドルは共有メンバーにできません。共有メンバーには別名は使用できません。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_ALIASSHARED
- OTLAPI_ERR_ILLEGALCOMBOALIAS
- OTLAPI_ERR_ILLEGALALIASSTRING
- OTLAPI_ERR_DUPLICATEALIAS

例

```
    Declare Function EsbOtlAddAliasCombination Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember
As Long, ByVal pszAliasTable As String, ByVal pszAlias As String, ByVal
pszCombination As String) As Long

Sub ESB_EsbOtlAddAliasCombination()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
```

```
Dim hMemberJan As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Jan", hMemberJan)
End If
If sts = 0 And hMemberJan <> 0 Then
 sts = EsbOtlAddAliasCombination(hOutline,
  hMemberJan, "Default", "alias combination",
  "Year->Market")
End If
End Sub
```

関連トピック

- EsbOtlDeleteAliasCombination
- EsbOtlGetNextAliasCombination

EsbOtlAddDimension

アウトラインに次元を追加し、メンバーの属性を設定します。

構文

```
      EsbOtlAddDimension
       (
      hOutline, pMemberInfo, hPrevSibling, pszDataMbr, phMember
      )
ByVal
      hOutline
        As Long

      pMemberInfo
       As ESB_MBRINFO_T
ByVal
      hPrevSibling
       As Long
ByVal
      pszDataMbr
        As String

      phMember
        As Long
```

パラメータ　説明

hOutline	アウトラインのコンテキスト・ハンドル。

パラメータ　説明

pMemberInfo　メンバーとその属性を定義しているメンバー情報構造体。

HPrevSibling 前の兄弟のハンドル。このフィールドが ESB_NULL である場合、次元は、アウトラインの最初の次元になります。それ以外の場合、次元は hPrevSibling で指定された次元の後に配置されます。

PszDataMbr　アウトラインが再構成されるときにデータ値を受領する、新規次元のメンバーのメンバー名。このフィールドが ESB_NULL である場合、次元メンバー自身が使用されます。

phMember　API から戻された新規メンバーのハンドル。

備考

- この関数は、アウトラインの再構築時にデータを割り当てることのできる新規次元のメンバーを指定します。
- ESB_MBRINFO_T 構造体を作成し、値を入れてから、この関数を呼び出す必要があります。
- 属性次元を追加するには、この関数を呼び出す必要があります。
- 属性次元でない次元を追加するには、この関数または **EsbOtlAddMember()**を呼び出します。
  - EsbOtlAddDimension()を使用すると、追加された次元の任意のメンバーを選択して、既存の次元に割り当てられているデータ値を割り当てることができます。
  - **EsbOtlAddMember()**を使用した場合、追加された次元の最上位メンバー(次元)が使用されます。
- pszDataMbr フィールドを有効にするには、fKeepTrans フラグを ESB_YES に設定して **EsbOtlOpenOutline()**を使用して、アウトラインを開いておくことが必要です。
- pszDataMbr フィールドで参照されたメンバーは、次元が作成された後 **EsbOtlAddMember()**を使用して、新しい次元に追加されます。再構成のときに参照されたメンバーが存在しない場合、次元メンバーがそのかわりに使用されます。
- 属性次元については、ESB_MBRINFO_T の各フィールドを次のように設定する必要があります:

フィールド	設定
usConsolidation	ESB_UCALC_NOOP
fTwoPass	ESB_FALSE
fExpense	ESB_FALSE
usConversion	ESB_CONV_NONE
usTimeBalance	ESB_TIMEBAL_NONE
usSkip	ESB_SKIP_NONE
usShare	ESB_SHARE_DYNCALCNOSTORE

フィールド	設定
usStorage	ESB_DIMTYPE_SPARSE
usCategory	ESB_CAT_ATTRIBUTE
usStorageCategory	ESB_STORECAT_ATTRIBUTE
Attribute	属性値。次の属性メンバー・データ型のいずれかになります: ❍ ESB_ATTRMBRDT_BOOL ❍ ESB_ATTRMBRDT_DATETIME ❍ ESB_ATTRMBRDT_DOUBLE ❍ ESB_ATTRMBRDT_STRING

- 属性次元に基本次元を関連付ける必要があります。
- 属性次元は、基本次元および標準次元の後ろに置く必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME
- OTLAPI_ERR_ADDDELETEDIMDYNAMICCALC
- OTLAPI_ERR_ADDNAMEUSED
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE
- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NONATTRDIMFOLLOWED
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM

例

```
    Declare Function EsbOtlAddDimension Lib "ESBOTLN"
(ByVal hOutline As Long, pMemberInfo As ESB_MBRINFO_T,
ByVal hPrevSibling As Long, ByVal pszDataMbr As String,
phMember As Long) As Long

Sub ESB_OtlAddDimension()
Dim sts As Long
Dim NewInfo as ESB_OUTLINEINFO_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
Dim hDimMeasures As Long
NewInfo.usOutlineType = ESB_DBTYPE_NORMAL
NewInfo.fCaseSensitive = ESB_FALSE
NewInfo.fAutoConfigure = ESB_TRUE
sts = EsbOtlNewOutline(hLocalCtx, NewInfo, hOutline)
If sts = 0 Then
 MbrInfo.szMember = "Measures"
  sts = EsbOtlAddDimension(hOutline,
   MbrInfo, ESB_NULL, "Profit", hDimMeasures)
End If
End Sub
```

関連トピック

- EsbOtlAddMember
- EsbOtlDeleteDimension
- EsbOtlDeleteMember
- EsbOtlGetMemberInfo

EsbOtlAddMember

アウトラインにメンバーを追加し、メンバーの属性を設定します。

構文

```
      EsbOtlAddMember
       (
      hOutline, pMemberInfo, hParent, hPrevSibling, phMember
      )
ByVal
      hOutline
        As Long

      pMemberInfo
       As ESB_MBRINFO_T
ByVal
      hParent
         As Long
ByVal
      hPrevSibling
       As Long
```

```
    phMember
      As Long
```

パラメータ　説明

hOutline　アウトラインのコンテキスト・ハンドル。

pMemberInfo　メンバーとその属性を定義しているメンバー情報構造体。

hparent　親のハンドル。このフィールドは、hPrevSibling フィールドが ESB_NULL の場合にのみ使用されます。

hPrevSibling　前の兄弟のハンドル。

phMember　API から戻された新規メンバーのハンドル。

備考

- ESB_MBRINFO_T 構造体を作成し、値を入れてから、この関数を呼び出す必要があります。
- 共有メンバーを作成している場合以外、メンバー名は一意である必要があります。
- 追加されたメンバーの位置は次のようになります:
  - 新規メンバーは、hPrevSibling メンバーの後に挿入されます。
  - hPrevSibling フィールドが ESB_NULL の場合、新規メンバーは hParent で指定した親の最初の子になります。
  - hParent も hPrevSibling も ESB_NULL の場合、新規メンバーはアウトラインの最初の次元になります。
- 共有メンバーを追加するには、次の条件に従います:
  - 共有メンバーはゼロ レベル(リーフ・ノード)メンバーである必要があります。(共有メンバーは子を持つことができません。)
  - 実際のメンバーが、次元内にすでに存在している必要があります。
  - ESB_MBRINFO_T 構造体の usShare フィールドを ESB_SHARE_SHARE に設定します。
- LABEL メンバーを追加するには、次の手順に従います:
  - 最初に、ラベル属性を設定せずにメンバーを追加します。
  - 次に、その子を追加します。
  - その後、**EsbOtlSetMemberInfo()**を使用してラベル・メンバーのラベル・タグを設定します。(ラベル・メンバーには子が必要です。)
- 属性メンバーを追加するには、ESB_MBRINFO_T の各フィールドを次のように設定します:

フィールド	設定
usConsolidation	ESB_UCALC_NOOP
fTwoPass	ESB_FALSE

フィールド	設定
fExpense	ESB_FALSE
usConversion	ESB_CONV_NONE
usTimeBalance	ESB_TIMEBAL_NONE
usSkip	ESB_SKIP_NONE
usShare	ESB_SHARE_DYNCALCNOSTORE
usStorage	ESB_DIMTYPE_SPARSE
usCategory	ESB_CAT_ATTRIBUTE
usStorageCategory	ESB_STORECAT_ATTRIBUTE
Attribute	属性値。属性次元またはゼロレベル(リーフ・ノード)の属性メンバーは、次のいずれかのデータ型になります: ❍ ブール(True|False) ❍ 日付(「09/19/2006」) ❍ Double (3.14) ❍ 文字列(「Hello」) 属性次元ではなく、属性メンバーの場合: ESB_ATTRMBRDT_NONE = 空を含むそれ以外すべて。

- **属性メンバーの追加に関する注意:**
  - ESB_ATTRMBRDT_STRING 型でないゼロレベルの属性メンバーを追加すると、1269 ページの「ESB_ATTRSPECS_T」構造体のアウトラインの指定を使用して、ESB_MBRINFO_T 構造体の szMember フィールドも属性メンバーのロング名に設定されます。
  - 属性次元のみでなく属性メンバーにも usCategory および usStorageCategory を設定する必要があります。(基本メンバーには usCategory および usStorageCategory を設定する必要はありません。基本次元のみに対して設定する必要があります。)
  - ESB_MBRINFO_T 構造体の szDimName フィールドは設定しません。
  - ESB_ATTRMBRDT_STRING 型でないゼロレベルの属性メンバーには、Attribute フィールドを設定しないでください。属性値は、属性メンバーのロング名を変換して内部で導出されます。
  - 属性メンバーのデータ型を ESB_ATTRMBRDT_AUTO に設定した場合、Essbase は次の処理を実行します:
    - メンバー名がその型の値に変換できる場合、メンバーのデータ型をその次元のデータ型に設定します。
    - メンバー名がその次元のデータ型の値に変換できない場合、メンバーのデータ型を ESB_ATTRMBRDT_NONE に設定します。

        - ESB_ATTRMBRDT_AUTO から ESB_ATTRMBRDT_NONE 以外のデータ型に変換された最初の子メンバーについては、親のロング名をショート名に変換します。

- 次元を追加する場合:
    - 属性次元を追加するには、**EsbOtlAddDimension()**を呼び出します。**EsbOtlAddMember()**を呼び出さないでください。
    - 属性次元でない次元を追加するには、**EsbOtlAddDimension()**または**EsbOtlAddMember()**を呼び出します。
        - **EsbOtlAddDimension()**を使用すると、追加された次元の任意のメンバーを選択して、既存の次元に関連付けられているデータ値を割り当てることができます。
        - **EsbOtlAddMember()**を使用する場合、既存の次元に関連付けられているデータ値が、追加された次元の最上位メンバー(次元)に割り当てられます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME
- OTLAPI_ERR_ADDNAMEUSED
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE
- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM

例

```
    Declare Function EsbOtlAddMember Lib "ESBOTLN"
(ByVal hOutline As Long, pMemberInfo As ESB_MBRINFO_T,
ByVal hParent As Long, ByVal hPrevSibling As Long,
```

```
phMember As Long) As Long

Sub ESB_OtlAddMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
Dim hMemberProfit As Long
Dim hNewMember As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Profit", hMemberProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
 MbrInfo.szMember = "Inventory"
  sts = EsbOtlAddMember(hOutline, MbrInfo,
   ESB_NULL, hMemberProfit, hNewMember)
End If
End Sub
```

関連トピック

- EsbOtlAddDimension
- EsbOtlDeleteMember
- EsbOtlDeleteDimension
- EsbOtlSetMemberInfo
- EsbOtlFindMember

EsbOtlAssociateAttributeDimension

属性次元を標準次元または基本次元に関連付けます。

構文

```
      EsbOtlAssociateAttributeDimension
       (
      hOutline
      ,
      BaseDimension
      ,
      AttributeDimension
      )
ByVal
      hOutline
           As Long
ByVal
      BaseDimension
```

```
        As Long
ByVal
     AttributeDimension
      As Long
```

パラメータ	説明
hOutline	アウトラインのハンドル
BaseDimension	標準次元または基本次元のハンドル
AttributeDimension	基本次元のハンドル

備考

- 属性次元は疎である必要があります。
- 標準または基本次元は疎である必要があります。
- 属性次元を標準次元または基本次元に関連付ける必要があります。
- 複数の属性次元を 1 つの基本次元に関連付けることができます。
- 1 つの属性次元を複数の基本次元に関連付けることはできません。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlAssociateAttributeDimension()
    ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text box
    Dim sts as long
    Dim hOutline as long
Dim BaseMbr As Long
Dim AttrMbr As Long

hOutline = ESB_OtlOpenOutline
If hOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub
 ' abstract function (using EsbOtlFindMember()) to get member handle, while passing
in a prompt string
BaseMbr = ESB_OtlFindMember("Enter base dimension: ")
If BaseMbr = vbNull Then
Out "ESB_OtlFindMember() failed."
Exit Sub
End If

' abstract function (using EsbOtlFindMember()) to get member handle, while passing in
a prompt string
AttrMbr = ESB_OtlFindMember("Enter attribute dimension: ")
If AttrMbr = vbNull Then Out "ESB_OtlFindMember() failed.": Exit Sub

sts = EsbOtlAssociateAttributeDimension(ghOutline, BaseMbr, AttrMbr)
```

```
' abstract sub to call EsbOtlVerifyOutline(), ESBOTLNriteOutline(),
EsbOtlRestructure(),EsbUnlockObject() and
' EsbOtlCloseOutline() as neededà
tuckinoutline
If sts <> 0 Then Out "EsbOtlAssociateAttributeDimension failed: " & sts: Exit Sub
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlAssociateAttributeMember

属性メンバーを標準または基本メンバーに関連付けます。

構文

```
      EsbOtlAssociateAttributeMember
       (
      hOutline
      ,
      BaseMember
      ,
      AttributeMember
      )
ByVal
      hOutline
          As Long
ByVal
      BaseMember
         As Long
ByVal
      AttributeMember
       As Long
```

パラメータ	説明
hOutline	アウトラインのハンドル
BaseMember	標準または基本メンバーのハンドル

パラメータ　　説明

AttributeMember 属性メンバーのハンドル

備考

- この関数を使用して属性メンバーを標準または基本メンバーに関連付ける前に、**EsbOtlAssociateAttributeDimension()**を使用して、属性メンバーの次元を標準または基本メンバーの次元に関連付けてください。
- 属性メンバーは基本次元には関連付けられません。
- ゼロレベルの属性メンバーのみを標準または基本メンバーに関連付けられます。
- 与えられた属性次元のメンバーをレベルの異なる基本メンバーには関連付けられません。
- 1 つの属性次元の複数のメンバーを 1 つの基本メンバーに関連付けられません。
- 1 つの基本メンバーに 1 つ以上の属性次元のメンバーを関連付けられます。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
        Sub ESB_OtlAssociateAttributeMember()
        ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text
box
Dim BaseMbr As Long
    Dim AttrMbr As Long
    Dim sts as long
    Dim hOutline as long
hOutline = ESB_OtlOpenOutline
If hOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub

BaseMbr = ESB_OtlFindMember("Enter base dimension: ")
If BaseMbr = vbNull Then
Out "No valid member found."
Out "ESB_OtlAssociateAttributeDimension() failed."
Exit Sub
End If

    AttrMbr = ESB_OtlFindMember("Enter attribute dimension: ")
    If AttrMbr = vbNull Then
Out "No valid member found."
Out "ESB_OtlAssociateAttributeMember() failed."
Exit Sub
End If
sts = EsbOtlAssociateAttributeMember(hOutline, BaseMbr, AttrMbr)
' abstract sub to call EsbOtlVerifyOutline(), ESBOTLNriteOutline(),
```

```
EsbOtlRestructure(),EsbUnlockObject() and
' EsbOtlCloseOutline() as neededà
tuckinoutline
If sts <> 0 Then Out "EsbOtlAssociateAttributeMember failed" & sts: Exit Sub
ESB_OtlGetAttributeInfo
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlClearAliasTable

既存の別名テーブルを削除せずにそのエントリをすべて消去します。

構文

```
      EsbOtlClearAliasTable
       (
      hOutline, pszAliasTable
      )
ByVal
      hOutline
          As Long
ByVal
      pszAliasTable
        As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAliasTable	消去する別名テーブルの名前。デフォルト・テーブルには""または"デフォルト"を使用します。

備考

別名テーブルから別名を消去すると、その別名テーブルに関連付けられた言語コードは削除されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_ALIASTABLE

例

```
    Declare Function EsbOtlClearAliasTable Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal pszAliasTable As String) As Long

Sub ESB_OtlClearAliasTable()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlClearAliasTable(hOutline, "Default")
End If
End Sub
```

関連トピック

- EsbOtlCreateAliasTable
- EsbOtlCopyAliasTable
- EsbOtlRenameAliasTable
- EsbOtlDeleteAliasTable
- EsbOtlSetAliasTableLanguage

EsbOtlClearAliasTableLanguages

指定した別名テーブルに関連付けられた言語コードのセットが消去されます。

構文

```
      ESB_FUNC_M
      EsbOtlClearAliasTableLanguages
       (
      hOutline
      ,
      pszAliasTable
      )
ByVal
      hOutline
          As Long
ByVal
      pszAliasTable
        As String
```

パラメータ　　説明

hOutline　　アウトラインのハンドル。

pszAliasTable 関連付けられた言語コードがすべて削除される別名テーブル名。

戻り値

- 成功の場合、0 が戻されます。
- 処理に失敗すると、エラー OTLAPI_BAD_ALIASTABLE (無効な別名テーブル) が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbOtlGetAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String, pulCount As Long) As Long
Declare Function EsbOtlSetAliasTableLanguage Lib "esbotln" (ByVal hOutline As Long,
ByVal pszAliasTable As String, ByVal pszLanguageCode As String) As Long
Declare Function EsbOtlClearAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String) As Long

Sub ESB_Sub ()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Items As Long
Dim AliasLang As String * ESB_ALIASNAMELEN

Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlCreateAliasTable(hOutline,
  "French Alias Table")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr-CA")
End If

If sts = 0 Then
 sts = EsbOtlGetAliasTableLanguages(hOutline,
```

```
  "French Alias Table", Items)
 If sts = 0 Then
    For N = 1 To Items
          sts = EsbGetNextItem(hCtx, ESB_ALIASLANG_TYPE, ByVal AliasLang)
     Next
 End If
End If

If sts = 0 Then
 sts = EsbOtlClearAliasTableLanguages(hOutline,
  "French Alias Table")
End If

End Sub
```

関連トピック

- EsbOtlGetAliasTableLanguages
- EsbOtlSetAliasTableLanguage

EsbOtlCloseOutline

アウトラインに関連するすべての情報を解放します。

構文

```
      EsbOtlCloseOutline
       (
      hOutline
      )
ByVal
      hOutline
       As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

備考

- **EsbOtlNewOutline()**または **EsbOtlOpenOutline()**を呼び出す場合、常にこの関数を呼び出す必要があります。
- オブジェクトが開かれるときにロックされている場合、この呼出しを行う前に **EsbUnlockObject()**を呼び出す必要があります。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlCloseOutline Lib
```

```
"ESBOTLN" (ByVal hOutline As Long) As Long

Sub ESB_OtlCloseOutline()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
'body of code...
If sts = 0 Then
 sts = ESBOTLNriteOutline(hOutline, Object)
End If
'restructure outline using EsbOtlRestructure()
If sts = 0 Then
 sts = EsbOtlCloseOutline(hOutline)
End If
End Sub
```

関連トピック

- EsbOtlOpenOutline
- EsbOtlWriteOutline
- EsbOtlRestructure

EsbOtlCopyAliasTable

別名テーブルを他の別名テーブルにコピーします。

構文

```
     EsbOtlCopyAliasTable
      (
     hOutline, pszSourceAliasTable, pszDestAliasTable, fMerge
     )
ByVal
     hOutline
           As Long
ByVal
     pszSourceAliasTable
       As String
ByVal
     pszDestAliasTable
        As String
ByVal
     fMerge
            As Integer
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszSourceAliasTable	コピー元の別名テーブル名。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszDestAliasTable	コピー先の別名テーブル名。
fMerge	コピー元のファイルを既存のコピー先別名テーブルにマージする場合は、ESB_YES に設定します。コピー前にコピー先の別名テーブルを消去するには ESB_NO に設定します。

備考

- コピー先の別名テーブルが存在しない場合は、作成されます。コピー先の別名テーブルが存在する場合は、fMerge フラグが ESB_YES に設定されている場合を除き、最初に消去されます。
- 単一のブロック・ストレージまたは集約ストレージ・データベース・アウトライン内の別名テーブルの最大数は(デフォルトのテーブルを含めて)32 です。
- 別名テーブルをコピーすると、別名テーブルに関連付けられている言語コードがコピーされた別名テーブルから削除されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_MAXALIASTABLES
- OTLAPI_ERR_ALIASTABLENAME

例

```
    Declare Function EsbOtlCopyAliasTable Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszSourceAliasTable
As String, ByVal pszDestAliasTable As String,
ByVal fMerge As Integer) As Long

Sub ESB_OtlCopyAliasTable()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx,
Object, ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlCopyAliasTable
  (hOutline, "", "Alias Table 2", ESB_YES)
End If
```

```
End Sub
```

関連トピック

- EsbOtlCreateAliasTable
- EsbOtlClearAliasTable
- EsbOtlRenameAliasTable
- EsbOtlDeleteAliasTable
- EsbOtlSetAliasTableLanguage

EsbOtlCreateAliasTable

アウトラインに空の別名テーブルを作成します。

構文

```
      EsbOtlCreateAliasTable
       (
      hOutline, pszAliasTable
      )
ByVal
      hOutline
         As Long
ByVal
      pszAliasTable
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAliasTable	作成する別名テーブル名。

備考

- デフォルトの別名テーブルは常に存在するため、"デフォルト(Default)"という名前の別名テーブルは作成できません。
- 単一のブロック・ストレージまたは集約ストレージ・データベース・アウトライン内の別名テーブルの最大数は(デフォルトのテーブルを含めて)32 です。
- EsbOtlSetAliasTableLanguage API を使用して、別名テーブルに対して複数の言語コードを指定できます。別名テーブルを作成するとき、言語コードは指定されません

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_ALIASTABLEEXISTS
- OTLAPI_ERR_MAXALIASTABLES
- OTLAPI_ERR_ALIASTABLENAME

例

```
    Declare Function EsbOtlCreateAliasTable Lib
"ESBOTLN" (ByVal hOutline As Long,
ByVal pszAliasTable As String) As Long

Sub ESB_OtlCreateAliasTable()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlCreateAliasTable(hOutline,
  "Alias Table 1")
End If
End Sub
```

関連トピック

- EsbOtlCopyAliasTable
- EsbOtlRenameAliasTable
- EsbOtlDeleteAliasTable
- EsbOtlSetAliasTableLanguage

EsbOtlDeleteAliasCombination

単一の別名テーブルに対するメンバーから別名の組合せを削除します。

構文

```
      EsbOtlDeleteAliasCombination
       (
      hOutline, hMember, pszAliasTable, pszAlias
      )
ByVal
      hOutline
         As Long
ByVal
      hMember
          As Long
ByVal
      pszAliasTable
       As String
ByVal
      pszAlias
         As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名の組合せを削除するメンバーのハンドル。
pszAliasTable	組合せを削除する別名テーブル。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszAlias	削除する別名。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_NOALIASCOMBO

例

```
    Declare Function EsbOtlDeleteAliasCombination Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszAliasTable As String, ByVal pszAlias As String) As Long

Sub ESB_OtlDeleteAliasCombination()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberJan As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Jan", hMemberJan)
End If
If sts = 0 And hMemberJan <> 0 Then
 sts = EsbOtlDeleteAliasCombination(hOutline,
  hMemberJan, "Default", "alias combination")
End If
End Sub
```

関連トピック

- EsbOtlAddAliasCombination
- EsbOtlGetNextAliasCombination

EsbOtlDeleteAliasTable

アウトラインから指定された別名テーブルを削除し、そのエントリをすべて消去します。

構文

```
      EsbOtlDeleteAliasTable
       (
      hOutline, pszAliasTable
      )
ByVal
      hOutline
         As Long
ByVal
      pszAliasTable
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAliasTable	削除する別名テーブル名。

備考

デフォルトの別名テーブルは削除できません。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_DELETEDEFALIAS

例

```
    Declare Function EsbOtlDeleteAliasTable Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal
pszAliasTable As String) As Long

Sub ESB_OtlDeleteAliasTable()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlDeleteAliasTable(hOutline,
```

```
  "Alias Table 1")
End If
End Sub
```

関連トピック

- EsbOtlCreateAliasTable
- EsbOtlCopyAliasTable
- EsbOtlRenameAliasTable
- EsbOtlClearAliasTable

EsbOtlDeleteDimension

アウトラインから次元を削除します。また、この呼出しでは、アウトラインの再構築時にデータを保持しておく、削除対象の次元メンバーも指定します。

構文

```
      EsbOtlDeleteDimension
       (
      hOutline, hMember, pszDataMbr
      )

ByVal
      hOutline
        As Long
ByVal
      hMember
        As Long
ByVal
      pszDataMbr
       As String
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　削除するメンバーのハンドル。

pszDataMbr アウトラインの再構築時に保存するデータが含まれた次元メンバーの名前。このフィールドが""の場合は、次元が使用されます。

備考

- 次元およびその子孫のすべての共有メンバーが削除されます。
- 次元のすべてのメンバーが削除されます。
- 次元を削除するには、この呼出しまたは **EsbOtlDeleteMember()**を使用します。**EsbOtlDeleteDimension()**では、データベースの再構築時に他の次元に使用されるデータ値を含むメンバーを、削除された次元から選択できるという利点があります。**EsbOtlDeleteMember()**を使用する場合、削除された次元の上位メンバー(次元)のデータ値が使用されます。

- 「pszDataMbr」フィールドを有効にするには、fKeepTrans フラグを ESB_YES に設定して、**EsbOtlOpenOutline()**を呼び出してアウトラインを開いておく必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

OTLAPI_ERR_NOTIMEDIM

例

```
    Declare Function EsbOtlDeleteDimension Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszDataMbr As String) As Long

Sub ESB_OtlDeleteDimension()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberScenario As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Scenario", hMemberScenario)
End If
If sts = 0 And hScenario <> 0 Then
 sts = EsbOtlDeleteDimension(hOutline,
  hMemberScenario, "Actual")
End If
End Sub
```

関連トピック

- EsbOtlDeleteMember
- EsbOtlAddDimension
- EsbOtlAddMember
- EsbOtlFindMember
- EsbOtlGetMemberInfo

EsbOtlDeleteDTSMemberAlias

DTS メンバーの別名を削除します。

構文

EsbOtlDeleteDTSMemberAlias

```
      (
      hOutline, pszDTSMember, pszAliasTable
      )
ByVal
      hOutline
         As Long
ByVal
      pszDTSMember
       As String
ByVal
      pszAliasTable
       As String
```

パラメータ	説明
hOutline	EsbOtlOpenOutlineQuery 呼出しから戻される Esbbase アウトライン・ハンドル。
pszDTSMember	別名を提供する DTS メンバー名。
pzsAliasTable	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルを使用します。

備考

この関数は別名のみを消去します。DTS メンバーは使用不可にしません (EsbOtlEnableDTSMember を参照)。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_NOALIAS

例

```
    Public Sub ESB_OtlDeleteDTSMemberAlias()
 Dim DTSMember As String * ESB_MBRNAMELEN
 Dim AliasTable As String * ESB_ALIASNAMELEN

 DTSMember = "H-T-D"
 AliasTable = "default"

 sts = EsbOtlDeleteDTSMemberAlias(hOutline, _
               DTSMember, AliasTable)
End Sub
```

関連トピック

- EsbOtlEnableDTSMember
- EsbOtlGetEnabledDTSMembers
- EsbOtlGetDTSMemberAlias
- EsbOtlSetDTSMemberAlias

EsbOtlDeleteGenName

次元内の特定の世代名を削除します。世代名は、EsbOtlSetGenName を使用してアウトラインに明示的に追加されます。

構文

```
      EsbOtlDeleteGenName
       (
      hOutline, pszDimension, usGen
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      usGen
          As Integer
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象の世代を含む次元の名前。
usGen	名前を削除する世代の番号。リーフ・メンバーはレベル 0 です。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_NO_GENLEVELNAME
- OTLAPI_ERR_NOTADIM

例

```
    Declare Function EsbOtlDeleteGenName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension
As String, ByVal usGen As Integer) As Long

Sub ESB_OtlDeleteGenName()
Dim sts As Long
Dim Dimension As String
Dim GenNum As Integer
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
```

```
ESB_YES, hOutline)
'*********************************
'******* Delete Generation Name ***
'*********************************
Dimension = "Year"
GenNum = 2
GenName = "Qtr1 Qtr2 Qtr3 Qtr4"
If sts = 0 Then
  sts = EsbOtlDeleteGenName(hOutline,
   Dimension, GenNum)
End If
End Sub
```

関連トピック

- EsbOtlGetGenName
- EsbOtlGetGenNames
- EsbOtlSetGenName

EsbOtlDeleteLevelName

次元内の特定のレベルの名前を削除します。レベル名は EsbOtlSetLevelName でアウトラインに明示的に追加されます。

構文

```
      EsbOtlDeleteLevelName
       (
      hOutline, pszDimension, usLevel
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      usLevel
         As Integer
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象のレベル名を含む次元の名前。
usLevel	名前を削除するレベルの番号。リーフ・メンバーはレベル 0 です。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_NO_GENLEVELNAME

- OTLAPI_ERR_NOTADIM

例

```
    Declare Function EsbOtlDeleteLevelName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension
As String, ByVal usLevel As Integer) As Long

Sub ESB_OtlDeleteLevelName()
Dim sts As Long
Dim Dimension As String
Dim LevelNum As Integer
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
'***********************************
'******* Delete Level Name **********
'***********************************
Dimension = "Year"
LevelNum = 1
LevelName = "Month"
If sts = 0 Then
  sts = EsbOtlDeleteLevelName(hOutline,
   Dimension, LevelNum)
End If
End Sub
```

関連トピック

- EsbOtlGetLevelName
- EsbOtlGetLevelNames
- EsbOtlSetLevelName

EsbOtlDeleteMember

アウトラインからメンバーを削除します。

構文

```
      EsbOtlDeleteMember
       (
      hOutline, hMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
```

```
        As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　削除するメンバーのハンドル。

備考

- メンバーのすべての子孫が削除されます。
- メンバーおよびその子孫のすべての共有メンバーが削除されます。
- 共有メンバーの場合、指定されたメンバーのみ削除されます。
- 次元を削除するには、この呼出しまたは **EsbOtlDeleteDimension()**を使用します。**EsbOtlDeleteDimension()**では、データベースの再構築時に他の次元に使用されるデータ値を含むメンバーを、削除された次元から選択できるという利点があります。**EsbOtlDeleteMember()**を使用する場合、削除された次元の上位メンバー(次元)のデータ値が使用されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NOTIMEDIM

例

```
    Declare Function EsbOtlDeleteMember Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal
hMember As Long) As Long

Sub ESB_OtlDeleteMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hCOGS As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "COGS", hCOGS)
End If
If sts = 0 And hCOGS <> 0 Then
 sts = EsbOtlDeleteMember(hOutline, hCOGS)
End If
End Sub
```

関連トピック

- EsbOtlDeleteDimension
- EsbOtlAddMember
- EsbOtlAddDimension
- EsbOtlFindMember
- EsbOtlGetMemberInfo

EsbOtlDeleteMemberAlias

指定された別名テーブルの指定されたメンバーに対する、デフォルトのメンバー別名を削除します。

構文

```
      EsbOtlDeleteMemberAlias
       (
      hOutline, hMember, pszAliasTable
      )
ByVal
      hOutline
         As Long
ByVal
      hMember
          As Long
ByVal
      pszAliasTable
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名を削除するメンバーのハンドル。
pszAliasTable	別名を削除する別名テーブル。このパラメータが""の場合、デフォルトのテーブルが使用されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

OTLAPI_ERR_NOALIAS

例

```
    Declare Function EsbOtlDeleteMemberAlias Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszAliasTable As String) As Long

Sub ESB_OtlDeleteMemberAlias()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
```

```
Dim hOutline As Long
Dim hMemberYear As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Year", hMemberYear)
End If
If sts = 0 And hMemberYear <> 0 Then
 sts = EsbOtlDeleteMemberAlias(hOutline, hMemberYear, "")
End If
End Sub
```

関連トピック

- EsbOtlGetMemberAlias
- EsbOtlSetMemberAlias

EsbOtlDeleteMemberFormula

指定されたメンバーの式を削除します。

構文

```
      EsbOtlDeleteMemberFormula
       (
      hOutline, hMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　メンバーのハンドル。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

OTLAPI_ERR_NOFORMULA

例

```
    Declare Function EsbOtlDeleteMemberFormula Lib
"ESBOTLN" (ByVal hOutline As Long,
ByVal hMember As Long) As Long

Sub ESB_OtlDeleteMemberFormula()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberProfit As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Profit", hMemberProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
 sts = EsbOtlDeleteMemberFormula(hOutline, hMemberProfit)
End If
End Sub
```

関連トピック

- EsbOtlSetMemberFormula
- EsbOtlDeleteMemberFormula

EsbOtlDeleteUserAttribute

メンバーのユーザー定義属性を削除します。

構文

```
      EsbOtlDeleteUserAttribute
       (
      hOutline, hMember, pszString
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
        As Long
ByVal
      pszString
       As String
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル

hMember　削除対象の属性を持つメンバーのハンドル。

pszString　ユーザー属性の文字列。

備考

呼出し元は、属性を識別するために文字列で値を渡します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_NO_USERATTR。

例

```
    Declare Function EsbOtlDeleteUserAttribute Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszString As String) As Long

Sub ESB_OtlDeleteUserAttribute()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMember As Long
Dim AttributeList As String
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
AttributeList = "Read Write"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Jan",
  hMember)
End If
If sts = 0 And hMember <> 0 Then
'********************
' Delete User Attributes
'********************
 sts = EsbOtlDeleteUserAttribute(hOutline,
  hMember, AttributeList)
 End If
End Sub
```

関連トピック

- EsbOtlGetUserAttributes
- EsbOtlSetUserAttribute

EsbOtlDisassociateAttributeDimension

属性次元と基本次元との関連付けを解除します。

構文

```
      EsbOtlDisassociateAttributeDimension
       (
      hOutline
      ,
      BaseDimension
      ,
      AttributeDimension
      )
ByVal
      hOutline
           As Long
ByVal
      BaseDimension
         As Long
ByVal
      AttributeDimension
       As Long
```

パラメータ	説明
hOutline	アウトラインのハンドル
BaseDimension	基本次元のハンドル
AttributeDimension	属性次元のハンドル

備考

- 属性次元の基本次元との関連付けを解除すると、属性次元のすべてのメンバーと基本次元メンバーとの関連付けが解除されます。
- 関連付けを解除された属性次元は、確認されてディスクに書き込まれるときにアウトラインに保持されません。この状況に対処するために推奨されるのは、関連付けを解除された次元をアウトラインから削除することです。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
        Sub ESB_OtlDisAssociateAttributeDimension()
       ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text
box
       Dim sts as long
       Dim hOutline as long
```

```
Dim BaseMbr As Long
Dim AttrMbr As Long
hOutline = ESB_OtlOpenOutline
If hOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub
BaseMbr = ESB_OtlFindMember("Enter base dimension: ")
If BaseMbr = vbNull Then MsgBox "ESB_OtlDisAssociateAttributeDimension() failed.":
Exit Sub
       AttrMbr = ESB_OtlFindMember("Enter attribute dimension: ")
       If AttrMbr = vbNull Then MsgBox "ESB_OtlDisAssociateAttributeDimension()
failed.": Exit Sub
       sts = EsbOtlDisassociateAttributeDimension(ghOutline, BaseMbr, AttrMbr)
       sts = EsbOtlDeleteDimension(ghOutline, AttrMbr, "")
       If sts <> 0 Then
              Out "EsbOtlDeleteDimension failed" & sts: Exit Sub
       Else
              Out "EsbOtlDeleteDimension succeeded: " & sts
       End If
' abstract sub to call EsbOtlVerifyOutline(), ESBOTLNriteOutline(),
EsbOtlRestructure(),EsbUnlockObject() and
' EsbOtlCloseOutline() as neededà
tuckinoutline
       If sts <> 0 Then Out "EsbOtlDisassociateAttributeDimension failed: " & sts:
Exit Sub
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlDisassociateAttributeMember

属性メンバーと基本メンバーとの関連付けを解除します。

構文

```
EsbOtlDisassociateAttributeMember
 (
hOutline
,
BaseMember
```

```
      ,
      AttributeMember
      )
ByVal
      hOutline
          As Long
ByVal
      BaseMember
         As Long
ByVal
      AttributeMember
       As Long
```

パラメータ	説明
hOutline	アウトラインのハンドル
BaseMember	基本メンバーのハンドル
AttributeMember	属性メンバーのハンドル

備考

属性次元の基本次元との関連付けを解除すると、属性次元のすべてのメンバーと基本次元メンバーとの関連付けが解除されます。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlDisassociateAttributeMember()
Dim BaseMbr As Long
Dim AttributeMbr As Long
Dim sts as long
Dim hOutline as long
hOutline = ESB_OtlOpenOutline
       If hOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub
BaseMbr = ESB_OtlFindMember("Enter base member: ")
If BaseMbr = vbNull Then
Out "ESB_OtlGetMemberInfo() failed in ESB_OtlFindMember. " & sts: Exit Sub
AttributeMbr = ESB_OtlFindMember("Enter attribute member: ")
If AttributeMbr = vbNull Then Out "ESB_OtlGetMemberInfo() failed in
ESB_OtlFindMember. " & sts: Exit Sub
       sts = EsbOtlDisassociateAttributeMember(hOutline, BaseMbr, AttributeMbr)
If sts = 0 Then Out "EsbOtlDisassociateAttributeMember failed " & sts: Exit Sub
       sts = EsbOtlDeleteMember(ghOutline, AttrMbr)
If sts <> 0 Then Out "EsbOtlDeleteMember failed" & sts: Exit Sub
' abstract sub to call EsbOtlVerifyOutline(), ESBOTLNriteOutline(),
EsbOtlRestructure(),EsbUnlockObject() and
' EsbOtlCloseOutline() as neededà
```

```
tuckinoutline
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlEnableDTSMember

アウトラインに対して新規 DTS メンバーを使用可能にします。

構文

```
      EsbOtlEnableDTSMember
       (
      hOutline, pszDTSMember, usGen, bEnable
      )
ByVal
      hOutline
        As Long
ByVal
      pszDTSMember
       As String
ByVal
      usGen
          As Integer
ByVal
      bEnable
         As Integer
```

パラメータ	説明
hOutline	EsbOtlOpenOutline 関数から戻された Essbase アウトライン・ハンドル。
pszDTSMember	使用可能にする DTS メンバー名を含む文字列。
usGen	DTS メンバーを使用可能にする世代番号。
bEnable	フラグ。TRUE の場合メンバーを使用可能にし、FALSE の場合メンバーを使用不可にします。

備考

この関数は、自分に渡された ESB_DTSMBRNAME_T 構造体にも値を入れます。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、**EsbOtlQueryMembers()**呼出しのステータスを戻します。

例

```
    Public Sub ESB_OtlEnableDTSMember()
 Dim DTSMember As String
 Dim GenNum As Integer
 Dim Enable As Integer

 DTSMember = "H-T-D"
 GenNum = 1
 Enable = ESB_TRUE

 sts = EsbOtlEnableDTSMember(hOutline, DTSMember, _
             GenNum, Enable)
End Sub
```

関連トピック

- EsbOtlDeleteDTSMemberAlias
- EsbOtlGetEnabledDTSMembers
- EsbOtlGetDTSMemberAlias
- EsbOtlSetDTSMemberAlias

EsbOtlFindAlias

指定した別名を持つメンバーを検索し、そのメンバーにハンドルを戻します。

構文

```
     EsbOtlFindAlias
      (
     hOutline, pszAlias, pszAliasTable, phMember
     )
ByVal
     hOutline
        As Long
ByVal
     pszAlias
        As String
ByVal
     pszAliasTable
      As String

     phMember
        As Long
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAlias	検索対象の別名。
pszAliasTable	検索する別名テーブル。すべての別名テーブルを検索するには、""を使用します。デフォルトの別名テーブルを検索するには、"Default"を使用します。
phMember	メンバー・ハンドルの戻り変数。メンバーが見つからなかった場合は ESB_NULL です。

備考

- 別名の組合せで使用されている別名も検索されます。
- メンバーが見つからない場合、phMember が""に設定され、呼出しから 0 が戻されます。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlFindAlias Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszAlias
As String, ByVal pszAliasTable
As String, phMember As Long) As Long
Sub ESB_OtlFindAlias()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberAlias As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
  sts = EsbOtlFindAlias(hOutline, "Root Beer", "", hMemberAlias)
End If
End Sub
```

関連トピック

- EsbOtlGetOutlineInfo
- EsbOtlGetMemberAlias

EsbOtlFindAttributeMembers

指定されたショート名を持つすべての属性メンバーを戻します。

構文

EsbOtlFindAttributeMembers

```
      (
      hOutline
      ,
      MemberName
      ,
      DimensionName
      ,
      Count
      ,
      MemberArray
      )
ByVal
      hOutline
         As Long
ByVal
      MemberName
        As String
ByVal
      DimensionName
       As String

      Count
          As Integer

      MemberArray
        As Variant
```

パラメータ	説明
hOutline	アウトラインのハンドル
MemberName	属性のショート名
DimenisonName	属性次元名(オプション)
Count	戻されたメンバーの数
MemberArray	基本メンバーのハンドルの配列

備考

- MemberName は、ショート名である必要があります。
- DimensionName はオプションです。NULL を入力できます。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlFindAttributeMembers()
   Dim MemberName As String
```

```
    Dim DimensionName As String
    Dim hMember() As Long
    Dim Count As Integer
    Dim MbrArr As Variant
    Dim MbrInfo As ESB_MBRINFO_T
    Dim index As Integer
    ghOutline = ESB_OtlOpenOutline
    If ghOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub
    ' expecting return of handle to "caffeinated_true"
    MemberName = "true"
    ' "null" by default - dimension name is optional
    DimensionName = ""
    sts = EsbOtlFindAttributeMembers(ghOutline, MemberName, DimensionName, Count,
MbrArr)
    ' sts = EsbOtlFindAttributeMembers(ghOutline, MemberName, Count, MbrArr)
    If sts = 0 Then
        Out "EsbOtlFindAttributeMembers passed " & sts
        Out "Count is : " & Count
        For index = 0 To Count - 1
        sts = EsbOtlGetMemberInfo(ghOutline, MbrArr(index), MbrInfo)
        Out "Member Name : " & MbrInfo.szMember
        Next index
    Else
        Out "EsbOtlFindAttributeMembers failed " & sts
        Exit Sub
    End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlFindMember

指定した名前を持つメンバーを検索し、そのメンバーにハンドルを戻します。

構文

```
EsbOtlFindMember
 (
```

```
     hOutline, pszMember, phMember
     )
ByVal
     hOutline
      As Long
ByVal
     pszMember
      As String

     phMember
      As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

pszMember　検索対象のメンバー名。

phMember　メンバー・ハンドルの戻り変数。メンバーが見つからなかった場合は ESB_NULL です。

備考

- 検索対象のメンバーが共有メンバーを持っている場合、実メンバーへのハンドルのみが戻されます。ハンドルを持っている場合、**EsbOtlGetNextSharedMember()**を使用して共有メンバー情報を取得します。
- メンバーが見つからない場合、phMember が ESB_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlFindMember Lib
"ESBOTLN" (ByVal hOutline As Long,
ByVal pszMember As String, phMember As Long) As Long
Sub ESB_OtlFindMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
Dim hMemberProfit As Long
Dim hNewMember As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit",
  hMemberProfit)
End If
```

```
End Sub
```

関連トピック

- EsbOtlMoveMember
- EsbOtlRenameMember
- EsbOtlAddMember
- EsbOtlDeleteMember
- EsbOtlGetNextSharedMember

EsbOtlFreeMember

EsbGetNextItem が呼び出されたときに EsbOtlQueryMembers から戻されたメンバーを解放します。

構文

```
      EsbOtlFreeMember
       (
      hOutline, hMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long
```

パラメータ 説明

hOutline　EsbOtlOpenOutlineQuery()から戻された Essbase アウトライン・ハンドル。

hMember　解放するメンバーを定義しているメンバーのハンドル。

備考

EsbOtlQueryMembers()からの結果は、**EsbGetNextItem()**の呼出しを介して一度に 1 つのメンバーを戻します。これらの使用されるアイテムがそれぞれ実行された場合、プログラマは **EsbOtlFreeMember()**を呼び出す必要があります。

戻り値

正常終了の場合は 0 が戻されます。

例

```
    Declare Function EsbOtlFreeMember Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long) As Long
Declare Function EsbOtlQueryMembers Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long,
pPredicate As ESB_PREDICATE_T, pCounts As ESB_MBRCOUNTS_T) As Long

Sub ESB_OtlQueryMembers()
```

```
 Dim sts As Long
 Dim hOutline As Long
 Dim hMember As Long
 Dim ihMember As Long
 Dim Object As ESB_OBJDEF_T
 Dim MbrInfo As ESB_MBRINFO_T
 Dim Predicate As ESB_PREDICATE_T
 Dim Counts As ESB_MBRCOUNTS_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 If sts = 0 Then
  sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 Predicate.ulQuery = ESB_CHILDREN
 Predicate.pszDimension = "Year"
 Counts.ulStart = 0
 Counts.ulMaxCount = 10
 If sts = 0 Then
   sts = EsbOtlQueryMembers(hOutline, hMember, Predicate, Counts)
  If sts = 0 And Counts.ulReturnCount <> 0 Then
   For n% = 1 To Counts.ulReturnCount
     sts = EsbGetNextItem(hCtx, ESB_HMEMBER_TYPE, ihMember)
   If sts = 0 And ihMember <> 0 Then
       sts = EsbOtlFreeMember(hOutline, ihMember)
   End If
  Next
  End If
 End If
End If
End Sub
```

関連トピック

- EsbOtlOpenOutlineQuery
- EsbOtlQueryMembers
- EsbOtlQueryMembersByName

EsbOtlGenerateCurrencyOutline

既存のアウトラインを基に通貨アウトラインを生成します。

構文

```
      EsbOtlGenerateCurrencyOutline
       (
      hOutline, phCurOutline
      )
ByVal
      hOutline
        As Long
```

```
    phCurOutline
     As Long
```

パラメータ　説明

hOutline　アウトラインのコンテキスト・ハンドル。

phCurOutline 通貨アウトラインのハンドルに対する変数を戻します。

備考

- ソースのアウトラインには、時間次元、会計次元および国次元が存在している必要があります。
- 時間次元とすべての子孫は、ソースのアウトラインから新規アウトラインの時間次元に直接コピーされます。
- CurCategory(Dense, Category = Accounts)という名前の次元が、新規アウトラインに作成されます。ソース・アカウント次元内のすべての通貨カテゴリが、新規アウトラインの CurCategory 次元の子になります。
- CurName(Dense, Category = Country)という名前の次元が、新規アウトラインに作成されます。ソースの国次元のすべての通貨名が、新規アウトラインの CurName 次元の子になります。
- CurType(Sparse, Category = Type)という名前の次元が、新規アウトラインに子なしで作成されます。
- 通貨アウトラインは、**ESBOTLNriteOutline()**、**EsbOtlRestructure()**の順に呼び出して保存し、**EsbOtlCloseOutline()**を呼び出して閉じる必要があります。
- 新規アウトラインには、次の属性があります:
  - 自動構成は、ESB_TRUE に設定されています
  - 大文字と小文字の区別は、元のアウトラインと同様に設定されています

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_ALREADYCURRENCY
- OTLAPI_CUR_NOACCOUNTS
- OTLAPI_CUR_NOTIME
- OTLAPI_CUR_NOCOUNTRY

例

```
    Declare Function EsbOtlGenerateCurrencyOutline Lib
"ESBOTLN" (ByVal hOutline As Long,
phCurOutline As Long) As Long
Sub ESB_OtlGenerateCurrencyOutline()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
```

```
Dim hCurOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Interntl"
Object.FileName = " Interntl "
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlGenerateCurrencyOutline
  (hOutline, hCurOutline)
End If
End Sub
```

関連トピック

- EsbOtlOpenOutline
- EsbOtlWriteOutline
- EsbOtlRestructure

EsbOtlGetAliasTableLanguages

指定した別名テーブルに関連付けられた言語コードの数が戻され、**EsbGetNextItem()** からアクセス可能な別名テーブルの文字列のリストが生成されます。

構文

```
      ESB_FUNC_M
      EsbOtlGetAliasTableLanguages
       (
      hOutline
      ,
      pszAliasTable
      ,
      pItems
      )
ByVal
      hOutline
          As Long
ByVal
      pszAliasTable
        As String

      pItems
           As Long
```

パラメータ	説明
hOutline	アウトラインのハンドル。
pszAliasTable	関連付けられた言語コードを取得する別名テーブル名。
pItems	別名テーブルに関連付けられた言語コードの数が戻される変数のアドレス。

戻り値

- 正常終了の場合は、別名テーブルの言語の数が pItems に戻され、**EsbGetNextItem**()からアクセス可能な別名テーブルの文字列のリストが生成されます。
- 処理に失敗すると、エラー OTLAPI_BAD_ALIASTABLE (無効な別名テーブル)が戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbOtlGetAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String, pulCount As Long) As Long
Declare Function EsbOtlSetAliasTableLanguage Lib "esbotln" (ByVal hOutline As Long,
ByVal pszAliasTable As String, ByVal pszLanguageCode As String) As Long
Declare Function EsbOtlClearAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String) As Long

Sub ESB_Sub ()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Items As Long
Dim AliasLang As String * ESB_ALIASNAMELEN

Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlCreateAliasTable(hOutline,
  "French Alias Table")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr-CA")
End If

If sts = 0 Then
 sts = EsbOtlGetAliasTableLanguages(hOutline,
  "French Alias Table", Items)
 If sts = 0 Then
    For N = 1 To Items
          sts = EsbGetNextItem(hCtx, ESB_ALIASLANG_TYPE, ByVal AliasLang)
     Next
 End If
```

```
End If

If sts = 0 Then
 sts = EsbOtlClearAliasTableLanguages(hOutline,
  "French Alias Table")
End If

End Sub
```

関連トピック

- EsbOtlClearAliasTableLanguages
- EsbOtlSetAliasTableLanguage

EsbOtlGetAssociatedAttributes

基本メンバーまたは基本次元に関連付けられているすべての属性メンバーを戻します。

構文

```
      EsbOtlGetAssociatedAttributes
       (
      hOutline
      ,
      Member
      ,
      Count
      ,
      MemberArray
      )
ByVal
      hOutline
        As Long
ByVal
      Member
         As Long

      Count
          As Integer

      MemberArray
       As Variant
```

パラメータ　説明

hOutline	アウトラインのハンドル
Member	基本メンバーまたは基本次元のハンドル
Count	戻された属性メンバー数
MemberArray	属性メンバーのハンドルの配列

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlGetAssociatedAttributes()
  Dim hMember As Long
  Dim Count As Integer            '*** Return of number of attributes
  Dim MbrArr As Variant           '*** Returns member array in this
  Dim MbrInfo As ESB_MBRINFO_T    '*** Returned MbrInfo structure
  Dim index As Integer
  eraser
  hMember = ESB_OtlFindMember("Enter target member: ")
  If hMember = vbNull Then Out "ESB_OtlGetAssociatedAttributes() failed.": Exit Sub
  sts = EsbOtlGetAssociatedAttributes(ghOutline, hMember, Count, MbrArr)
  If sts = 0 Then
    Out "Count is : " & Count
    For index = 0 To (Count - 1)
      sts = EsbOtlGetMemberInfo(ghOutline, MbrArr(index), MbrInfo)
      Out "Member Name : " & MbrInfo.szMember
    Next index
  Else
    Out "EsbOtlGetAttributeInfo failed" & sts: Exit Sub
  End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlGetAttributeInfo

指定した属性メンバーまたは属性次元に関する属性情報を戻します。

構文

```
      EsbOtlGetAttributeInfo
       (
      hOutline
      ,
      Member
      ,
      AttrInfo
      )
ByVal
      hOutline
       As Long
ByVal
      Member
        As Long

      AttrInfo
       As ESB_ATTRIBUTEINFO_T
```

パラメータ 説明

hOutline　アウトラインのハンドル

Member　属性メンバーまたは属性次元のハンドル

AttrInfo　属性情報

備考

この関数は **EsbGetAttributeInfo()** と同様のものです。

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlGetAttributeInfo()
    ' NOTE: 'Out' is a sub to print the output within quotes to a listbox or text box
Dim sts As Long
Dim OutAttrInfo As ESB_ATTRIBUTEINFO_T
Dim MbrName As String
Dim hCtx as long
    MbrName = InputBox("Enter Member Name")
    sts = EsbGetAttributeInfo(hCtx, MbrName, OutAttrInfo)
    If sts = 0 Then
        Select Case VarType(OutAttrInfo.Attribute)
        Case vbDouble
            Out "Data Type   : Numeric(Double)"
            Out "Data Value  : " & OutAttrInfo.Attribute
            Out ""
```

```
        Case vbBoolean
            Out "Data Type   : Boolean"
            Out "Data Value  : " & OutAttrInfo.Attribute
            Out ""
        Case vbDate
            Out "Data Type   : Date"
            Out "Data Value  : " & OutAttrInfo.DimName
            Out ""
        Case vbString
            Out "Data Type   : String"
            Out "Data Value  : " & OutAttrInfo.Attribute
            Out ""
        End Select
    Else
        Out "ESB_OtlGetAttributeInfo failed" & sts
        Exit Sub
    End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlGetAttributeSpecifications

アウトラインの属性指定を取得します。

構文

```
      EsbOtlGetAttributeSpecifications
       (
      hOutline
      ,
      AttrSpecs
      )
ByVal
      hOutline
       As Long

      AttrSpecs
```

```
As ESB_ATTRSPECS_T
```

パラメータ 説明

hOutline　アウトラインのハンドル

AttrSpecs　属性指定

備考

- この関数は、開かれたアウトラインから情報を戻すということを除けば、**EsbGetAttributeSpecifications()**と同様の関数です。
- アウトラインの属性指定を設定するには、**EsbOtlSetAttributeSpecifications()**を使用します。
- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供

戻り値

正常終了の場合は STS = 0 が戻されます。それ以外の場合、エラー・コードが戻されます。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
 Sub ESB_OtlGetAttributeSpecifications()
Dim OutAttrSpecs As ESB_ATTRSPECS_T
Dim test As String
Dim sts as long
hOutline = ESB_OtlOpenOutline
If hOutline = vbNull Then Out "ESB_OtlOpenOutline() failed: " & sts: Exit Sub
sts = EsbOtlGetAttributeSpecifications(hOutline, OutAttrSpecs)
If sts <> 0 Then Out "ESB_OtlGetAttributeSpecifications failed" & sts: Exit Sub
Out "ESB_OtlGetAttributeSpecifications passed: " & sts
Out "DefaultTrueString : " & OutAttrSpecs.DefaultTrueString
Out "DefaultFalseString : " & OutAttrSpecs.DefaultFalseString
Out "DefaultAttrCalcDimName : " & OutAttrSpecs.DefaultAttrCalcDimName
Out "DefaultSumMbrName : " & OutAttrSpecs.DefaultSumMbrName
Out "DefaultCountMbrName : " & OutAttrSpecs.DefaultCountMbrName
Out "DefaultAverageMbrName : " & OutAttrSpecs.DefaultAverageMbrName
Out "DefaultMinMbrName : " & OutAttrSpecs.DefaultMinMbrName
Out "DefaultMaxMbrName : " & OutAttrSpecs.DefaultMaxMbrName
test = OutAttrSpecs.GenNameBy
Select Case test
    Case ESB_GENNAMEBY_PREFIX
        Out "GenNameBy : ESB_GENNAMEBY_PREFIX"
    Case ESB_GENNAMEBY_SUFFIX
        Out "GenNameBy : ESB_GENNAMEBY_SUFFIX"
```

```
        Case Else
            Out "GenNameBy : invalid"
    End Select
    test = OutAttrSpecs.UseNameOf
    Select Case test
        Case ESB_USENAMEOF_NONE
            Out "UseNameOf : ESB_USENAMEOF_NONE"
        Case ESB_USENAMEOF_PARENT
            Out "UseNameOf : ESB_USENAMEOF_PARENT"
        Case ESB_USENAMEOF_GRANDPARENTANDPARENT
            Out "UseNameOf : ESB_USENAMEOF_GRANDPARENTANDPARENT"
        Case ESB_USENAMEOF_ALLANCESTORS
            Out "UseNameOf : ESB_USENAMEOF_ALLANCESTORS"
        Case ESB_USENAMEOF_DIMENSION
            Out "UseNameOf : ESB_USENAMEOF_DIMENSION"
        Case Else
            Out "UseNameOf : invalid"
    End Select
    test = OutAttrSpecs.Delimiter
    Select Case test
        Case ESB_DELIMITER_UNDERSCORE
            Out "Delimiter : ESB_DELIMITER_UNDERSCORE"
        Case ESB_DELIMITER_PIPE
            Out "Delimiter : ESB_DELIMITER_PIPE"
        Case ESB_DELIMITER_CARET
            Out "Delimiter : ESB_DELIMITER_CARET"
        Case Else
            Out "Delimiter : invalid"
    End Select
    test = OutAttrSpecs.DateFormat
    Select Case test
        Case ESB_DATEFORMAT_MMDDYYYY
            Out "DateFormat : ESB_DATEFORMAT_MMDDYYYY"
        Case ESB_DATEFORMAT_DDMMYYYY
            Out "DateFormat : ESB_DATEFORMAT_DDMMYYYY"
        Case Else
            Out "Delimiter : invalid"
    End Select
    test = OutAttrSpecs.BucketingType
    Select Case test
        Case ESB_UPPERBOUNDINCLUSIVE
            Out "BucketingType : ESB_UPPERBOUNDINCLUSIVE"
        Case ESB_LOWERBOUNDINCLUSIVE
            Out "BucketingType : ESB_ESB_LOWERBOUNDINCLUSIVE"
        Case ESB_UPPERBOUNDNONINCLUSIVE
            Out "BucketingType : ESB_UPPERBOUNDNONINCLUSIVE"
        Case ESB_LOWERBOUNDNONINCLUSIVE
            Out "BucketingType : ESB_LOWERBOUNDNONINCLUSIVE"
        Case Else
            Out "BucketingType : invalid"
    End Select
End Sub
```

関連トピック

- EsbCheckAttributes

- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlQueryAttributes
- EsbOtlSetAttributeSpecifications

EsbOtlGetChild

メンバーの子を戻します。

構文

```
      EsbOtlGetChild
       (
      hOutline, hMember, phMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long

      phMember
       As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　子を取得するメンバーのハンドル。

phMember　hMember パラメータの子のハンドルに対する変数を戻します。

備考

- 子がない場合、*phMember は ESB_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetChild Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
```

```
phMember As Long) As Long
Sub ESB_OtlGetChild()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberParent As Long
Dim hMemberChild As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline,
  "Year", hMemberParent)
End If
If sts = 0 And hMemberParent <> 0 Then
 sts = EsbOtlGetChild(hOutline,
  hMemberParent, hMemberChild)
End If
End Sub
```

関連トピック

- EsbOtlGetParent
- EsbOtlGetNextSibling
- EsbOtlGetPrevSibling
- EsbOtlGetFirstMember

EsbOtlGetDimensionUserAttributes

指定された次元で使用されるユーザー定義属性を戻します。

構文

```
      EsbOtlGetDimensionUserAttributes (
      hOutline, pPredicate, pCounts
      )
ByVal
      hOutline
        As Long

      pPredicate
       As ESB_PREDICATE_T

      pCounts
        As ESB_MBRCOUNTS_T
```

パラメータ 説明

hOutline　Essbase アウトライン・ハンドル。**EsbOtlOpenOutlineQuery()**から戻されている必要があります。

pPredicate クエリーを定義している構造体。この構造体のフィールドの用法:

- **ulQuery** – 実行する操作を定義する値。ESB_DIMUSERATTRIBUTES のみが有効な値です。
- **szDimension** – クエリーの範囲を制限する次元。有効な次元名を指定します。

pCounts　カウントに関する情報を定義している構造体。次のフィールドが含まれます:

- **ulStart**– 戻す最初の番号
- **ulMaxCount**– 戻すメンバー名の最大数。
- **ulTotalCount**– クエリーの実行結果において定義されるメンバーの合計数。
- **pulReturnCount**– このクエリーで戻されたメンバー名の数。

備考

この関数は、特定の次元に対してユーザーが定義した属性を入手するためのみに使用します。したがって、述部に有効な唯一の値は ESB_DIMUSERATTRIBUTES_T です。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlGetDimensionUserAttributes Lib "ESBOTLN"
(ByVal hOutline As Long, pPredicate As ESB_PREDICATE_T,
pCounts As ESB_MBRCOUNTS_T) As Long

Sub ESB_OtlQueryMembers()
 Dim sts As Long
 Dim hOutline As Long
 Dim AttrName As String * ESB_MBRNAMELEN
 Dim Predicate As ESB_PREDICATE_T
 Dim Counts As ESB_MBRCOUNTS_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 If sts = 0 Then
  sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 Predicate.ulQuery = ESB_DIMUSERATTRIBUTES_T
 Predicate.pszDimension = "Product"
 Counts.ulStart = 0
 Counts.ulMaxCount = 10
 If sts = 0 Then
   sts = EsbOtlGetDimensionUserAttributes(hOutline, Predicate, Counts)
  If sts = 0 And Counts.ulReturnCount <> 0 Then
   For n% = 1 To Counts.ulReturnCount
     sts = EsbGetNextItem(hCtx, ESB_MBRNAME_TYPE, ByVal AttrName)
```

```
        MsgBox AttrName
    Next
   End If
  End If
End If
End Sub
```

関連トピック

- EsbGetNextItem
- EsbOtlOpenOutlineQuery
- EsbOtlQueryMembers
- EsbOtlQueryMembersByName

EsbOtlGetDTSMemberAlias

DTS メンバーの別名を取得します。

構文

```
      EsbOtlGetDTSMemberAlias
       (
      hOutline, pszDTSMember, pszAliasTable, ppszAlias
      )
ByVal
      hOutline
         As Long
ByVal
      pszDTSMember
       As String
ByVal
      pszAliasTable
       As String
ByVal
      pszAlias
         As String
```

パラメータ	説明
hOutline	EsbOtlOpenOutlineQuery 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember	別名を提供する DTS メンバー名。
pzsAliasTable	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルが使用されます。
ppszAlias	DTS メンバーの別名を含む C 文字列を指すポインタへのポインタ。

備考

固定長の ESB_ALIASNAMELEN は変数の別名に対して文字列長を設定します。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE

例

```
    Public Sub ESB_OtlGetDTSMemberAlias()
 Dim DTSMember As String * ESB_MBRNAMELEN
 Dim AliasTable As String * ESB_ALIASNAMELEN
 Dim Alias As String * ESB_ALIASNAMELEN

 DTSMember = "H-T-D"
 AliasTable = "Default"

 sts = EsbOtlGetDTSMemberAlias(hOutline, DTSMember, _
              AliasTable, Alias)
 MsgBox Alias

End Sub
```

関連トピック

- EsbOtlDeleteDTSMemberAlias
- EsbOtlEnableDTSMember
- EsbOtlGetEnabledDTSMembers
- EsbOtlSetDTSMemberAlias

EsbOtlGetEnabledDTSMembers

アウトラインに対して定義された DTS メンバーを取得します。

構文

```
      EsbOtlGetEnabledDTSMembers
       (
      hOutline, pusCount
      )
ByVal
      hOutline
       As Long

      pusCount
       As Integer
```

パラメータ 説明

hOutline　EsbOtlOpenOutlineQuery()呼出しから戻されたEssbaseアウトライン・ハンドル。

pusCount　定義されたDTSメンバーの数。

備考

呼出しが正常に行われると、**EsbGetNextItem** の呼出しは、Count に対して戻された値によって特定される使用可能な各 DTS メンバーに対して呼び出されます。

戻り値

成功の場合、戻り値はゼロです。そうでない場合は、**EsbOtlQueryMembers**()呼出しのステータスを戻します。

例

```
    Public Sub ESB_OtlGetEnabledDTSMembers()
 Dim Count As Integer
 Dim DTSMbr As String * ESB_MBRNAMELEN
 Dim i As Integer

 sts = EsbOtlGetEnabledDTSMembers(hOutline, Count)
 If sts = 0 Then
  For i = 1 To Count
   sts = EsbGetNextItem(hCtx, ESB_DTS_TYPE, ByVal DTSMbr)
   MsgBox "DTSMbr"
  Next i
 End If

End Sub
```

関連トピック

- EsbOtlDeleteDTSMemberAlias
- EsbOtlEnableDTSMember
- EsbOtlGetDTSMemberAlias
- EsbOtlSetDTSMemberAlias

EsbOtlGetFirstMember

アウトラインの最初のメンバーのハンドルを戻します。最初のメンバーはアウトラインで最初に定義されている次元です。

構文

```
     EsbOtlGetFirstMember
      (
     hOutline, phMember
     )
ByVal
     hOutline
      As Long

     phMember
      As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

phMember　アウトラインの最初のメンバーのハンドルの変数。このパラメータはアウトラインをたどる後続の呼出しに渡されます。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetFirstMember Lib
"ESBOTLN" (ByVal hOutline As Long, phMember As Long) As Long

Sub ESB_OtlGetFirstMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberFirst As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlGetFirstMember
  (hOutline, hMemberFirst)
End If
End Sub
```

関連トピック

- EsbOtlGetParent
- EsbOtlGetNextSibling
- EsbOtlGetPrevSibling
- EsbOtlGetChild

EsbOtlGetGenName

次元内の特定の世代の名前を取得します。世代名は、EsbOtlSetGenName を使用してアウトラインに明示的に追加されます。

構文

```
     EsbOtlGetGenName
      (
     hOutline, pszDimension, usGen, pszName
     )
ByVal
     hOutline
```

```
        As Long
ByVal
      pszDimension
       As String
ByVal
      usGen
          As Integer
ByVal
      pszName
         As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象の世代名を含む次元の名前。
usGen	名前を取得する世代の番号。次元は世代 1 です。
ppszName	呼出し元が割り当てた世代名を戻すためのバッファ。バッファは有効なメンバー名(ESB_MBRNAMELEN)を保存するのに十分な大きさである必要があります。

備考

- 世代名はメンバー名と同じルールに従い、メンバー名全体で一意性が必要です。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- 世代名は自動的には付与されません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EsbOtlSetGenName で割り当てられます

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

- OTLAPI_NO_GENLEVELNAME
- OTLAPI_ERR_NOTADIM

例

```
    Declare Function EsbOtlGetGenName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension
As String, ByVal usGen As Integer, ByVal pszName
As String) As Long

Sub ESB_OtlGetGenName()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Dimension As String
Dim GenNum As Integer
Dim GenName As String * ESB_MBRNAMELEN
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
```

```
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
'*********************************************
'************** Get Gen Name *****************
'*********************************************
Dimension = "Year"
GenNum = 2
If Not sts Then
 sts = EsbOtlGetGenName(hOutline, Dimension,
  GenNum, GenName)
End If
End Sub
```

関連トピック

- EsbOtlGetGenNames
- EsbOtlDeleteGenName
- EsbOtlSetGenName

EsbOtlGetGenNames

特定の次元に対して指定されたすべての世代名を取得します。世代名は、EsbOtlSetGenName を使用してアウトラインに明示的に追加されます。

構文

```
      EsbOtlGetGenNames
      (
      hOutline, pszDimension, ulOptions, pulCount
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      ulOptions
        As Long

      pulCount
        As Long
```

パラメータ	説明
hOutline	Essbase アウトライン・ハンドル。
pszDimension	世代名を取得する次元。

パラメータ	説明
ulOptions	これは、次のいずれかの値にできます: ● ESB_GENLEV_ALL – デフォルトと実際の世代名が戻されます ● ESB_GENLEV_ACTUAL – 実際に定義された世代名のみが戻されます ● ESB_GENLEV_DEFAULT – すべてのデフォルト世代名が戻されます。これには、実際の名前がある世代のデフォルト名も含まれます。 ● ESB_GENLEV_NOACTUAL – デフォルト世代名が戻されます。これには、実際の名前がない世代のみが含まれます
pulCount	pNameArray に要素数が戻されます。指定したメンバーの世代名の数です。
pNameArray	指定された次元に対する世代名の構造体の配列。

備考

- 呼出し元は、pNameArray 構造体の使用後、**EsbFree()**を呼び出してこの構造体を解放する必要があります。
- プログラマは各世代名構造体が戻されるたびに **EsbGetNextItem()**を 1 度呼び出す必要があります。
- この呼出しは、**EsbOtlOpenOutline()**および **EsbOtlOpenOutlineQuery()**の両方に機能します。**EsbOtlOpenOutlineQuery()**呼出し中にサーバーから戻されるため、情報は両方に対してローカルに存在します。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlGetGenNames Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal pszDimension As String, ByVal ulOptions
As Long, pulCount As Long) As Long

Sub ESB_OtlGetNames()
 Dim sts As Long
 Dim hOutline As Long
 Dim Object As ESB_OBJDEF_T
 Dim Dimension As String
 Dim GenOpt As Long
 Dim Count As Long
 Dim pGenName As ESB_GENLEVELNAME_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbSetActive(hCtx, AppName, DbName, Access)
 If sts=0 Then
  sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
  '*************** Get Gen Names ******************
  Dimension = "Year"
  GenOtp = ESB_GENLEV_DEFAULT
  If sts = 0 Then
```

```
    sts = EsbOtlGetGenNames(hOutline, Dimension, GenOpt, Count)
    If sts = 0 And Count <> 0 Then
     For n% = 1 To Count
     sts = EsbGetNextItem(hCtx, ESB_GENLEVELNAME_TYPE, pGenName)
      Next
    End If
  End If
 End If
End Sub
```

関連トピック

- EsbGetNextItem
- EsbOtlGetGenName
- EsbOtlGetLevelName
- EsbOtlGetLevelNames
- EsbOtlOpenOutline
- EsbOtlOpenOutlineQuery

EsbOtlGetLevelName

次元内の特定のレベルの名前を取得します。レベル名は EsbOtlSetLevelName でアウトラインに明示的に追加されます。

構文

```
      EsbOtlGetLevelName
       (
      hOutline, pszDimension, usLevel, pszName
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      usLevel
         As Integer
ByVal
      pszName
         As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象の世代を含む次元の名前。
usLevel	名前を取得するレベル番号の番号。リーフ・メンバーはレベル 0 です。

パラメータ	説明
pszName	呼出し元によって割り当てられた、指定された次元のレベルの戻しのためのバッファ。バッファは有効なメンバー名(ESB_MBRNAMELEN)を保存するのに十分な大きさである必要があります。

備考

- 世代名はメンバー名と同じルールに従い、メンバー名全体で一意性が必要です。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- レベル名は自動的には割り当てられません。この関数で名前を戻すには、名前を割り当てておく必要があります。名前は EsbOtlSetLevelName で割り当てられます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。それ以外の場合のコマンドの戻り値は次のとおりです:

- OTLAPI_NO_GENLEVELNAME
- OTLAPI_ERR_NOTADIM

例

```
    Declare Function EsbOtlGetLevelName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension
As String, ByVal usLevel As Integer, ByVal pszName
As String) As Long

Sub ESB_OtlGetLevelName()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Dimension As String
Dim LevelNum As Integer
Dim LevelName As String * ESB_MBRNAMELEN
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
'***************************************
'********** Get Level Name ***************
'***************************************
Dimension = "Year"
LevelNum = 2
If Not sts Then
  sts = EsbOtlGetLevelName(hOutline,
  Dimension, LevelNum, LevelName)
End If
End Sub
```

関連トピック

- EsbOtlGetLevelNames
- EsbOtlDeleteLevelName
- EsbOtlSetLevelName

EsbOtlGetLevelNames

特定の次元に対して指定されたすべてのレベル名を取得します。レベル名は EsbOtlSetLevelName でアウトラインに明示的に追加されます。

構文

```
      EsbOtlGetLevelNames
      (
      hOutline, pszDimension, ulOptions, pulCount
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      ulOptions
        As Long

      pulCount
        As Long
```

パラメータ	説明
hOutline	Esbbase のアウトラインのハンドル。
pszDimension	レベル名を取得する次元。
ulOptions	これは、次のいずれかの値にできます: ● ESB_GENLEV_ALL – デフォルトおよび実際のレベル名が戻されます。 ● ESB_GENLEV_ACTUAL – 実際に定義されているレベル名のみが戻されます。 ● ESB_GENLEV_DEFAULT – すべてのデフォルト・レベル名が戻されます。これには、実際の名前があるレベルのデフォルト名も含まれます。 ● ESB_GENLEV_NOACTUAL – デフォルト・レベル名が戻されます。これには、実際のレベル名前がないレベルのみが含まれます。
pulCount	pNameArray に要素数が戻されます。指定したメンバーのレベル名の数です。
pulCount	指定した次元に対するレベル名の構造体の配列。

備考

- 呼出し元は、pNameArray 構造体の使用後、**EsbFree()**を呼び出してこの構造体を解放する必要があります。

- プログラマは各レベル名構造体が戻されると、**EsbGetNextItem()**を 1 度呼び出す必要があります。
- この呼出しは、**EsbOtlOpenOutline()**および **EsbOtlOpenOutlineQuery()**の両方に機能します。**EsbOtlOpenOutlineQuery()**呼出し中にサーバーから戻されるため、情報は両方に対してローカルに存在します。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlGetLevelNames Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal pszDimension As String, ByVal ulOptions
As Long, pulCount As Long) As Long

Sub ESB_OtlGetLevelNames()
 Dim sts As Long
 Dim hOutline As Long
 Dim Object As ESB_OBJDEF_T
 Dim Dimension As String
 Dim LevOpt As Long
 Dim Count As Long
 Dim pLevName As ESB_GENLEVELNAME_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbSetActive(hCtx, AppName, DbName, Access)
 If sts = 0 Then
 sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 '*************** Get Level Names ******************
 Dimension = "Year"
 LevOtp = ESB_GENLEV_DEFAULT
 If sts = 0 Then
  sts = EsbOtlGetLevelNames(hOutline, Dimension,
     LevOpt, Count)
   If sts = 0 And pCount <> 0 Then
    For n = 1 To Count
     sts = EsbGetNextItem(hCtx, ESB_GENLEVELNAME_TYPE, pLevName)
      Next
   End If
  End If
 End If
End Sub
```

関連トピック

- EsbOtlGetGenName
- EsbOtlGetGenNames
- EsbOtlGetLevelName
- EsbOtlOpenOutline
- EsbOtlOpenOutlineQuery

EsbOtlGetMemberAlias

指定された別名テーブルの指定されたメンバーに対する、デフォルトのメンバー別名を取得します。

構文

```
      EsbOtlGetMemberAlias
       (
      hOutline, hMember, pszAliasTable, pszAlias
      )
ByVal
      hOutline
         As Long
ByVal
      hMember
          As Long
ByVal
      pszAliasTable
       As String
ByVal
      pszAlias
         As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名を取得するメンバーのハンドル。
pszAliasTable	別名を取得する別名テーブル。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszAlias	別名の戻り値が格納されるバッファ。バッファは呼出し元によって割り当てられます。

備考

pszAlias パラメータは呼出し元によって割り当てられるバッファで、少なくとも ESB_MBRNAMELEN バイトの大きさが必要です。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_ALIASTABLE

例

```
    Declare Function EsbOtlGetMemberAlias Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember
As Long, ByVal pszAliasTable As String, ByVal pszAlias
As String) As Long

Sub ESB_OtlGetMemberAlias()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
```

```
Dim hMemberProfit As Long
Dim szAlias As String * ESB_MBRNAMELEN
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
  sts = EsbOtlFindMember(hOutline,
   "Profit", hMemberProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
  sts = EsbOtlGetMemberAlias(hOutline,
   hMemberProfit, "Default", szAlias)
End If
End Sub
```

関連トピック

- EsbOtlSetMemberAlias
- EsbOtlDeleteMemberAlias

EsbOtlGetMemberFormula

指定されたメンバーの式を取得します。

構文

```
      EsbOtlGetMemberFormula
       (
      hOutline, hMember, pszFormula, usBufSize
      )
ByVal
      hOutline
        As Long
ByVal
      hMember
         As Long
ByVal
      pszFormula
       As String
ByVal
      usBufSize
       As Integer
```

パラメータ　説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　メンバーのハンドル。

パラメータ 説明

pszFormula メンバーの式の戻り変数。バッファは呼出し元によって割り当てられ、長さは usBufSize パラメータで指定されます。

usBufSize pszFormula バッファのサイズ。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetMemberFormula Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember
As Long, ByVal pszFormula As String, ByVal usBufSize
As Integer) As Long

Sub ESB_OtlGetMemberFormula()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberProfit As Long
Dim szFormula As String * 100
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit",
 hMemberProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
 sts = EsbOtlGetMemberFormula(hOutline,
 hMemberProfit, szFormula, 100)
End If
End Sub
```

関連トピック

- EsbOtlSetMemberFormula
- EsbOtlDeleteMemberFormula

EsbOtlGetMemberInfo

指定されたメンバーの情報を取得します。

構文

```
EsbOtlGetMemberInfo
 (
hOutline, hMember, pInfo
```

```
     )
ByVal
     hOutline
      As Long
ByVal
     hMember
      As Long

     pInfo
       As ESB_MBRINFO_T
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　メンバーのハンドル。

pInfo　メンバー情報の構造体の戻り変数。この構造体は呼出し元によって割り当てられます。

備考

- メンバーのハンドルは **EsbOtlFindMember()**を呼び出すことによって取得できます。
- 1612 ページの「ESB_MBRINFO_T」構造体の次の 2 つのフィールドは属性専用です:
  - Attribute
  - IsAttributed

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetMemberInfo Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember
As Long, pInfo As ESB_MBRINFO_T) As Long

Sub ESB_OtlGetMemberInfo()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
Dim hMemberProfit As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
  sts = EsbOtlFindMember(hOutline, "Profit",
  hMemberProfit)
End If
```

```
If sts = 0 And hMemberProfit <> 0 Then
 sts = EsbOtlGetMemberInfo(hOutline, hMemberProfit,
  MbrInfo)
End If
End Sub
```

関連トピック

- EsbOtlFindMember
- EsbOtlGetFirstMember

EsbOtlGetMemberLastFormula

メンバーの計算に使用された最後の式を戻します。

構文

```
      EsbOtlGetMemberLastFormula
       (
      hOutline, hMember, pszFormula, usBufSize
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
        As Long
ByVal
      pszFormula
       As String
ByVal
      usBufSize
       As Integer
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル

hMember　メンバーのハンドル。

pszFormula　メンバーの式の戻り変数。バッファは呼出し元によって割り当てられ、長さは usBufSize パラメータで指定されます。

usBufSize　pszFormula バッファのサイズ。

備考

- **EsbFree()**を使用して式のバッファを解放します。
- この呼出しは、**EsbOtlOpenOutline()**および **EsbOtlOpenOutlineQuery()**の両方に機能します。
- **EsbOtlGetMemberLastFormula()**は、選択したメンバーに最後に適用された式を戻しますが、これはそのメンバーに関連付けられているデータベース・アウトラインの式とは異なっている場合があります。

- 最後の式は、そのメンバーに対して最後に実行された計算(アウトラインまたは計算スクリプト)から導出されます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlGetMemberLastFormula Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long, ByVal pszFormula As String,
ByVal usBufSize As Integer) As Long

Sub ESB_OtlGetMemberLastFormula()
 Dim sts As Long
 Dim Object As ESB_OBJDEF_T
 Dim hOutline As Long
 Dim hMember As Long
 Dim szFormula As String * 100
 Object.hCtx = hCtx
 Object.Type = ESB_OBJTYPE_OUTLINE
 Object.AppName = "Sample"
 Object.DbName = "Basic"
 Object.FileName = "Basic"
 sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES, ESB_YES, hOutline)
 If sts = 0 Then
  sts = EsbOtlFindMember(hOutline, "Margin", hMember)
 End If
 If sts = 0 And hMember <> 0 Then
  sts = EsbOtlGetMemberLastFormula(hOutline,
     hMember, szFormula, 100)
 End If
End Sub
```

関連トピック

- EsbOtlDeleteMemberFormula
- EsbOtlGetMemberFormula
- EsbOtlOpenOutline
- EsbOtlOpenOutlineQuery
- EsbOtlSetMemberFormula

EsbOtlGetNextAliasCombination

指定された別名テーブルの指定されたメンバーに対する、別名の組合せを戻します。別名は pszAlias パラメータに戻され、メンバーの組合せは pszCombination に戻されます。

構文

```
     EsbOtlGetNextAliasCombination
      (
     hOutline, hMember, pszAliasTable, pszAlias, pszCombination,
usBufSize
```

```
     )
ByVal
     hOutline
        As Long
ByVal
     hMember
         As Long
ByVal
     pszAliasTable
      As String
ByVal
     pszAlias
        As String
ByVal
     pszCombination
      As String
ByVal
     usBufSize
        As Integer
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名の組合せを取得するメンバーのハンドル。
pszAliasTable	別名の組合せを取得する先の別名テーブル。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszAlias	次の別名を戻すためのバッファ。次の別名は、このパラメータでの指定によって決定します。ゼロ長の文字列の場合は、最初の別名が戻されます。パラメータが有効な別名の組合せの場合は、次の別名が戻されます。
pszCombination	戻された別名のメンバー組合せ。このバッファは呼出し元によって割り当てられます。
usBufSize	pszCombination バッファのサイズ。

備考

- 呼び出す前に、pszAlias 用にサイズ ESB_MBRNAMELINE のスペースを割り当てる必要があります。
- pszCombination にスペースを割り当てる必要があります。呼出し元は usBufSize パラメータでこのバッファの長さを設定しておく必要があります。
- pszAlias パラメータを使用して、次の組合せを検索します。次の組合せの取得方法の詳細は、このパラメータの説明を参照してください。
- 別名の組合せが(それ以上)ない場合、pszCombination は ESB_NULL に設定され、呼出しは 0 を戻します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_ALIASTABLE

例

```
    Declare Function EsbOtlGetNextAliasCombination Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszAliasTable As String, ByVal pszAlias As String,
ByVal pszCombination As String, ByVal usBufSize As Integer) As Long

Sub ESB_OtlGetNextAliasCombination()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberJan As Long
Dim szAlias As String * ESB_MBRNAMELEN
Dim szCombination As String * 100
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Jan", hMemberJan)
End If
If sts = 0 And hMemberJan <> 0 Then
szCombination = "xxx"
  Do While sts = 0 And Left$(szCombination, 1)
  <> Chr$(0)
   sts = EsbOtlGetNextAliasCombination
    (hOutline, hMemberJan, "Default", szAlias, szCombination, 100)
  Loop
End If
End Sub
```

関連トピック

- EsbOtlAddAliasCombination
- EsbOtlDeleteAliasCombination

EsbOtlGetNextSharedMember

指定されたメンバーの次の共有メンバーにメンバー・ハンドルを戻します。

構文

```
      EsbOtlGetNextSharedMember
       (
      hOutline, hMember, phMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
```

```
 As Long

phMember
 As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　次の共有メンバーを検索するメンバー。

phMember　アウトラインの次の共有メンバーの戻り変数。共有メンバーがもうない場合は、このパラメータは ESB_NULL です。

備考

- hMember が実際のメンバーの場合、最初の共有メンバーが phMember パラメータに戻されます。hMember が共有メンバーの場合、次の共有メンバーが phMember パラメータに戻されます。
- 共有メンバーが(それ以上)ない場合、phMember は ESB_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetNextSharedMember Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
phMember As Long) As Long

Sub ESB_OtlGetNextSharedMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMargin As Long
Dim hShared As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Margin", hMargin)
End If
If sts = 0 Then
 Do While sts = 0 And hMargin <> 0
  sts = EsbOtlGetNextSharedMember(hOutline, hMargin, hShared)
   hMargin = hShared
   hShared = ESB_NULL
 Loop
End If
```

```
End Sub
```

関連トピック

- EsbOtlFindMember

EsbOtlGetNextSibling

メンバーの次の兄弟を戻します。

構文

```
      EsbOtlGetNextSibling
       (
      hOutline, hMember, phMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long

      phMember
       As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　兄弟を取得するメンバーのハンドル。

phMember　hMember パラメータの兄弟のハンドルの戻り変数。

備考

次の兄弟がない場合、phMember は ESB_NULL に設定され、呼び出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetNextSibling Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
phMember As Long) As Long

Sub ESB_OtlGetNextSibling()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hChild As Long
Dim hNextSibling As Long
```

```
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit", hChild)
End If
If sts = 0 And hChild <> 0 Then
 sts = EsbOtlGetNextSibling(hOutline, hChild, hNextSibling)
End If
End Sub
```

関連トピック

- EsbOtlGetPrevSibling
- EsbOtlGetParent
- EsbOtlGetChild
- EsbOtlGetFirstMember

EsbOtlGetOutlineInfo

アウトライン・ファイルに関する情報を戻します。

構文

```
      EsbOtlGetOutlineInfo
       (
      hOutline, pInfo, pusCount
      )
ByVal
      hOutline
       As Long

      pInfo
        As ESB_OUTLINEINFO_T

      pusCount
       As Integer
```

パラメータ 説明

hOutline	アウトラインのコンテキスト・ハンドル。
pInfo	情報構造体の戻り変数。ESB_OUTLINEINFO_T 構造体は呼出し元が割り当てる必要があります。
pusCount	アウトラインの別名テーブルの数に対する戻り変数。

備考

- 呼出し元では、(pusCount 変数で戻された)各別名テーブルごとに **EsbGetNextItem**()を 1 回呼び出す必要があります。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetOutlineInfo Lib
"ESBOTLN" (ByVal hOutline As Long, pInfo As ESB_OUTLINEINFO_T,
pusCount As Integer) As Long

Sub ESB_OtlGetOutlineInfo()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Info As ESB_OUTLINEINFO_T
Dim szAliasTable As String * ESB_ALIASNAMELEN
Dim usCount As Integer
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
  sts = EsbOtlGetOutlineInfo(hOutline, Info, usCount)
  Do While sts = 0 And usCount > 0
   sts = EsbGetNextItem(hCtx, ESB_OUTLINEINFO_TYPE,
    ByVal szAliasTable)
   usCount = usCount - 1
  Loop
End If
End Sub
```

関連トピック

- EsbOtlSetOutlineInfo

EsbOtlGetParent

メンバーの親を戻します。

構文

```
      EsbOtlGetParent
       (
      hOutline, hMember, phMember
      )
ByVal
      hOutline
```

```
      As Long
ByVal
      hMember
       As Long

      phMember
       As Long
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　親を取得するメンバーのハンドル。

phMember　hMember パラメータの親のメンバーのハンドルに対する戻り変数。

備考

- 親がない場合、phMember は ESB_NULL に設定され、呼出しは 0 を戻します。(hMember は次元です。)

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetParent Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
phMember As Long) As Long

Sub ESB_OtlGetParent()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberProfit As Long
Dim hParent As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit",
  hMemberProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
 sts = EsbOtlGetParent(hOutline, hMemberProfit,
  hParent)
End If
End Sub
```

関連トピック

- EsbOtlGetChild
- EsbOtlGetNextSibling
- EsbOtlGetPrevSibling
- EsbOtlGetFirstMember

EsbOtlGetPrevSibling

メンバーの前の兄弟を戻します。

構文

```
      EsbOtlGetPrevSibling
       (
      hOutline, hMember, phMember
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long

      phMember
       As Long
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	前の兄弟を取得するメンバーのハンドル。
phMember	hMember パラメータの前の兄弟のハンドルの戻り変数。

備考

- 前の兄弟がない場合、phMember は ESB_NULL に設定され、呼出しは 0 を戻します。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetPrevSibling Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
phMember As Long) As Long

Sub ESB_OtlGetPrevSibling()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hChild As Long
```

```
Dim hPrevSibling As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit", hChild)
End If
If sts = 0 And hChild <> 0 Then
 sts = EsbOtlGetPrevSibling(hOutline, hChild, hPrevSibling)
End If
End Sub
```

関連トピック

- EsbOtlGetNextSibling
- EsbOtlGetParent
- EsbOtlGetChild
- EsbOtlGetFirstMember

EsbOtlGetUpdateTime

指定したアウトラインのタイムスタンプが戻されます。

構文

```
      EsbOtlGetUpdateTime
       (
      hOutline
      ,
      TimeStamp
      )
ByVal
      hOutline
       As Long

      TimeStamp
       As Long
```

パラメータ 説明

hOutline　アウトライン・ハンドル

TimeStamp　アウトラインのタイムスタンプ

備考

- 時刻の値(Long 型)は、00:00:00 1/1/1970 GMT からの秒数で示されます。
- 時刻の値には永続性はありません。したがって、サーバーがデータベースをロードするとリセットされます。

戻り値

指定したアウトラインのタイムスタンプが戻されます。

関連トピック

- EsbOtlGetOutlineInfo
- EsbOtlSetOutlineInfo
- EsbOtlVerifyOutline
- EsbOtlSortChildren
- EsbOtlGenerateCurrencyOutline

EsbOtlGetUserAttributes

メンバーのユーザー定義属性をすべて取得します。

構文

```
      EsbOtlGetUserAttributes
       (
      hOutline, hMember, pusCount
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
       As Long

      pusCount
       As Integer
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　ユーザー定義属性を取得するメンバーのハンドル。

pusCount　戻されるユーザー属性の数。ppAttributeList 配列の要素数を定義します。

備考

- 各ユーザー定義属性(*pusCount 属性)ごとに 1 回 **EsbGetNextItem()**を呼び出します。
- 呼出し元は **EsbOtlSetUserAttribute()**を使用してメンバーの任意の数のユーザー定義属性を設定できます。各属性は、メンバー名と同じ表記規則に従った一意の文字列として定義されます。
- ユーザー属性は、メンバー名、別名、世代名またはレベル名と同じであっても構いません。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlGetUserAttributes Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
pusCount As Integer) As Long

Sub ESB_OtlGetUserAttributes()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMember As Long
Dim AttributeList As String * ESB_MBRNAMELEN
Dim n As Integer
Dim Count As Integer
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Jan",
  hMember)
End If
If sts = 0 And hMember <> 0 Then
  '********************
  ' Get User Attributes
  '********************
 sts = EsbOtlGetUserAttributes(hOutline,
  hMember, Count)
End If
If sts = 0 And Count <> 0 Then
 For n = 1 To Count
  '*******************************
  ' Get next User Attribute String
  ' from the list
  '*******************************
 sts = EsbGetNextItem(hCtx,
  ESB_OTLUSERATTR_TYPE, ByVal AttributeList)
  Next
End If
End Sub
```

関連トピック

- EsbOtlDeleteUserAttribute
- EsbOtlSetUserAttribute

EsbOtlMoveMember

メンバーを移動します。

構文

```
      EsbOtlMoveMember
       (
      hOutline, hMember, hNewParent, hNewPrevSibling
      )
ByVal
      hOutline
          As Long
ByVal
      hMember
           As Long
ByVal
      hNewParent
         As Long
ByVal
      hNewPrevSibling
       As Long
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	移動するメンバーのハンドル。
hNewParent	新しい親のハンドル。このフィールドは、hNewPrevSibling フィールドが ESB_NULL の場合にのみ使用されます。
hNewPrevSibling	新しい以前の兄弟のハンドル。

備考

- 移動したメンバーは、hPrevSibling メンバーの後に挿入されます。このフィールドが ESB_NULL である場合、移動したメンバーは hParent にで指定した親の最初の子になります。
- hParent および hPrevSibling が ESB_NULL である場合、移動したメンバーはアウトラインの最初の次元になります。
- 型が ESB_ATTRMBRDT_STRING でないゼロレベル(リーフ・ノード)の属性メンバーを移動すると、1269 ページの「ESB_ATTRSPECS_T」構造体のアウトラインの定義によって、メンバーのロング名がリセットされます。
- 祖先を移動すると、ゼロレベルの属性メンバーのロング名に影響する場合があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_MOVE

例

```
    Declare Function EsbOtlMoveMember Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal hNewParent As Long, ByVal hNewPrevSibling As Long) As Long
```

```
Sub ESB_OtlMoveMember()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberProfit As Long
Dim hFQ As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "First Q", hFQ)
End If
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit", hMemberProfit)
End If
If sts = 0 And hFQ And hMemberProfit Then
 sts = EsbOtlMoveMember(hOutline, hFQ,
  hMemberProfit, ESB_NULL)
End If
End Sub
```

関連トピック

- EsbOtlFindMember
- EsbOtlRenameMember
- EsbOtlAddMember
- EsbOtlDeleteMember

EsbOtlNewOutline

ファイルを作成せずにアウトラインを作成します。**EsbOtlOpenOutline**()の代替として使用されます。

構文

```
      EsbOtlNewOutline
       (
      hCtx, pNewInfo, phOutline
      )
ByVal
      hCtx
         As Long

      pNewInfo
       As ESB_OUTLINEINFO_T

      phOutline
       As Long
```

パラメータ	説明
hCtx	Essbase コンテキスト・ハンドル。
pNewInfo	新規アウトラインを記述する構造体。
phOutline	ESB_HOUTLINE_T 値の戻り変数。このハンドルは API によって設定され、後続のアウトライン API 関数に渡される必要があります。

備考

- この関数では、メモリーに空のアウトラインが作成されます。
- この呼出しが使用された場合はトランザクションは維持されません。トランザクションの維持の詳細は、**EsbOtlOpenOutline()**を参照してください。

戻り値

成功の場合、0 が戻されます。

例

```
    Declare Function EsbOtlNewOutline Lib
"ESBOTLN.DLL" (ByVal hCtx As Long, pNewInfo As ESB_OUTLINEINFO_T,
phOutline As Long) As Long

Sub ESB_OtlNewOutline()
Dim sts As Long
Dim NewInfo As ESB_OUTLINEINFO_T
Dim hOutline As Long
NewInfo.usOutlineType = ESB_DBTYPE_NORMAL
NewInfo.fCaseSensitive = ESB_FALSE
NewInfo.fAutoConfigure = ESB_TRUE
sts = EsbOtlNewOutline(hCtx, NewInfo, hOutline)
End Sub
```

関連トピック

- EsbOtlOpenOutline
- EsbOtlWriteOutline
- EsbOtlRestructure
- EsbOtlCloseOutline
- EsbOtlVerifyOutline

EsbOtlOpenOutline

既存のアウトラインを開いて読み取ります。アウトラインに対する操作を実行する前に、この関数(または **EsbOtlNewOutline()**)を呼び出す必要があります。

構文

```
EsbOtlOpenOutline
 (
hCtx, pObject, fLock, fKeepTrans, phOutline
)
```

```
ByVal
      hCtx
         As Long

      pObject
        As ESB_OBJDEF_T
ByVal
      fLock
         As Integer
ByVal
      fKeepTrans
       As Integer

      phOutline
       As Long
```

パラメータ 説明

hCtx　Essbase コンテキスト・ハンドル。

pObject　開く対象のアウトライン・オブジェクト。

fLock　開いたときにアウトラインをロックするかどうかを特定するフラグ。サーバー・アウトラインの場合にのみ有効です。

fKeepTrans　トランザクションを保持するかどうかを特定するフラグ。

既存のアウトラインを開いて変更する場合、データベースの再構築を行ってデータを保持する場合には、フラグを ESB_YES に設定してください。ESB_YES では、アウトラインに対する操作のログが残されます。

空のアウトラインから始める場合や、再構築の際にデータを保存しない場合には、このフィールドを ESB_NO に設定してください。ESB_NO に設定すると、ログがとられないので、時間やメモリーを節約できます。

phOutline　ESB_HOUTLINE_T の戻り変数。このハンドルは API によって設定され、後続のアウトライン API 関数に渡される必要があります。

備考

- アウトライン・ファイルがサーバー上に存在する場合、この呼出しは、クライアントのアクセスのためにそのファイルをローカルにコピーします。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_FILEOPEN
- OTLAPI_ERR_FILEIO

アクセス

この関数を使用するには、呼出し元は指定したアプリケーション、アウトライン・オブジェクトが含まれているデータベースのいずれか、またはその両方に対して、適切なレベルのアクセス権を持っている必要があります。アウトライン・オブジェクトをロックするには(lock フラグを ESB_YES に設定)、指定したアプリケーショ

ンまたはアウトラインが含まれているデータベースに対して、アプリケーション・デザイナまたはデータベース・デザイナ権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)を持っている必要があります。

例

```
    Declare Function EsbOtlOpenOutline Lib
"ESBOTLN.DLL" (ByVal hCtx As Long, pObject As ESB_OBJDEF_T,
ByVal fLock As Integer, ByVal fKeepTrans As Integer,
phOutline As Long) As Long

Sub ESB_OtlOpenOutline()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
End Sub
```

関連トピック

- EsbOtlNewOutline
- EsbOtlWriteOutline
- EsbOtlRestructure
- EsbOtlCloseOutline
- EsbOtlVerifyOutline

EsbOtlOpenOutlineQuery

既存のアウトラインを開きます。

構文

```
     EsbOtlOpenOutlineQuery
      (
     hCtx, pObject, phOutline
     )
ByVal
     hCtx
        As Long

     pObject
       As ESB_OBJDEF_T

     phOutline
      As Long
```

パラメータ 説明

hCtx　アウトラインのコンテキスト・ハンドル。有効なサーバー・ログイン・コンテキストである必要があります。

pObject　開く対象のアウトライン・オブジェクトを定義しているオブジェクト構造体を指すポインタ。現在これは無視されています。アクセスするデータベースに対して **EsbSetActive()** を呼び出す必要があります。

phOutline　ESB_HOUTLINE_T 変数を指すポインタ。API によって設定され、以降の API 関数に渡されます。

備考

- **EsbOtlQueryMembers()**を使用してアウトラインにアクセスするには、この関数を使用します。
- この関数を呼び出してもアウトラインはダウンロードされず、ファイル全体がメモリーにロードされます。
- したがって、多くのアウトライン API 関数は、この関数呼出しから戻される hOutline を処理できません。
- この呼出しの後で次の呼出しにアクセスできます。他のすべてのアウトライン API 呼出しはエラーを戻します。
  - EsbOtlCloseOutline
  - EsbOtlGetMemberAlias
  - EsbOtlGetMemberFormula
  - EsbOtlGetMemberInfo
  - EsbOtlGetNextAliasCombination
  - EsbOtlGetOutlineInfo
  - EsbOtlGetUserAttributes
  - EsbOtlGetGenName
  - EsbOtlGetGenNames
  - EsbOtlGetLevelName
  - EsbOtlGetLevelNames

戻り値

関数が正常終了した場合、戻り値は 0 になります。

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_FILEOPEN
- OTLAPI_ERR_FILEIO

例

```
    Declare Function EsbOtlOpenOutlineQuery Lib "ESBOTLN.DLL"
(ByVal hCtx As Long, pObject As ESB_OBJDEF_T, phOutline As Long) As Long

Sub ESB_OtlOpenOutlineQuery()
```

```
 Dim sts As Long
 Dim hOutline As Long
 Dim Object As ESB_OBJDEF_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String
 AppName = "Sample"
 DbName = "Basic"
 sts = EsbSetActive(hCtx, AppName, DbName, Access)
 If sts = 0 Then
  sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 End If
End Sub
```

関連トピック

- EsbOtlCloseOutline
- EsbOtlOpenOutline
- EsbOtlQueryMembers
- EsbOtlQueryMembersByName
- EsbSetActive

EsbOtlQueryAttributes

指定した属性メンバーまたは属性次元についてのメンバー情報にクエリーを行います。

構文

```
      EsbOtlQueryAttributes
       (
      hOutline
      ,
      AttrQuery
      ,
      Count
      ,
      MemberArray
      )
ByVal
      hOutline
        As Long

      AttrQuery
       As ESB_ATTRIBUTEQUERY_T

      Count
         As Long

      MemberArray
       As Variant
```

パラメータ　説明

hOutline　アウトラインのハンドル

AttrQuery　クエリーを定義する構造体

Count　戻されたメンバーのハンドルの数

MemberArray 戻されたメンバーのハンドルの配列

備考

この関数を呼び出す前に、**EsbOpenOutlineQuery()**を呼び出してクエリー・モードでアウトラインを開いてください。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
        Sub ESB_OtlQueryAttributes()
    Dim OutAttrInfo As ESB_ATTRIBUTEINFO_T
    Dim InAttrQuery As ESB_ATTRIBUTEQUERY_T
    Dim MbrInfo As ESB_MBRINFO_T
    Dim index As Integer
    Dim test As Integer
    Dim Count As Long
    Dim sts As Long
    Dim Dummy As String
    Dim MbrName As String
    Dim attribdtvar As Variant
    Dim OutMemberArray As Variant
    InAttrQuery.InputMember = "Product"
    InAttrQuery.InputMemberType = ESB_STANDARD_DIMENSION
    InAttrQuery.OutputMemberType = ESB_ATTRIBUTE_DIMENSION
    InAttrQuery.Operation = ESB_ALL
    InAttrQuery.Attribute = ""
    sts = EsbOtlQueryAttributes(ghOutline, InAttrQuery, Count, OutMemberArray)
    If sts = 0 Then
        Out "attribute query Count is : " & Count
        Out "EsbOtlGetMemberInfo passed"
        For index = 0 To Count - 1
            sts = EsbOtlGetMemberInfo(ghOutline, OutMemberArray(index), MbrInfo)
            If sts = 0 Then
                Out "MbrName  : " & MbrInfo.szMember
            Else
            Out "EsbOtlGetMemberInfo Failed: " & sts
            End If
        Next index
    Else
        Out "EsbOtlQueryAttributes failed: " & sts
    End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlSetAttributeSpecifications

EsbOtlQueryMembers

アウトラインをクエリーします。

構文

```
      EsbOtlQueryMembers (
      hOutline, hMember, pPredicate, pMbrCounts, pulCount
      )
ByVal
      hOutline
        As Long
ByVal
      hMember
        As Long

      pPredicate
       As ESB_PREDICATE_T

      pMbrCounts
       As ESB_MBRCOUNTS_T
```

パラメータ	説明
hOutline	Essbase アウトライン・ハンドル。**EsbOtlOpenOutlineQuery()**から戻されている必要があります。
hMember	操作を行うメンバーのメンバー・ハンドル。この値が NULL である場合、その次元の論理親を表し、アウトラインの最上部であるとみなされます。次のオプションでは、この値は無視されます: ● ESB_NAMEDGENERATION ● ESB_NAMEDLEVEL ● ESB_USERATTRIBUTE ● ESB_SEARCH ● ESB_WILDSEARCH

パラメータ	説明
pPredicate	クエリーを定義している構造体。この構造体のフィールドは、「注意」を参照してください。
pMbrCounts	カウントに関する情報を定義している構造体。次のフィールドが含まれます: ● ulStart- 戻される開始番号。 ● ulMaxCount- 戻されるメンバーのハンドルの最大数。 ● ulTotalCount- クエリーの実行結果において定義されるメンバーの合計数。 ● pulReturnCount- このクエリーにおいて戻されるメンバーのハンドルの数。
phMemberArray	クエリーから戻されたメンバーのハンドルの配列。

備考

- この呼出しは、操作対象のメンバーのハンドルを使用して、オプション値で指定された基準に適合するメンバーのハンドル配列を戻します。
- 戻された phMembers のメンバー配列が不要になった場合、呼出し元は **EsbOtlFreeMember()**を呼び出す必要があります。
- 配列の各 hMember 要素は、**EsbOtlOpenOutlineQuery()**にリストされた呼出しのみで使用できます。たとえば、戻されたメンバー・ハンドルは、**EsbOtlGetSibling()**を呼び出すために使用できません。
- プログラマは、各メンバーのハンドルが戻されるたびに、**EsbGetNextItem()**を 1 度呼び出す必要があります。
- pPredicate 構造体のフィールドは、次のように使用されます:
  - **ulQuery** - 実行する操作を定義する値。次のいずれかになります:
    - ESB_CHILDREN
    - ESB_DESCENDANTS
    - ESB_BOTTOMLEVEL
    - ESB_SIBLINGS
    - ESB_SAMELEVEL
    - ESB_SAMEGENERATION
    - ESB_PARENT
    - ESB_DIMENSION
    - ESB_NAMEDGENERATION
    - ESB_NAMEDLEVEL
    - ESB_SEARCH
    - ESB_WILDSEARCH
    - ESB_USERATTRIBUTE
    - ESB_ANCESTORS
    - ESB_DTSMEMBERS
    - ESB_DIMUSERATTRIBUTES

- **ulOptions** - 任意のオプションを定義する値。次のクエリー・オプションとともに使用されます:
  - ESB_SEARCH、ESB_WILDSEARCH - 次のいずれかの値になります:
    - ESB_MEMBERSONLY
    - ESB_ALIASESONLY
    - ESB_MEMBERSANDALIASES
  - すべてのオプション - ESB_COUNTONLY: メンバーのハンドルは戻さずに、pCounts 構造体の pTotalCount フィールドにのみ値を入れます。
- **szDimension** - クエリーの範囲を制限する次元。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESB_NAMEDGENERATION
  - ESB_NAMEDLEVEL
  - ESB_USERATTRIBUTE
  - ESB_SEARCH - すべての次元を検索するには、NULL に設定します
  - ESB_WILDSEARCH - すべての次元を検索するには、NULL に設定します
- **pszString1**- オプションによって特定される入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESB_NAMEDGENERATION - 世代の名前。
  - ESB_NAMEDLEVEL - レベルの名前。
  - ESB_SEARCH - 検索する文字列。この文字列は完全一致として定義されています。
  - ESB_WILDSEARCH - 検索する文字列。この文字列は、末尾にオプションの'*'が付いた完全一致検索文字列として指定され、'*'部分は任意の文字の組合せが可能です。
  - ESB_USERATTRIBUTE - ユーザー定義属性。
- **pszString2**- オプションによって特定される入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESB_USERATTRIBUTE - ユーザー定義属性。
  - ESB_SEARCH、ESB_WILDSEARCH - オプションで別名テーブルを検索するよう設定されている場合、この文字列は検索対象の別名テーブルを指定します。このフィールドを NULL にした場合は、すべての別名テーブルがサーチされます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlQueryMembers Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long,
```

```
pPredicate As ESB_PREDICATE_T, pCounts As ESB_MBRCOUNTS_T) As Long
Declare Function EsbOtlFreeMember Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long) As Long

Sub ESB_OtlQueryMembers()
 Dim sts As Long
 Dim hOutline As Long
 Dim hMember As Long
 Dim ihMember As Long
 Dim Object As ESB_OBJDEF_T
 Dim MbrInfo As ESB_MBRINFO_T
 Dim Predicate As ESB_PREDICATE_T
 Dim Counts As ESB_MBRCOUNTS_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 AppName = "Sample"
 DbName = "Basic"
 sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 If sts = 0 Then
  sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)
 Predicate.ulQuery = ESB_CHILDREN
 Predicate.pszDimension = "Year"
 Counts.ulStart = 0
 Counts.ulMaxCount = 10
 If sts = 0 Then
   sts = EsbOtlQueryMembers(hOutline, hMember, Predicate, Counts)
  If sts = 0 And Counts.ulReturnCount <> 0 Then
   For n% = 1 To Counts.ulReturnCount
     sts = EsbGetNextItem(hCtx, ESB_HMEMBER_TYPE, ihMember)
   If sts = 0 And ihMember <> 0 Then
       sts = EsbOtlFreeMember(hOutline, ihMember)
   End If
  Next
  End If
 End If
End If
End Sub
```

関連トピック

- EsbGetNextItem
- EsbOtlFreeMember
- EsbOtlGetDimensionUserAttributes
- EsbOtlOpenOutlineQuery
- EsbOtlQueryMembersByName

EsbOtlQueryMembersByName

アウトラインをクエリーします。

構文

```
      EsbOtlQueryMembersByName
       (
      hOutline, pszMember, pPredicate, pCounts
      )
ByVal
      hOutline
        As Long
ByVal
      pszMember
       As String

      pPredicate
       As ESB_PREDICATE_T

      pCounts
        As ESB_MBRCOUNTS_T
```

パラメータ 説明

hOutline　Essbase アウトライン・ハンドル。EsbOtlOpenOutlineQuery()から戻されている必要があります。

pszMember　操作を行うメンバーのメンバー名文字列。この値が NULL である場合、その次元の論理親を表し、アウトラインの最上部であるとみなされます。次のオプションでは、この値は無視されます:

- ESB_NAMEDGENERATION
- ESB_NAMEDLEVEL
- ESB_USERATTRIBUTE
- ESB_SEARCH
- ESB_WILDSEARCH

pPredicate　クエリーを定義している構造体。この構造体のフィールドは、「注意」を参照してください。

pCounts　カウントに関する情報を定義している構造体。次のフィールドが含まれます:

- ulStart- 戻される開始番号。
- ulMaxCount- 戻されるメンバーのハンドルの最大数。
- ulTotalCount- クエリーの実行結果において定義されるメンバーの合計数。
- pulReturnCount- このクエリーにおいて戻されるメンバーのハンドルの数。

備考

- この呼出しは、操作対象のメンバー名文字列を使用して、オプション値で指定された基準に適合するメンバーのハンドル配列を戻します。
- 戻された phMembers のメンバー配列が不要になった場合、呼出し元は **EsbOtlFreeMember()**を呼び出す必要があります。
- 配列の各 hMember 要素は、**EsbOtlOpenOutlineQuery()**にリストされた呼出しのみで使用できます。たとえば、戻されたメンバー・ハンドルは、**EsbOtlGetSibling()**を呼び出すために使用できません。

- プログラマは、各メンバーのハンドルが戻されるたびに、**EsbGetNextItem()**を1 度呼び出す必要があります。
- pPredicate 構造体のフィールドは、次のように使用されます:
  - **ulQuery** - 実行する操作を定義する値。次のいずれかになります:
    - ESB_CHILDREN
    - ESB_DESCENDANTS
    - ESB_BOTTOMLEVEL
    - ESB_SIBLINGS
    - ESB_SAMELEVEL
    - ESB_SAMEGENERATION
    - ESB_PARENT
    - ESB_DIMENSION
    - ESB_NAMEDGENERATION
    - ESB_NAMEDLEVEL
    - ESB_SEARCH
    - ESB_WILDSEARCH
    - ESB_USERATTRIBUTE
    - ESB_ANCESTORS
    - ESB_DTSMEMBERS
    - ESB_DIMUSERATTRIBUTES
  - **ulOptions** - 任意のオプションを定義する値。次のクエリー・オプションとともに使用されます:
    - ESB_SEARCH、ESB_WILDSEARCH - 次のいずれかの値になります:
      - ESB_MEMBERSONLY
      - ESB_ALIASESONLY
      - ESB_MEMBERSANDALIASES
    - すべてのオプション - ESB_COUNTONLY: メンバーのハンドルは戻さずに、pCounts 構造体の pTotalCount フィールドにのみ値を入れます
  - **szDimension** - クエリーの範囲を制限する次元。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
    - SB_NAMEDGENERATION
    - ESB_NAMEDLEVEL
    - ESB_USERATTRIBUTE
    - ESB_SEARCH - すべての次元を検索するには、NULL に設定します
    - ESB_WILDSEARCH - すべての次元を検索するには、NULL に設定します

- **pszString1**- オプションによって決まる入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESB_NAMEDGENERATION - 世代の名前
  - ESB_NAMEDLEVEL - レベルの名前
  - ESB_SEARCH - 検索する文字列。この文字列は完全一致として定義されています
  - ESB_WILDSEARCH - 検索する文字列。この文字列は、末尾にオプションの'*'が付いた完全一致検索文字列として指定され、'*'部分は任意の文字の組合せが可能です。
  - ESB_USERATTRIBUTE - ユーザー定義属性
- **pszString2**- オプションによって特定される入力文字列。このフィールドは次のクエリー・オプションで使用され、それ以外では無視されます:
  - ESB_USERATTRIBUTE - ユーザー定義属性。
  - ESB_SEARCH、ESB_WILDSEARCH - オプションで別名テーブルを検索するよう設定されている場合、この文字列は検索対象の別名テーブルを指定します。NULL の場合、すべての別名テーブルが検索されます。

戻り値

関数が正常終了した場合、戻り値は 0 になります。

例

```
    Declare Function EsbOtlQueryMembersByName Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal pszMember As String,
pPredicate As ESB_PREDICATE_T, pCounts As ESB_MBRCOUNTS_T) As Long
Declare Function EsbOtlFreeMember Lib "ESBOTLN"
(ByVal hOutline As Long, ByVal hMember As Long) As Long

Sub ESB_OtlQueryMembersByName()
 Dim sts As Long
 Dim hOutline As Long
 Dim pszMember As String
 Dim ihMember As Long
 Dim Object As ESB_OBJDEF_T
 Dim MbrInfo As ESB_MBRINFO_T
 Dim Predicate As ESB_PREDICATE_T
 Dim Counts As ESB_MBRCOUNTS_T
 Dim Access As Integer
 Dim AppName As String
 Dim DbName As String

 pszMember =  "Qtr1"
 AppName = "Sample"
 DbName = "Basic"
 sts = EsbOtlOpenOutlineQuery(hCtx, Object, hOutline)  'open outline
 If sts = 0 Then                   'proceed if open successful
                         'else message with error
  Predicate.ulQuery = ESB_CHILDREN
  Predicate.pszDimension = "Year"
  Counts.ulStart = 0
```

```
   Counts.ulMaxCount = 10
   If sts = 0 Then
    sts = EsbOtlQueryMembersByName(hOutline, pszMember, Predicate, Counts)
    If sts = 0 And Counts.ulReturnCount <> 0 Then
     For n% = 1 To Counts.ulReturnCount
      sts = EsbGetNextItem(hCtx, ESB_HMEMBER_TYPE, ihMember)
     If sts = 0 And ihMember <> 0 Then
         sts = EsbOtlFreeMember(hOutline, ihMember)
     End If
     Next
    End If
   End If
  Else
   msgbox "Outline open failed with error: " & sts
  Endif
End Sub
```

関連トピック

- EsbGetNextItem
- EsbOtlFreeMember
- EsbOtlGetDimensionUserAttributes
- EsbOtlOpenOutlineQuery
- EsbOtlQueryMembers

EsbOtlRenameAliasTable

既存の別名テーブル名を変更します。

構文

```
      EsbOtlRenameAliasTable
       (
      hOutline, pszAliasTable, pszNewAliasTable
      )
ByVal
      hOutline
          As Long
ByVal
      pszAliasTable
        As String
ByVal
      pszNewAliasTable
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAliasTable	変更する別名テーブル名。
pszNewAliasTable	新しい別名テーブル名。

備考

- デフォルトの別名テーブルの名前を、"Default"から変更しないでください。
- 別名テーブルの名前を変更した場合、別名テーブルに関連付けられた言語コードは名前が変更された別名テーブルに保持されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_RENAMEDEFALIAS
- OTLAPI_ERR_ALIASTABLENAME
- OTLAPI_ERR_ALIASTABLEEXISTS

例

```
    Declare Function EsbOtlRenameAliasTable Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszAliasTable As String,
ByVal pszNewAliasTable As String) As Long

Sub ESB_OtlRenameAliasTable()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 EsbOtlRenameAliasTable(hOutline, "Alias Table 1",
  "1st Alias Table")
End If
End Sub
```

関連トピック

- EsbOtlCreateAliasTable
- EsbOtlCopyAliasTable
- EsbOtlClearAliasTable
- EsbOtlDeleteAliasTable
- EsbOtlSetAliasTableLanguage

EsbOtlRenameMember

メンバー名を変更します。

構文

```
      EsbOtlRenameMember
       (
      hOutline, hMember, pszNewMember
      )
ByVal
      hOutline
        As Long
ByVal
      hMember
         As Long
ByVal
      pszNewMember
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	名前を変更するメンバーのハンドル。
pszNewMember	新しいメンバー名。

備考

- すべての共有メンバー名も変更されます。
- hMember が共有メンバーを指している場合、この呼出しは失敗します。
- ESB_ATTRMBRDT_STRING 型に属さないゼロ レベル(リーフ・ノード)の属性メンバー名を変更すると、次がリセットされます:
  - 属性値
  - メンバーのロング名。1269 ページの「ESB_ATTRSPECS_T」構造体のアウトラインに対する定義を使用
- 祖先の名前を変更すると、ゼロ レベルの属性メンバーのロング名に影響する場合があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_MBRNAME
- TLAPI_BAD_RENAMESHARE
- OTLAPI_ERR_RENAMENAMEUSED

例

```
    Declare Function EsbOtlRenameMember Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszNewMember As String) As Long

Sub ESB_OtlRenameMember()
```

```
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemProfit As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit",
  hMemProfit)
End If
If sts = 0 And hMemberProfit <> 0 Then
 sts = EsbOtlRenameMember(hOutline, hMemProfit,
  "Prelim Profit")
End If
End Sub
```

関連トピック

- EsbOtlFindMember
- EsbOtlMoveMember
- EsbOtlAddMember
- EsbOtlDeleteMember

EsbOtlRestructure

サーバー上のアウトラインを再構築します。これは非同期の呼出しです。

構文

```
      EsbOtlRestructure
       (
      hCtx, usRestructType
      )
ByVal
      hCtx
           As Long
ByVal
      usRestructType
       As Integer
```

パラメータ	説明
hCtx	サーバー・ログイン・コンテキスト・ハンドル。ESBOTLNriteOutline()を使用してアウトラインが保存されたサーバーである必要があります。

パラメータ	説明
usRestructType	実行する再構築のタイプ。これは、次のいずれかの値にできます: ● ESB_DOR_ALLDATA ● ESB_DOR_INDATA ● ESB_DOR_LOWDATA ● ESB_DOR_NODATA

備考

- この関数を呼び出す前に、呼出し元は **ESBOTLNriteOutline()**を使用してアウトラインを保存しておく必要があります。
- この関数呼出しは、サーバーに保存されたアウトラインに対してのみ有効です。
- これは非同期の呼出しです。この呼出しを行った後、**EsbGetProcessState()**で再構築操作の完了を示すステータスが戻されるまで、**EsbGetProcessState()**を呼び出す必要があります。
- (データが保存され)データが正しく再構築されるには、fKeepTrans フラグを ESB_YES に設定して **EsbOtlOpenOutline()**を使用して、アウトラインを開いておく必要があります。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次の値が戻されます:

OTLAPI_BAD_RESTRUCTTYPE

アクセス

この関数を使用するには、呼出し元は指定したアプリケーション、アウトライン・オブジェクトが含まれているデータベースのいずれか、またはその両方に対して、適切なレベルのアクセス権を持っている必要があります。アウトライン・オブジェクトを再構築するには、指定したアプリケーション、またはアウトラインが含まれるデータベースに対して、アプリケーション・デザイナまたはデータベース・デザイナの権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)が必要です。

例

```
    Declare Function EsbOtlRestructure Lib
"ESBOTLN.DLL" (ByVal hCtx As Long, ByVal
usRestructType As Integer) As Long

Sub ESB_OtlRestructure()
Dim hCtx As Long
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
```

```
ESB_YES, ESB_YES, hOutline)
'***
'body of code
'write outline to server using
'ESBOTLNriteOutline()
'***
If sts = 0 Then
 sts = EsbOtlRestructure(hCtx, ESB_DOR_ALLDATA)
End If
'***
'need to call EsbGetProcessState()
'to check for completion before proceeding
'***
End Sub
```

関連トピック

- EsbOtlOpenOutline
- EsbOtlNewOutline
- EsbOtlWriteOutline
- EsbOtlVerifyOutline
- EsbOtlCloseOutline

EsbOtlSetAliasTableLanguage

指定した別名テーブルの言語コードを設定します。

別名テーブルの言語コードを設定すると、ApplCore セッションで実行されているアプリケーションが Essbase データベースにアクセスしたときに、アプリケーション選択で正しい別名テーブルが自動的に選択されます。

構文

```
      ESB_FUNC_M
      EsbOtlSetAliasTableLanguage
       (
      hOutline
      ,
      pszAliasTable
      ,
      pszLanguageCode
      )
ByVal
      hOutline
          As Long
ByVal
      pszAliasTable
        As String
ByVal
      pszLanguageCode
       As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszAliasTable	言語コードを設定する別名テーブル名。
pszLanguageCode	pszAliasTable で指定された別名テーブルに割り当てる言語コード。 言語コードは、ApplCore セッションからの中間層言語タグである必要があります。言語コードの大文字と小文字は区別されません。

備考

- デフォルトの別名テーブルで言語コードを設定することはできません。
- 別名テーブルにはいくつでも言語コードを割り当てることができます。複数の言語コードを設定するには、言語コードごとにこの関数を呼び出します。
- 新しい言語コードを設定しても、別名テーブルに現在割り当てられている言語コードは上書きされません。
- 同じ言語コードを同じデータベース内の別の別名テーブルに割り当てないでください。

戻り値

- 成功の場合、0 が戻されます。
- 失敗した場合は、次のいずれかのエラーが戻されます:
  - OTLAPI_BAD_ALIASTABLE (無効な別名テーブル)
  - OTLAPI_ERR_DUP_LANGCODE (言語コードが同じデータベース内の別の別名テーブルに割り当てられている)

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Declare Function EsbOtlGetAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String, pulCount As Long) As Long
Declare Function EsbOtlSetAliasTableLanguage Lib "esbotln" (ByVal hOutline As Long,
ByVal pszAliasTable As String, ByVal pszLanguageCode As String) As Long
Declare Function EsbOtlClearAliasTableLanguages Lib "esbotln" (ByVal hOutline As
Long, ByVal pszAliasTable As String) As Long

Sub ESB_Sub ()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Items As Long
Dim AliasLang As String * ESB_ALIASNAMELEN

Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
```

```
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlCreateAliasTable(hOutline,
  "French Alias Table")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr")
End If
If sts = 0 Then
 sts = EsbOtlSetAliasTableLanguage(hOutline,
  "French Alias Table", "fr-CA")
End If

If sts = 0 Then
 sts = EsbOtlGetAliasTableLanguages(hOutline,
  "French Alias Table", Items)
 If sts = 0 Then
    For N = 1 To Items
          sts = EsbGetNextItem(hCtx, ESB_ALIASLANG_TYPE, ByVal AliasLang)
     Next
 End If
End If

If sts = 0 Then
 sts = EsbOtlClearAliasTableLanguages(hOutline,
  "French Alias Table")
End If

End Sub
```

関連トピック

- EsbOtlGetAliasTableLanguages
- EsbOtlClearAliasTableLanguages

EsbOtlSetAttributeSpecifications

アウトラインの属性指定を設定します。

構文

```
     EsbOtlSetAttributeSpecifications
      (
     hOutline
     ,
     AttrSpecs
     )
ByVal
     hOutline
      As Long

     AttrSpecs
```

```
      As ESB_ATTRSPECS_T
```

パラメータ 説明

hOutline　アウトラインのハンドル

AttrSpecs　属性指定

備考

- 属性指定は、次のような場合に使用します:
  - ロング名の生成
  - 日時属性のフォーマットの指定
  - 数値属性のバケットのタイプの指定
  - 属性計算次元名およびそこで使用される値の名前の提供
- 属性指定を設定しない場合、アウトラインではデフォルトの属性指定が使用されます。
- 属性指定を変更すると、再構築されることがあります。

アクセス

この関数を使用するのに、特別な権限は必要ありません。

例

```
    Sub ESB_OtlSetAttributeSpecifications()
  Dim InAttrSpecs As ESB_ATTRSPECS_T
  eraser
  InAttrSpecs.GenNameBy = InputBox("Enter GenNameBy:" & vbCrLf & _
    "0. ESB_GENNAMEBY_PREFIX" & vbCrLf & _
    "1. ESB_GENNAMEBY_SUFFIX")
  InAttrSpecs.UseNameOf = InputBox("Enter UseNameOf:" & vbCrLf & _
    "0. ESB_USENAMEOF_NONE" & vbCrLf & _
    "1. ESB_USENAMEOF_PARENT" & vbCrLf & _
    "2. ESB_USENAMEOF_GRANDPARENTANDPARENT" & vbCrLf & _
    "3. ESB_USENAMEOF_ALLANCESTORS" & vbCrLf & _
    "4. ESB_USENAMEOF_DIMENSION")
  InAttrSpecs.Delimiter = InputBox("Enter Delimiter:" & vbCrLf & _
    "0. ESB_DELIMITER_UNDERSCORE" & vbCrLf & _
    "1. ESB_DELIMITER_PIPE" & vbCrLf & _
    "2. ESB_DELIMITER_CARET")
  InAttrSpecs.DateFormat = InputBox("Enter DateFormat:" & vbCrLf & _
    "0. ESB_DATEFORMAT_MMDDYYYY" & vbCrLf & _
    "1. ESB_DATEFORMAT_DDMMYYYY")
  InAttrSpecs.BucketingType = InputBox("Enter BucketingType:" & vbCrLf & _
    "0. ESB_UPPERBOUNDINCLUSIVE" & vbCrLf & _
    "1. ESB_ESB_LOWERBOUNDINCLUSIVE" & vbCrLf & _
    "2. ESB_UPPERBOUNDNONINCLUSIVE" & vbCrLf & _
    "3. ESB_ESB_LOWERBOUNDNONINCLUSIVE")
  InAttrSpecs.DefaultTrueString = InputBox("Enter DefaultTrueString: ", ,
"ESB_DEFAULT_TRUESTRING")
  InAttrSpecs.DefaultFalseString = InputBox("Enter DefaultFalseString: ", ,
"ESB_DEFAULT_FALSESTRING")
```

```
  InAttrSpecs.DefaultAttrCalcDimName = InputBox("Enter DefaultAttrCalcDimName: ", ,
"ESB_DEFAULT_ATTRIBUTECALCULATIONS")
  InAttrSpecs.DefaultSumMbrName = InputBox("Enter DefaultSumMbrName: ", ,
"ESB_DEFAULT_SUM")
  InAttrSpecs.DefaultCountMbrName = InputBox("Enter DefaultCountMbrName: ", ,
"ESB_DEFAULT_COUNT")
  InAttrSpecs.DefaultAverageMbrName = InputBox("Enter DefaultAverageMbrName: ", ,
"ESB_DEFAULT_AVERAGE")
  InAttrSpecs.DefaultMinMbrName = InputBox("Enter DefaultMinMbrName: ", ,
"ESB_DEFAULT_MIN")
  InAttrSpecs.DefaultMaxMbrName = InputBox("Enter DefaultMaxMbrName: ", ,
"ESB_DEFAULT_MAX")
  sts = EsbOtlSetAttributeSpecifications(ghOutline, InAttrSpecs)
  If sts = 0 Then
    Out "ESB_OtlSetAttributeSpecifications passed: " & sts
    Out "GenNameBy : " & InAttrSpecs.GenNameBy
    Out "UseNameOf : " & InAttrSpecs.UseNameOf
    Out "Delimiter : " & InAttrSpecs.Delimiter
    Out "DateFormat : " & InAttrSpecs.DateFormat
    Out "BucketingType : " & InAttrSpecs.BucketingType
    Out "DefaultTrueString : " & InAttrSpecs.DefaultTrueString
    Out "DefaultFalseString : " & InAttrSpecs.DefaultFalseString
    Out "DefaultAttrCalcDimName : " & InAttrSpecs.DefaultAttrCalcDimName
    Out "DefaultSumMbrName : " & InAttrSpecs.DefaultSumMbrName
    Out "DefaultCountMbrName : " & InAttrSpecs.DefaultCountMbrName
    Out "DefaultAverageMbrName : " & InAttrSpecs.DefaultAverageMbrName
    Out "DefaultMinMbrName : " & InAttrSpecs.DefaultMinMbrName
    Out "DefaultMaxMbrName : " & InAttrSpecs.DefaultMaxMbrName
  Else
    Out "ESB_OtlSetAttributeSpecifications failed" & sts
    Exit Sub
  End If
End Sub
```

関連トピック

- EsbCheckAttributes
- EsbGetAssociatedAttributesInfo
- EsbGetAttributeInfo
- EsbGetAttributeSpecifications
- EsbOtlAssociateAttributeDimension
- EsbOtlAssociateAttributeMember
- EsbOtlDisassociateAttributeDimension
- EsbOtlDisassociateAttributeMember
- EsbOtlFindAttributeMembers
- EsbOtlGetAssociatedAttributes
- EsbOtlGetAttributeInfo
- EsbOtlGetAttributeSpecifications
- EsbOtlQueryAttributes

EsbOtlSetDTSMemberAlias

DTS メンバーの別名を設定します。

構文

```
      EsbOtlSetDTSMemberAlias
       (
      hOutline, pszDTSMember, pszAlias, pszAliasTable
      )
ByVal
      hOutline
         As Long
ByVal
      pszDTSMember
       As String
ByVal
      pszAlias
         As String
ByVal
      pszAliasTable
       As String
```

パラメータ	説明
hOutline	EsbOtlOpenOutlineQuery 呼出しから戻される Essbase アウトライン・ハンドル。
pszDTSMember	別名を提供する DTS メンバー名。
pszAlias	DTS メンバーの別名を含む C 文字列を指すポインタ。
pzsAliasTable	別名を提供する別名テーブルの名前。NULL の場合は、デフォルトの別名テーブルが使用されます。

戻り値

成功の場合、戻り値はゼロです。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_DTSMBRNOTDEFINED
- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_ILLEGALALIASSTRING
- OTLAPI_ERR_DUPLICATEALIAS

例

```
   Public Sub ESB_OtlSetDTSMemberAlias()
 Dim DTSMember As String * ESB_MBRNAMELEN
 Dim Alias As String * ESB_ALIASNAMELEN
 Dim AliasTable As String * ESB_ALIASNAMELEN

 DTSMember = "Y-T-D"
 Alias = "Year_To_Date"
 AliasTable = "default"
```

```
  sts = EsbOtlSetDTSMemberAlias(hOutline, DTSMember, _
                Alias, AliasTable)
End Sub
```

関連トピック

- EsbOtlDeleteDTSMemberAlias
- EsbOtlEnableDTSMember
- EsbOtlGetEnabledDTSMembers
- EsbOtlGetDTSMemberAlias

EsbOtlSetGenName

次元内の特定世代に対して名前を設定します。

構文

```
      EsbOtlSetGenName
       (
      hOutline, pszDimension, usGen, pszName
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      usGen
         As Integer
ByVal
      pszName
        As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象の世代を含む次元の名前。
usGen	名前を設定する世代の番号。次元自体は世代 1 です。
pszName	世代に与える名前。

備考

- 世代名はメンバー名と同じルールに従い、メンバー名全体で一意性が必要です。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- 各特定次元および世代で持つことができる名前は 1 つです。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_GENLEVELNAME
- OTLAPI_ERR_GENLEVELNAMEEXISTS
- OTLAPI_ERR_GENLEVELEXISTS
- OTLAPI_ERR_GENLEVELVALUE
- LAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    Declare Function EsbOtlSetGenName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension As String,
ByVal usGen As Integer, ByVal pszName As String) As Long

Sub ESB_OtlSetGenName()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Dimension As String
Dim GenNum As Integer
Dim GenName As String
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
'***************************************
'********** Set Generation Name **********
'***************************************
Dimension = "Year"
GenNum = 2
GenName = "Qtr1 Qtr2 Qtr3 Qtr4"
If Not sts Then
  sts = EsbOtlSetGenName(hOutline, Dimension,
GenNum, GenName)
End If
End Sub
```

関連トピック

- EsbOtlDeleteGenName
- EsbOtlGetGenNames
- EsbOtlGetGenName

EsbOtlSetLevelName

次元内の特定レベルに対して名前を設定します。

構文

```
      EsbOtlSetLevelName
       (
      hOutline, pszDimension, usLevel, pszName
      )
ByVal
      hOutline
        As Long
ByVal
      pszDimension
       As String
ByVal
      usLevel
         As Integer
ByVal
      pszName
         As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pszDimension	対象のレベルを含む次元の名前。
usGen	名前を設定するレベルの番号。リーフ・メンバーはレベル 0 です。
pszName	レベルに与える名前。

備考

- レベル名はメンバー名と同じルールに従い、メンバーの名前スペース全体で一意の必要があります。他の世代、レベル、メンバー名、または別名と重複できません。重複した名前を追加しようとすると、エラーが発生します。
- 個々の特定次元およびレベルが持てる名前は 1 つです。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_GENLEVELNAME
- OTLAPI_ERR_GENLEVELNAMEEXISTS
- OTLAPI_ERR_GENLEVELEXISTS
- OTLAPI_ERR_NOTADIM
- OTLAPI_ ERR_GENLEVELNAMEMBR

例

```
    Declare Function EsbOtlSetLevelName Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal pszDimension As String,
ByVal usLevel As Integer, ByVal pszName As String) As Long

Sub ESB_OtlSetLevelName()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Dimension As String
Dim LevelNum As Integer
Dim LevelName As String
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
ESB_YES, hOutline)
'***********************************
'********** Set Level Name **********
'***********************************
Dimension = "Year"
LevelNum = 1
LevelName = "Month"
If Not sts Then
 sts = EsbOtlSetLevelName(hOutline,
  Dimension, LevelNum, LevelName)
End If
End Sub
```

関連トピック

- EsbOtlDeleteLevelName
- EsbOtlGetLevelName
- EsbOtlGetLevelNames

EsbOtlSetMemberAlias

指定した別名テーブルの指定したメンバーに対して、デフォルトのメンバー別名を設定します。

構文

```
     EsbOtlSetMemberAlias
      (
     hOutline, hMember, pszAliasTable, pszAlias
     )
ByVal
     hOutline
        As Long
ByVal
     hMember
```

```
        As Long
ByVal
     pszAliasTable
      As String
ByVal
     pszAlias
        As String
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
hMember	別名を設定するメンバーのハンドル。
pszAliasTable	別名を設定する別名テーブル。このパラメータが""の場合、デフォルトの別名テーブルが使用されます。
pszAlias	別名。

備考

- メンバー・ハンドルは共有メンバーにできません。共有メンバーに別名を使用できません。
- 別名を削除するには、**EsbOtlDeleteMemberAlias()**を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_ALIASTABLE
- OTLAPI_ERR_ALIASSHARED
- OTLAPI_ERR_ILLEGALDEFALIAS
- OTLAPI_ERR_ILLEGALCOMBOALIAS
- OTLAPI_ERR_ILLEGALALIASSTRING
- OTLAPI_ERR_DUPLICATEALIAS

例

```
    Declare Function EsbOtlSetMemberAlias Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszAliasTable As String, ByVal pszAlias As String) As Long

Sub ESB_OtlSetMemberAlias()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberYear As Long
Dim szAlias As String * ESB_MBRNAMELEN
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
```

```
sts = EsbOtlOpenOutline(hCtx, Object,
SB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Year",
  hMemberYear)
End If
If sts = 0 And hMemberYear <> 0 Then
 szAlias = "Year Dimension"
 sts = EsbOtlSetMemberAlias(hOutline,
  hMemberYear, "", szAlias)
End If
End Sub
```

関連トピック

- EsbOtlGetMemberAlias
- EsbOtlDeleteMemberAlias

EsbOtlSetMemberFormula

指定されたメンバーに対して式を設定します。

構文

```
      EsbOtlSetMemberFormula
       (
      hOutline, hMember, pszFormula
      )
ByVal
      hOutline
       As Long
ByVal
      hMember
        As Long
ByVal
      pszFormula
       As String
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　メンバーのハンドル。

pszFormula メンバー式を含むバッファ。

備考

メンバー式を削除するには、**EsbOtlDeleteMemberFormula()**を使用します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_ERR_SHAREDMEMBERFORMULA
- OTLAPI_ERR_MEMBERCALC

例

```
    Declare Function EsbOtlSetMemberFormula Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszFormula As String) As Long

Sub ESB_OtlSetMemberFormula()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMemberProfit As Long
Dim szFormula as String * 100
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit",
 hMemberProfit)
End If
If sts = 0 hMemberProfit <> 0 Then
 szFormula = "Profit = Gross / Margin;"
 sts = EsbOtlSetMemberFormula(hOutline,
 hMemberProfit, szFormula)
End If
End Sub
```

関連トピック

- EsbOtlGetMemberFormula
- EsbOtlDeleteMemberFormula

EsbOtlSetMemberInfo

この関数は、メンバー属性情報を設定します。

構文

```
     EsbOtlSetMemberInfo
      (
     hOutline, hMember, pInfo
     )
ByVal
     hOutline
      As Long
ByVal
     hMember
      As Long
```

```
pInfo
  As ESB_MBRINFO_T
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　属性を設定するメンバーのハンドル。

pInfo　メンバー情報の構造体。

備考

- **EsbOtlGetMemberInfo()**を呼び出して、ESB_MBRINFO_T 構造体の各フィールドを初期化する必要があります。
- 属性:
  - ESB_MBRINFO_T 構造体の、次の 2 つのフィールドは属性専用です:

データ型	フィールド	説明
As Variant	Attribute	属性値: 属性次元またはゼロレベル(リーフ・ノード)の属性メンバーについては、次のデータ型のいずれかになります: ❏ ESB_ATTRMBRDT_BOOL ❏ ESB_ATTRMBRDT_DATETIME ❏ ESB_ATTRMBRDT_DOUBLE ❏ ESB_ATTRMBRDT_STRING 属性次元ではなく、属性メンバーの場合: ❏ ESB_ATTRMBRDT_NONE ❏ ESB_ATTRMBRDT_AUTO
ESB_ATTRMBRDT_AUTO は、メンバーを追加する場合にのみ使用してください。「属性メンバーの追加に関する注意」を参照してください。		
As Integer	IsAttributed	メンバーに属性が関連付けられているかどうかを示します。

  - ESB_MBRINFO_T 構造体の、次の 2 つのフィールドの値は属性専用です:

データ型	フィールド	説明
As Integer	usCategory	次の次元カテゴリのいずれかになります: ❏ ESB_CAT_ATTRIBUTE ❏ ESB_CAT_ATTRCALC (内部での使用専用)

データ型	フィールド	説明
As Integer	usStorageCategory	次の次元ストレージ・カテゴリのいずれかになります: ❑ ESB_STORECAT_ATTRIBUTE ❑ ESB_STORECAT_ATTRCALC (内部での使用専用)

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_CONSOL
- OTLAPI_BAD_MBRNAME
- OTLAPI_BAD_MEMBER
- OTLAPI_ERR_ADDNAMEUSED
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_BADSHARE
- OTLAPI_ERR_BADSKIP
- OTLAPI_ERR_BADSTORAGE
- OTLAPI_ERR_BADSTORAGECATEGORY
- OTLAPI_ERR_BADTIMEBAL
- OTLAPI_ERR_ILLEGALBOOLEAN
- OTLAPI_ERR_ILLEGALCURRENCY
- OTLAPI_ERR_ILLEGALDATE
- OTLAPI_ERR_ILLEGALNAME
- OTLAPI_ERR_ILLEGALNUMERIC
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_LEAFLABEL
- OTLAPI_ERR_NOSHAREPROTO
- OTLAPI_ERR_NOTIMEDIM
- OTLAPI_ERR_SHARENOTLEVEL0

例

```
    Declare Function EsbOtlSetMemberInfo Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
pInfo As ESB_MBRINFO_T) As Long

Sub ESB_OtlSetMemberInfo()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim MbrInfo As ESB_MBRINFO_T
Dim hFeb As Long
```

```
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Feb", hFeb)
End If
If sts = 0 And hFeb <> 0 Then
 MbrInfo.fTwoPass = ESB_TRUE
 MbrInfo.fExpense = ESB_TRUE
 MbrInfo.usTimeBalance = ESB_TIMEBAL_AVG
 MbrInfo.usSkip = ESB_SKIP_ZEROS
 MbrInfo.usConsolidation = ESB_UCALC_MULT
 sts = EsbOtlSetMemberInfo(hOutline, hFeb, MbrInfo)
End If
End Sub
```

関連トピック

- EsbOtlGetMemberInfo
- EsbOtlFindMember

EsbOtlSetOutlineInfo

アウトライン情報を設定します。

構文

```
      EsbOtlSetOutlineInfo (
      hOutline, pInfo
      )
ByVal
      hOutline
       As Long

      pInfo
        As ESB_OUTLINEINFO_T
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

pInfo　呼出し元によって割り当てられる、アウトライン情報のストレージ変数。

備考

- ESB_OUTLINEINFO_T 構造体の一部のフィールドのみを使用して情報を設定します。詳細は「API 構造体」のセクションを参照してください。
- ESB_OUTLINEINFO_T 構造体の各フィールドを初期化するには、**EsbOtlGetOutlineInfo()**を呼び出します。

- ESB_OUTLINEINFO_T 構造体の fCaseSensitive フラグが ESB_TRUE から ESB_FALSE に変更され、そのためにメンバー名が重複してしまった場合、この呼出しは失敗します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OUTLINETYPE
- OTLAPI_ERR_DUPLICATEALIAS
- OTLAPI_ERR_CURTOOMANYDIMS
- OTLAPI_ERR_ILLEGALTAG
- OTLAPI_ERR_DUPLICATENAME

例

```
    Declare Function EsbOtlSetOutlineInfo Lib
"ESBOTLN" (ByVal hOutline As Long,
pInfo As ESB_OUTLINEINFO_T) As Long

Sub ESB_OtlSetOutlineInfo()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim Info As ESB_OUTLINEINFO_T
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, hOutline)
'call GetOutlineInfo() to fill structure
If sts = 0 Then
  Info.fCaseSensitive = ESB_FALSE
  sts = EsbOtlSetOutlineInfo(hOutline, Info)
End If
End Sub
```

関連トピック

- EsbOtlGetOutlineInfo

EsbOtlSetUserAttribute

メンバーに対するユーザー定義属性を設定します。

構文

```
EsbOtlSetUserAttribute
(
```

```
      hOutline, hMember, pszString
      )

ByVal
      hOutline
       As Long
ByVal
      hMember
        As Long
ByVal
      pszString
       As String
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hMember　ユーザー定義属性を設定するメンバーのハンドル。

pszString　設定するユーザー定義属性。

備考

- 呼出し元はメンバーに任意の数のユーザー定義属性を設定できます。一意に渡された文字列が各属性を定義し、ユーザー名と同様の表記規則に従います。**EsbOtlGetUserAttributes()**を参照してください。
- 共有メンバーに対してユーザー属性を設定しようとすると、エラーが発生します。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_USERATTR
- OTLAPI_ERR_SHAREUDA

例

```
    Declare Function EsbOtlSetUserAttribute Lib
"ESBOTLN" (ByVal hOutline As Long, ByVal hMember As Long,
ByVal pszString As String) As Long

Sub ESB_OtlSetUserAttribute()
Dim sts As Long Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hMember As Long
Dim AttributeList As String
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
AttributeList = "Read Write"
sts = EsbOtlOpenOutline(hCtx, Object, ESB_YES,
```

```
ESB_YES, hOutline)
If sts = 0 Then
  sts = EsbOtlFindMember(hOutline, "Jan",
  hMember)
End If
If sts = 0 And hMember <> 0 Then
  '********************
  ' Set User Attributes
  '********************
  sts = EsbOtlSetUserAttribute(hOutline,
  hMember, AttributeList)
End If
End Sub
```

関連トピック

- EsbOtlDeleteUserAttribute
- EsbOtlGetUserAttributes

EsbOtlSortChildren

アウトライン・メンバーの子をソートします。

構文

```
      EsbOtlSortChildren
       (
      hOutline, hParent, usType
      )
ByVal
      hOutline
       As Long
ByVal
      hParent
       As Long
ByVal
      usType
        As Integer
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

hParent　ソートする子の親のハンドル。ESB_NULL が指定されている場合は、次元がソートされます。

usType　ソート・タイプ。次のいずれかを指定できます:

- ESB_SORT_ASCENDING
- ESB_SORT_DESCENDING

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_SORTTYPE
- OTLAPI_BAD_SORTCOMPAREFUNC
- OTLAPI_SORT_TOOMANY

例

```
    Declare Function EsbOtlSortChildren Lib
"ESBOTLW" (ByVal hOutline As Long, ByVal hParent As Long,
ByVal usType As Integer) As Long
Sub ESB_OtlSortChildren()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim hParent As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
If sts = 0 Then
 sts = EsbOtlFindMember(hOutline, "Profit", hParent)
End If
If sts = 0 And hParent <> 0 Then
 sts = EsbOtlSortChildren(hOutline,
  hParent, ESB_SORT_DESCENDING)
End If
End Sub
```

関連トピック

- EsbOtlFindMember

EsbOtlVerifyOutline

アウトラインが正しいことを確認します。グローバルなアウトライン・エラーと、
不正なメンバーそれぞれのエラーの両方を戻します。

構文

```
     EsbOtlVerifyOutline
      (
     hOutline, pulErrors, pulCount
     )
ByVal
     hOutline
      As Long

     pulErrors
      As Long

     pulCount
```

```
        As Long
```

パラメータ	説明
hOutline	アウトラインのコンテキスト・ハンドル。
pulErrors	グローバル・アウトライン・エラーの戻り値のビットマスクを示す戻り変数。現在、このフィールドには次の値のみがあります: ESB_OUTERROR_CURTOOMANYDIMS
pulCount	エラーのあるメンバーの数。

備考

- この関数は、共有メンバーにおけるユーザー属性の重複、レベルまたは世代名、別名の重複がないかどうかを確認します。
- サーバーへのアウトラインの保存は、そのアウトラインにエラーがない場合のみ成功します(*pulErrors == 0 かつ*pulCount == 0)。

➤ エラー値を取得するには、次の手順を実行します:

1 **ESB_OUTERROR_T 構造体を割り当てます。**

2 **各エラー・メンバーごとに 1 回 EsbGetNextItem()を呼び出します(pulCount 変数で戻されます)。**

   EsbGetNextItem()を呼び出すたびに、メンバーのエラー情報が ESB_OUTERROR_T 構造体で戻されます。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- ESB_OUTERROR_SHAREUDA
- ESB_OUTERROR_DUPGENLEVNAME

例

```
    Declare Function EsbOtlVerifyOutline Lib
"ESBOTLN" (ByVal hOutline As Long, pulErrors As Long,
pulCount As Long) As Long

Sub ESB_OtlVerifyOutline()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Dim ulErrors As Long
Dim ulCount As Long
Dim pOutError As ESB_OUTERROR_T
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
```

```
'body of code
If sts = 0 Then
 sts = EsbOtlVerifyOutline(hOutline,
  ulErrors, ulCount)
 Do While sts = 0 And ulCount > 0
  sts = EsbGetNextItem(hCtx,
   ESB_OUTERROR_TYPE, pOutError)
   ulCount = ulCount - 1
   'do something with the error value
 Loop
End If
End Sub
```

関連トピック

- EsbOtlNewOutline
- EsbOtlOpenOutline
- EsbOtlWriteOutline

EsbOtlWriteOutline

既存のアウトライン情報をディスクに書き込みます。

構文

```
      EsbOtlWriteOutline (
      hOutline, pObject
      )
ByVal
      hOutline
       As Long

      pObject
       As ESB_OBJDEF_T
```

パラメータ 説明

hOutline　アウトラインのコンテキスト・ハンドル。

pObject　書込み対象のアウトライン・オブジェクト。

備考

- アウトラインをサーバー・オブジェクトとして保存する場合は、最初に.OTN ファイルとして保存されます。そして **EsbOtlRestructure()**を呼び出して、実際の.OTL ファイルを作成する必要があります。
- アウトラインをサーバー・オブジェクトとして保存する場合、オブジェクト名はデータベース名と同じである必要があります。
- サーバー・アウトライン・オブジェクトまたはクライアント・アウトライン・オブジェクトをローカル・データベースに保存する場合、データベースがすでに存在している必要があります。

- 指定されたユーザーによってアウトラインが現在ロックされていない場合、この呼出しは正常に実行されません(ESB_OBJDEF_T 構造体の hCtx パラメータ)。

戻り値

正常終了の場合は 0 が戻されます。それ以外の場合は、次のいずれかの値が戻されます:

- OTLAPI_BAD_OBJTYPE
- OTLAPI_ERR_NOTVERIFIED

アクセス

この関数を使用するには、呼出し元は指定したアプリケーション、アウトライン・オブジェクトが含まれているデータベースのいずれか、またはその両方に対して、適切なレベルのアクセス権を持っている必要があります。アウトライン・オブジェクトを書き込むには、指定したアプリケーション、またはアウトラインが含まれるデータベースに対して、アプリケーション・デザイナまたはデータベース・デザイナの権限(ESB_PRIV_APPDESIGN または ESB_PRIV_DBDESIGN)が必要です。

例

```
    Declare Function EsbOtlWriteOutline Lib
"ESBOTLN" (ByVal hOutline As Long,
pObject As ESB_OBJDEF_T) As Long

Sub ESB_OtlWriteOutline()
Dim sts As Long
Dim Object As ESB_OBJDEF_T
Dim hOutline As Long
Object.hCtx = hCtx
Object.Type = ESB_OBJTYPE_OUTLINE
Object.AppName = "Sample"
Object.DbName = "Basic"
Object.FileName = "Basic"
sts = EsbOtlOpenOutline(hCtx, Object,
ESB_YES, ESB_YES, hOutline)
'body of code
If sts = 0 Then
  sts = EsbOtlWriteOutline(hOutline, Object)
End If
'restructure db using EsbOtlRestructure()
End Sub
```

関連トピック

- EsbOtlOpenOutline
- EsbOtlNewOutline
- EsbOtlVerifyOutline
- EsbOtlRestructure
- EsbOtlCloseOutline

24 アウトラインの走査例(VB)

この例では、アウトライン・ツリーの使用方法を示します。TraverseTree は再帰的アルゴリズムで、アウトライン・ツリーを走査してすべてのアウトライン・メンバーへのアクセスを提供します。順番に各メンバーを選択して、最終メンバーに到達するまでそれぞれで処理します。コード中のコメントは、追加の処理ができることを示します。

このアルゴリズムには、複数の VB のアウトライン API コマンドが組み込まれています。

- **EsbOtlGetFirstMember()**は、アウトライン中の最初のメンバー(最初に定義されている次元)へのメンバーのハンドルを戻します。
- **EsbOtlGetMemberInfo()**は、指定されたメンバーに関する情報を取得します。
- **EsbOtlGetChild()**は、メンバーの子を戻します。
- **EsbOtlGetNextSibling()**は、メンバーの次の兄弟を戻します。

このコードを実行する前に、API を初期化してアウトラインを開きます。このコードの後に、アウトラインを閉じて API を終了します。

```
   TraverseTree (ESB_HOUTLINE_T)
{
 ESB_HMEMBER_T hMember;
 ESB_STS_T  sts = 0;

 sts = EsbOtlGetFirstMember(hOutline, &hMember);
 if (!sts && hMember)
  sts = TraverseTreeRecurse(hOutline, hMember);
}

TraverseTreeRecurse(ESB_HOUTLINE_T hOutline, ESB_HMEMBER_T hMember)
{
 ESB_MEMBERINFO_T MbrInfo;
 ESB_HMEMBER_T, hChild;
 ESB_STS_T  sts = 0;

 while (!sts && hMember)
 {
  sts = EsbOtlGetMemberInfo (hOutline, hMember, &MbrInfo);

   /* ADD THE PROCESSING FOR EACH MEMBER HERE. */

   if (!sts)
   {
    sts = EsbOtlGetChild(hOutline, hMember, &hChild);
```

```
   if (!sts && hChild)
   {
    sts = TraverseTreeRecurse(hOutline, hChild);
   }
  }
  sts = EsbOtlGetNextSibling(hOutline, hMember, &hMember);
 }
 return (sts);
}
```

第 V I I 部
その他のAPI

その他の API の内容：

25 Java APIリファレンス

Java API ドキュメントは、Oracle Hyperion Provider Services のインストールに Javadoc として含まれています(場所は、`EPM_ORACLE_INSTANCE\common\docs\en\aps`、および `EPM_ORACLE_INSTANCE\common\EssbaseAPI\release\docs\en\aps`)。また、Essbase Java API リファレンスは、Oracle Technology Network で入手できます。

26 MDXプロバイダAPI

この章の内容

MDX プロバイダ API 一般情報

MDX 機能指定で指定された構文に従う MDX クエリーは、MDX-API によってサーバーに提出できます。クエリーの結果は、この API を使用して、クライアントによって取得できます。

MDX ステートメントの構文は、『Oracle Essbase テクニカル・リファレンス』の MDX のセクションで説明しています。

いくつかの基本的な MDX の概念および用語をここで説明します。MDX クエリーは、いくつかの軸仕様およびオプションのスライサ仕様からなります。各軸は、集合値式を指定します。集合は、タプルの順序付けられたコレクションであり、タプルは、1 つ以上の次元メンバーの連続です。集合内のタプルは、次元性が均質です(各タプルには、同じ順序で同じ次元からのメンバーが含まれます)。

Sample Basic データベースに基づく集合式の 1 例は次のとおりです:

```
   Union(
  CrossJoin({[Sales], [Profit]}, {[Actual], [Budget]}),
  Union(
   CrossJoin([Total Expenses].Children, {[Actual]}),
   {([Opening Inventory], [Variance]), ([Additions], [Variance %])}
  )
 )
```

この式は、Union、CrossJoin、Children といった MDX 関数を使用します。この式の値は集合です:

```
   {
  ([Sales], [Actual]),
  ([Sales], [Budget]),
  ([Profit], [Actual]),
  ([Profit], [Budget]),
  ([Marketing], [Actual]),
  ([Payroll], [Actual]),
  ([Misc], [Actual]),
```

```
  ([Opening Inventory], [Variance]),
  ([Additions], [Variance %])
 }
```

CrossJoin の結果で、最初の次元の変化が最も遅くなるように、タプルが順序付けられていることに注意してください。この集合内のタプルには([Measures]、[Scenario])という次元性があります。軸集合全体にわたってタプルの次元性は重複してはいけません。

集合式に加えて、各軸は、軸名 (COLUMNS、ROWS、PAGES など)または軸番号(AXIS(0)、AXIS(1)など)を指定します。各軸から 1 つずつ、タプルのあらゆる可能な組合せからなるキューブが、クエリーの結果を構成します。軸およびスライサにない次元は、デフォルトでは、結果キューブを定義するときに自身のルート・メンバーを含むことになります。スライサ(存在する場合)は、単一のタプルで、集合を指定します。これは、それぞれの次元に関係のあるメンバーを識別します。これにより、最終結果を、軸から作成された 1 枚のキューブにします。MDX クエリーの結果は、結果キューブ内のセルのデータ値のみでなく、各軸およびスライサに関するメタデータを含んでいます。

完全な MDX クエリーを次に示します:

```
   SELECT
  Union(
   CrossJoin({[Sales], [Profit]}, {[Actual], [Budget]}),
   Union(
    CrossJoin([Total Expenses].Children, {[Actual]}),
    {([Opening Inventory], [Variance]), ([Additions], [Variance %])}
   )
  ) ON COLUMNS,
  CrossJoin(
   [200].Children,
   {[East], [West]}
  ) ON ROWS
 FROM
  Sample.Basic
 WHERE
  {[Jan]}
```

このクエリーの結果では、列軸に 9 個のタプルがあり、行軸に 8 個のタプルがあります。つまり、全部で 72 個のセルがあります。各セルには序数(すなわちオフセット)があり、これは各軸に沿ったタプルの位置により異なります。オフセットおよび位置は 0 から始まります。最初の軸位置が最も速く変化するように、セルは順序付けられています。

たとえば、列軸のタプル 3 および行軸のタプル 4 によって識別されるセルは、3 + 9*4 = 39 のオフセットに位置します。

- 列軸のタプル 3 は([Profit]、[Budget])です。
- 行軸のタプル 4 は([200-30]、[East])です。
- したがって、セル 39 は([Profit]、[Budget]、[200-30]、[East]、[Jan])になります。

クラスタの概念は、効率のために必要です。集合は、タプルの順序付けられたコレクションであると考えることができます。または、クラスタの順序付けられたコレクションと考えることができます。クラスタは、各集合の次元からのメンバーのあらゆる組合せを含むタプルのコレクションです。CrossJoin 関数の出力と同様に(最初の次元の変化が最も遅くなるように)、タプルを順序付ける必要があります。CrossJoin 関数の使用により、クラスタが作成されますが、サーバーは、他の関数の結果からでもクラスタを特定できます。

MDX プロバイダ API リファレンス

MDX クエリー処理用の C API は、既存の Essbase API と連携するよう設計されています。クライアント・プログラムには API 固有の様々な構造体へのハンドルが提供され、メソッドを使用してコンポーネントにアクセスします。関数の数は通常必要とされる出力結果を適切に組み合せて、少数に維持されています。API によって内部構造体に割り当てられたメモリーは、特に指定がないかぎり、クライアントがクエリー解放関数を呼び出したときに解放されます。ESS_MDX はハンドル・タイプに使用される接頭辞で、EssMdx は MDX-API が実装する関数に使用される接頭辞です。

MDX プロバイダの宣言

型宣言は次のとおりです:

```
   typedef void *ESS_MDX_QRYHDL_T;         /* MDX query handle */
typedef unsigned long ESS_MDX_MEMBERIDTYPE_T;  /* MDX mbr id type */
typedef void *ESS_MDX_AXISHDL_T;         /* MDX axis handle */
typedef void *ESS_MDX_DIMHDL_T;         /* MDX dim handle */
typedef unsigned long ESS_MDX_PROPTYPE_T;    /* MDX property type */
typedef void *ESS_MDX_PROPHDL_T;         /* MDX property handle */
typedef void *ESS_MDX_CLUSTERHDL_T;       /* MDX cluster handle */
typedef void *ESS_MDX_MBRHDL_T;         /* MDX mbr handle */
typedef void *ESS_MDX_CELLHDL_T;         /* MDX cell handle */
typedef unsigned long ESS_MDX_CELLSTATUS_T;   /* MDX cell status */
```

定数宣言は次のとおりです:

```
   /* MDX member identifier types (ESS_MDX_MEMBERIDTYPE_T) */
#define ESS_MDX_MEMBERIDTYPE_NAME      8
#define ESS_MDX_MEMBERIDTYPE_ALIAS     16

/* MDX property value types (ESS_MDX_PROPTYPE_T) */
#define ESS_MDX_PROPTYPE_BOOL        ESS_DT_BOOL
#define ESS_MDX_PROPTYPE_DOUBLE       ESS_DT_DOUBLE
#define ESS_MDX_PROPTYPE_DATETIME      ESS_DT_DATETIME
#define ESS_MDX_PROPTYPE_STRING       ESS_DT_STRING
#define ESS_MDX_PROPTYPE_ULONG       ESS_DT_ULONG
#define ESS_MDX_PROPTYPE_NONE        0

/* MDX cell status bitmasks (ESS_MDX_CELLSTATUS_T) */
```

```
#define ESS_MDX_CELLSTATUS_LINKEDOBJS     0x00000001
#define ESS_MDX_CELLSTATUS_DYNCALC       0x00000002
#define ESS_MDX_CELLSTATUS_CALCEDMBR      0x00000004
#define ESS_MDX_CELLSTATUS_READONLY      0x00000008

/* MDX cell property bitmasks (ESS_MDX_CELLPROP_T) */
#define ESS_MDX_CELLPROP_GLDRILLTHRU     0x00000008
```

ESS_MDX_PROPVALUE_T

```
   typedef struct ess_mdx_propvalue_t
{
 ESS_MDX_PROPTYPE_T ulPropType;  /* ESS_MDX_PROPTYPE_XXXX */
 union
 {
  ESS_BOOL_T bData;    /* Boolean value */
  ESS_ULONG_T ulData;   /* Ulong value */
  ESS_STR_T strData;    /* String value */
  ESS_DATETIME_T dtData;  /* Datetime value */
  ESS_DOUBLE_T dblData;  /* Double value */
 } value;
} ESS_MDX_PROPVALUE_T;
```

ESS_MDX_CELLVALUE_T

```
   typedef struct mdxcellvalue
{
 ESS_DOUBLE_T    dblVal;
 ESS_STR_T     fmtVal;
 ESS_STR_T     fmtStr;
 ESS_USHORT_T    smId;
 ESS_USHORT_T    type;
 ESS_ULONG_T    flags;   // captures drill through property.
} ESS_MDX_CELLVALUE_T;
```

EssMdxExecuteQuery

現在接続されているデータベース上で、指定されたクエリーを実行します。

構文

```
   ESS_FUNC_M EssMdxExecuteQuery(
 ESS_MDX_QRYHDL_T  hQry);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル

注意

この関数は、先に MDX クエリー by calling EssMDXNewQuery を呼び出して MDX クエリーを作成してから呼び出す必要があります。

EssMdxFreeQuery

指定されたクエリーに使用されたメモリーを解放します。

構文

```
  ESS_FUNC_M EssMdxFreeQuery(
ESS_MDX_QRYHDL_T  hQry);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル

EssMdxGetAxes

クエリー内の軸に関する情報を戻します。

送信するクエリーで軸の性質に関する情報を取得するには、1762 ページの「EssMdxExecuteQuery」を呼び出した後に次の API を使用します:

- 1763 ページの「EssMdxGetAxes」
- 1764 ページの「EssMdxGetAxisInfo」
- 1770 ページの「EssMdxGetDimInfo」

構文

```
  ESS_FUNC_M EssMdxGetAxes(
ESS_MDX_QRYHDL_T    hQry,
ESS_PULONG_T      pulNAxes,
ESS_MDX_PPAXISHDL_T  pphAxes,
ESS_MDX_PAXISHDL_T   phSlicer);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
pulNAxes	出力	軸の数
pphAxes	出力	軸ハンドルの配列
phSlicer	出力	スライサ軸ハンドル

EssMdxGetAxisInfo

指定された軸に関する情報を戻します。

送信するクエリーで軸の性質に関する情報を取得するには、1762 ページの「EssMdxExecuteQuery」 を呼び出した後に次の API を使用します:

- 1763 ページの「EssMdxGetAxes」
- 1764 ページの「EssMdxGetAxisInfo」
- 1770 ページの「EssMdxGetDimInfo」

構文

```
  ESS_FUNC_M EssMdxGetAxisInfo(
ESS_MDX_AXISHDL_T   hAxis,
ESS_PULONG_T     pulSize,
ESS_PULONG_T     pulNDims,
ESS_MDX_PPDIMHDL_T  pphDims);
```

パラメータ	タイプ	説明
hAxis	入力	軸ハンドル
pulSize	出力	軸の中のタプルの数
pulNDims	出力	軸の中の次元の数
pphDims	出力	次元ハンドルの配列

EssMdxGetAxisMembers

指定された軸の指定された位置でタプルを戻します。この関数を使用して、軸から特定のタプルを直接取得します。

注： クライアントは、`pphMbrs` で終了するときには、**EssFree()**を使用する必要があります。

軸セットのコンテンツに関する情報を取得するには、次の API を使用します:

- 1769 ページの「EssMdxGetClusters」
- 1768 ページの「EssMdxGetClusterInfo」
- 1769 ページの「EssMdxGetClusterMembers」
- 1764 ページの「EssMdxGetAxisMembers」
- 1772 ページの「EssMdxGetMbrIdentifier」
- 1772 ページの「EssMdxGetMbrProperty」

構文

```
  ESS_FUNC_M EssMdxGetAxisMembers(
ESS_MDX_AXISHDL_T   hAxis,
ESS_ULONG_T      ulIndex,
ESS_MDX_PPMBRHDL_T  pphMbrs);
```

パラメータ	タイプ	説明
hAxis	入力	軸ハンドル
ulIndex	入力	軸の中のタプルの位置
pphMbrs	出力	タプルについてのメンバーのハンドルの配列

EssMdxGetCellAtIndices

指定されたタプルのインデックスの交差部分におけるセルを戻します。

クエリーで、軸から形成されたキューブのセル値を取得するには、次の API を使用します:

- 1765 ページの「EssMdxGetCellAtOffset」
- 1765 ページの「EssMdxGetCellAtIndices」
- 1776 ページの「EssMdxGetValue」

構文

```
  ESS_FUNC_M EssMdxGetCellAtIndices(
ESS_MDX_QRYHDL_T   hQry,
ESS_PULONG_T     pulIndices,
ESS_MDX_PCELLHDL_T  phCell);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
pulIndices	入力	タプルのインデックス(各軸につき 1 つずつ)
phCell	出力	セル・ハンドル

EssMdxGetCellAtOffset

指定されたオフセットにおけるセルを戻します。

クエリーで、軸から形成されたキューブのセル値を取得するには、次の API を使用します:

- 1765 ページの「EssMdxGetCellAtOffset」

- 1765 ページの「EssMdxGetCellAtIndices」
- 1776 ページの「EssMdxGetValue」

構文

```
  ESS_FUNC_M EssMdxGetCellAtOffset(
ESS_MDX_QRYHDL_T   hQry,
ESS_ULONG_T     ulOffset,
ESS_MDX_PCELLHDL_T  phCell);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
ulOffset	入力	セルのオフセット(最初の軸の変更が最も早いものになります)
phCell	入力	セル・ハンドル

EssMdxGetCellInfo

入力セル・ハンドルに対応するセルのタイプを戻します。

構文

```
 ESS_FUNC_M EssMdxGetCellInfo (
ESS_MDX_CELLHDL_T    hCell,
ESS_PULONG_T      pUlType,
ESS_MDX_PCELLINFO_T   pulCellInfo,
ESS_MDX_PCELLSTATUS_T  pulStatus);
```

パラメータ	タイプ	説明
hCell	入力	セル・ハンドル
pulType	出力	セルのデータ型。値: ● ESS_MDX_VALTYPE_DOUBLE 数値型 ● ESS_MDX_VALTYPE_SMARTLIST スマートリスト・タイプ ● ESS_MDX_VALTYPE_DATE 日付型

パラメータ	タイプ	説明
pulCellInfo	出力	次のビットマスクを使用して指定されたセル・ステータス・ビット・マップ: ● ESS_MDX_CELLINFO_MISSING セル値がありません ● ESS_MDX_CELLINFO_NOACCESS 現在のユーザーはセル値にアクセスできません。 ● ESS_MDX_CELLINFO_MEANINGLESS 属性メンバーのコンテキストではセル値は意味がありません ● ESS_MDX_CELLINFO_OUTOFRANGE スマートリストのコンテキストでセル値が範囲外です
pulStatus	出力	セル・ステータス情報。これは EssMdxGetCellStatus 関数から戻される情報と同じです。詳細は、関数の説明を参照してください。ステータス情報は、関数 EssMdxSetNeedCellStatus が呼ばれたときにのみ戻されます。

EssMdxGetCellStatus

hCell で指定されたセルのステータスを戻します。ステータスは pulStatus のビットマスクと照らし合わせてテストでき、セルが対応するタイプであるかどうか判定できます。この関数は、1780 ページの「EssMdxSetNeedCellStatus」への以前の呼出しの後にのみ呼び出す必要があります。

構文

```
  ESS_FUNC_M EssMdxGetCellStatus(
ESS_MDX_QRYHDL_T     hQry,
ESS_MDX_CELLHDL_T    hCell,
ESS_MDX_PCELLSTATUS_T  pulStatus);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
hCell	入力	セル・ハンドル
pulStatus	出力	セルのステータス: 次のマスクを持つビットマップ: ● ESS_MDX_CELLSTATUS_LINKEDOBJS ● ESS_MDX_CELLSTATUS_DYNCALC ● ESS_MDX_CELLSTATUS_CALCED ● ESS_MDX_CELLSTATUS_READONLYMBR

EssMdxGetClusterDimMembers

指定したクラスタの中で、指定された次元についてメンバーのハンドルを戻します。

構文

```
  ESS_FUNC_M EssMdxGetClusterDimMembers(
ESS_MDX_CLUSTERHDL_T  hCluster,
ESS_ULONG_T      ulIndex,
ESS_MDX_PPMBRHDL_T   pphMbrs);
```

パラメータ	タイプ	説明
hCluster	入力	クラスタ・ハンドル
ulIndex	入力	クラスタを含む軸の中の次元インデックス
pphMbrs	出力	指定された次元についてのメンバーのハンドルの配列

EssMdxGetClusterInfo

指定されたクラスタに関する情報を戻します。

軸セットのコンテンツに関する情報を取得するには、次の API を使用します:

- 1769 ページの「EssMdxGetClusters」
- 1768 ページの「EssMdxGetClusterInfo」
- 1769 ページの「EssMdxGetClusterMembers」
- 1764 ページの「EssMdxGetAxisMembers」
- 1772 ページの「EssMdxGetMbrIdentifier」
- 1772 ページの「EssMdxGetMbrProperty」

構文

```
  ESS_FUNC_M EssMdxGetClusterInfo(
ESS_MDX_CLUSTERHDL_T  hCluster,
ESS_PULONG_T      pulSize,
ESS_PULONG_T      pulNDims,
ESS_PPULONG_T     ppulDimSizes);
```

パラメータ	タイプ	説明
hCluster	入力	クラスタ・ハンドル
pulSize	出力	クラスタの中のタプルの数
pulNDims	出力	クラスタの中の次元の数(このクラスタが属する軸における次元の数と同じ)

パラメータ	タイプ	説明
ppulDimSizes	出力	次元のサイズ(メンバーの数)の配列

EssMdxGetClusterMembers

指定したクラスタの中の、指定された位置におけるタプルを戻します。

注： クライアントは、pphMbrs で終了するときには、**EssFree()**を使用する必要があります。

軸セットのコンテンツに関する情報を取得するには、次の API を使用します:

- 1769 ページの「EssMdxGetClusters」
- 1768 ページの「EssMdxGetClusterInfo」
- 1769 ページの「EssMdxGetClusterMembers」
- 1764 ページの「EssMdxGetAxisMembers」
- 1772 ページの「EssMdxGetMbrIdentifier」
- 1772 ページの「EssMdxGetMbrProperty」

構文

```
  ESS_FUNC_M EssMdxGetClusterMembers(
ESS_MDX_CLUSTERHDL_T  hCluster,
ESS_ULONG_T       ulIndex,
ESS_MDX_PPMBRHDL_T   pphMbrs);
```

パラメータ	タイプ	説明
hCluster	入力	クラスタ・ハンドル
ulIndex	入力	クラスタの中のタプルの位置(最初の次元の変更が最も遅い順位でソートされます)
pphMbrs	出力	タプルについてのメンバーのハンドルの配列

EssMdxGetClusters

指定された軸の中のクラスタを戻します。

軸セットのコンテンツに関する情報を取得するには、次の API を使用します:

- 1769 ページの「EssMdxGetClusters」
- 1768 ページの「EssMdxGetClusterInfo」
- 1769 ページの「EssMdxGetClusterMembers」
- 1764 ページの「EssMdxGetAxisMembers」

- 1772 ページの「EssMdxGetMbrIdentifier」
- 1772 ページの「EssMdxGetMbrProperty」

構文

```
  ESS_FUNC_M EssMdxGetClusters(
ESS_MDX_AXISHDL_T     hAxis,
ESS_PULONG_T       pulNClusters,
ESS_MDX_PPCLUSTERHDL_T  pphClusters);
```

パラメータ	タイプ	説明
hAxis	入力	軸ハンドル
pulNClusters	出力	クラスタの数
pphClusters	出力	クラスタ・ハンドルの配列

EssMdxGetDimInfo

指定された次元に関する情報を戻します。この次元の中のメンバーについて使用可能なプロパティも含まれます。

構文

```
  ESS_FUNC_M EssMdxGetDimInfo(
ESS_MDX_DIMHDL_T    hDim,
ESS_PSTR_T       ppszName,
ESS_PULONG_T      pulNProps,
ESS_MDX_PPPROPHDL_T  pphProps);
```

パラメータ	タイプ	説明
hDim	入力	次元ハンドル
ppszDimName	出力	次元名
pulNProps	出力	戻されたプロパティの数
pphProps	出力	プロパティ・ハンドルの配列

注意

- このクエリーを呼び出す前に、1764 ページの「EssMdxGetAxisInfo」を呼び出して、軸に表示する次元を取得する必要があります。
- 次元のプロパティを取得するには、次の関数を呼び出して処理します:
  1. 1778 ページの「EssMdxNewQuery」を呼び出してクエリーを作成します。
  2. 1762 ページの「EssMdxExecuteQuery」を呼び出してクエリーを実行します。

3. 1763 ページの「EssMdxGetAxes」を呼び出して、クエリーの実行結果から軸の数と個別の軸のハンドルを取得します。
4. 1764 ページの「EssMdxGetAxisInfo」を呼び出して、1 つの軸のハンドルから個別の軸の情報(次元/タプル)を取得します。
5. 1770 ページの「EssMdxGetDimInfo」を呼び出して、次元の情報(次元名、この次元のプロパティの数およびプロパティ・ハンドル)を取得します。
6. 1774 ページの「EssMdxGetPropertyInfo」を呼び出して、次元プロパティを取得します。プロパティを取得するには、EssMdxQuery の MDX クエリーで DIMENSION PROPERTIES オプションを使用する必要があります。

EssMdxGetFormatString

指定したセルのフォーマットされた値を戻します。

注： セル・プロパティ・オプション ESS_MDX_CELLPROP_FORMAT_STRING が EssMdxSetQueryCellProperties を使用して設定されている場合にのみ、フォーマットされた値を戻します。

構文

```
  ESS_FUNC_M EssMdxGetFormatString(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_CELLHDL_T  hCell,
ESS_PSTR_T      pFmtStr);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
hCell	入力	セル・ハンドル
pFmtStr	出力	指定したセルのフォーマット文字列

関連項目

1771 ページの「EssMdxGetFormattedValue」

EssMdxGetFormattedValue

指定したセルのフォーマットされた値を戻します。

注： セル・プロパティ・オプション ESS_MDX_CELLPROP_FORMATTED_VALUE が EssMdxSetQueryCellProperties を使用して設定されている場合にのみ、フォーマットされた値を戻します。

構文

```
  ESS_FUNC_M EssMdxGetFormattedValue(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_CELLHDL_T  hCell,
ESS_PSTR_T     pFmtVal);
```

パラメータ	タイプ	説明
hQry	入力	入力クエリー・ハンドル
hCell	入力	入力セル・ハンドル
pFmtVal	出力	セルのフォーマットされた値

関連項目

1771 ページの「EssMdxGetFormatString」

EssMdxGetMbrIdentifier

指定されたメンバーについての識別子を戻します。

構文

```
  ESS_FUNC_M EssMdxGetMbrIdentifier(
ESS_MDX_MBRHDL_T  hMbr,
ESS_PSTR_T     ppszIdentifier);
```

パラメータ	タイプ	説明
hMbr	入力	メンバーのハンドル
ppszIdentifier	出力	メンバー識別子(名前または別名)

EssMdxGetMbrProperty

指定されたメンバーについて、指定されたプロパティの値を戻します。プロパティがメンバーに適用されない場合、プロパティ値のタイプはESS_MDX_PROPTYPE_NONE になります。

構文

```
  ESS_FUNC_M EssMdxGetMbrProperty(
ESS_MDX_MBRHDL_T    hMbr,
ESS_MDX_PROPHDL_T    hProp,
ESS_MDX_PPROPVALUE_T  pPropValue);
```

パラメータ	タイプ	説明
hMbr	入力	メンバーのハンドル
hProp	入力	プロパティ・ハンドル
pPropValue	出力	プロパティ値

EssMdxGetNamedSets

クエリーで名前付きセットを戻します。

構文

```
  ESS_FUNC_M EssMdxGetNamedSets(
ESS_HCTX_T   hCtx,
ESS_PULONG_T  pulCount,
ESS_PPSTR_T   ppNames,
ESS_PLONG_T   *ppTypes);
```

パラメータ	タイプ	説明
hCtx	入力	コンテキスト・ハンドル。
pulCount	出力	クエリーで戻された名前付きセットのカウント。
ppNames	出力	名前付きセットの配列。 ppNames に対して割り当てられたメモリーは、**EssFree()**を使用して解放する必要があります。
*ppTypes	出力	名前付きセット・タイプへのポインタ: ESS_MDX_NAMEDSET_TYPE_SESSION。

戻り値

戻り値は、pulCount の名前付きセット数、ppNames の名前付きセットおよび ppTypes の名前付きセットのタイプです。

Access

この関数を使用するのに、特別な権限は必要ありません。

例

```
   void TestGetNamedSets()
{
    ESS_STS_T sts = ESS_STS_NOERR;
    ESS_STR_T fileNames[2];
    ESS_CHAR_T qry[2][MAXQRYLEN];
    FILE *fileHandle;
    char *s;
    int length, e, i;
    ESS_ULONG_T ulCount, j;
    ESS_PSTR_T pNames;
```

```
    ESS_PLONG_T pTypes;

    fileNames[0] = "D:\\testarea\\MDXAPI\\query3.txt";
    fileNames[1] = "D:\\testarea\\MDXAPI\\query4.txt";

    for(i = 0; i < 2; i++)
    {
        fileHandle = fopen(fileNames[i], "r");
        if (!(fileHandle = fopen(fileNames[i], "r")))
        {
            printf("\nUnable to open file: %s\n", fileNames[i]);
            return;
        }
        else
        {
            s = qry[i];
            length = MAXQRYLEN;

            fgets(s, length, fileHandle);

            if ((e = ferror(fileHandle)) != 0)
            {
                printf("fgets error %d\n", e);
                exit((int) e);
            }
            fclose(fileHandle);
        }
        printf("\nThe query[%d]: \n%s\n", i, qry[i]);
    }

    ulCount = 0;
    sts = EssMdxGetNamedSets(hCtx, &ulCount, &pNames, &pTypes);
    printf("EssMdxGetNamedSets sts: %ld\n",sts);
    for(j = 0; j < ulCount; j++)
    {
        printf("\tpNames[%d]: %s\n", j, pNames[j]);
        printf("\tpTypes[%d]: %d\n", j, pTypes[j]);
        printf("\n");
    }

    sts = EssFree(hInst, (ESS_PVOID_T)pNames);
}
```

EssMdxGetPropertyInfo

指定されたプロパティに関する情報を戻します。

構文

```
  ESS_FUNC_M EssMdxGetPropertyInfo(
 ESS_MDX_PROPHDL_T   hProp,
 ESS_PSTR_T      ppszName,
 ESS_MDX_PPROPTYPE_T  pPropType);
```

パラメータ	タイプ	説明
hProp	入力	プロパティ・ハンドル
ppszName	出力	プロパティ名
pPropType	出力	プロパティ・タイプ: ● ESS_MDX_PROPTYPE_BOOL ● ESS_MDX_PROPTYPE_DOUBLE ● ESS_MDX_PROPTYPE_STRING ● ESS_MDX_PROPTYPE_DATETIME ● ESS_MDX_PROPTYPE_ULONG

EssMdxGetQueryCellProperties

このクエリーに有効なセル・プロパティを戻します。

構文

```
  ESS_FUNC_M    EssMdxGetQueryCellProperties(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_CELLPROPS_T  pulProp);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
pulProp	出力	戻されるセル・プロパティを指定するビットマスクへのポインタ

関連項目

1780 ページの「EssMdxSetQueryCellProperties」

EssMdxGetQueryOptions

現在のクエリーに有効なクエリー・プロパティを戻します。

構文

```
  ESS_FUNC_M    EssMdxGetQueryOptions(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_PQRYOPT_T  pulOpt);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
pulOpt	出力	有効な現在のクエリー・オプションを指定するビットマスクへのポインタ

関連項目

1780 ページの「EssMdxSetQueryOptions」

EssMdxGetSmartlistforCell

セル・タイプが ESS_MDX_VALTYPE_SMARTLIST の場合、セルと関連付けられたスマートリスト・オブジェクト名を戻します。Essbase データベースには、これらのオブジェクトと関連付けられた複数のスマートリスト・オブジェクトとスマートリスト・メンバーを保存できます。この関数は、セルが関連付けられているスマートリスト・オブジェクトを識別します。

注： セル・プロパティ・オプション ESS_MDX_CELLPROP_SMLIST_NAME が EssMdxSetQueryCellProperties を使用して設定されている場合にのみ、フォーマットされた値を戻します。

構文

```
  ESS_FUNC_M EssMdxGetSmartlistforCell(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_CELLHDL_T  hCell,
ESS_PSTR_T     pSmartlist);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
hCell	入力	セル・ハンドル
pSmartlist	出力	セルが関連付けられているスマートリスト・オブジェクトの名前

EssMdxGetValue

指定されたセルの値を戻します。

クエリーで、軸から形成されたキューブのセル値を取得するには、次の API を使用します:

- 1765 ページの「EssMdxGetCellAtOffset」
- 1765 ページの「EssMdxGetCellAtIndices」
- 1776 ページの「EssMdxGetValue」

構文

```
  ESS_FUNC_M EssMdxGetValue(
ESS_MDX_CELLHDL_T  hCell,
ESS_PBOOL_T     pbIsMissing,
ESS_PBOOL_T     pbNoAccess,
```

```
ESS_PDOUBLE_T    pdValue);
```

パラメータ	タイプ	説明
hCell	入力	セル・ハンドル
pbIsMissing	出力	セル値が#Missing かどうか
pbNoAccess	出力	セル値が#NoAccess かどうか
pdValue	出力	#Missing でなければセル値

EssMDXIsCellGLDrillable

セルがドリルスルー URL に関連付けられているかどうかをチェックします。

構文

```
ESS_FUNC_M EssMdxIsCellGLDrillable (hQry, hCell, pIsDrillable);
```

パラメータ	データ型	説明
hQry	ESS_MDX_QRYHDL_T	クエリー・ハンドル
hCell	ESS_MDX_CELLHDL_T	セル・ハンドル
pIsDrillable	ESS_PBOOL_T	セルがドリルスルー URL に関連付けられている場合は TRUE、それ以外の場合は FALSE

戻り値

- 正常に処理されると、セルのステータスに基づいて pIsDrillable が設定されます。
- 処理に失敗すると、エラー・メッセージが戻されます。

例

```
  #define ESS_MDX_CELLPROP_GLDRILLTHRU     0x00000008

if ((sts = EssMdxNewQuery(hCtx, qry, &hQry)) != ESS_STS_NOERR)
{
 printf("EssMdxNewQuery failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxNewQuery sts: %ld\n", sts);

if ((sts = EssMdxSetQueryCellProperties(hQry,
   (ESS_MDX_CELLPROP_GLDRILLTHRU
   )
 )) != ESS_STS_NOERR)
{
 printf("EssMdxSetQueryCellProperties failure: %ld\n", sts);
```

```
  exit ((int) sts);
 }
 if ((sts = EssMdxExecuteQuery(hQry)) != ESS_STS_NOERR)
 {
  printf("EssMdxExecuteQuery failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxExecuteQuery sts: %ld\n", sts);

/* To retrieve IsCellGLDrillable property of a cell, use EssMdxIsCellGLDrillable*/

 if ((sts = EssMdxIsCellGLDrillable(hQry, hCell, &bIsCellGLDT))
  != ESS_STS_NOERR)
 {
  printf("EssMdxIsCellGLDrillable failure: %ld\n", sts);
    exit ((int) sts);
 }
 if (bIsCellGLDT)
  printf(" Is Cell Drillable: TRUE\n");
 else
  printf(" Is Cell Drillable: FALSE\n");
```

EssMdxNewQuery

pszQry で指定された MDX クエリーを取り、クエリー・ハンドルを戻します。

構文

```
  ESS_FUNC_M EssMdxNewQuery(
ESS_HCTX_T     hCtx,
ESS_STR_T      pszQry,
ESS_MDX_PQRYHDL_T  phQry);
```

パラメータ	タイプ	説明
hCtx	入力	API コンテキスト・ハンドル
pszQry	入力	クエリー・テキスト
phQry	出力	クエリー・ハンドル

注意

この関数は MDX クエリーを作成する前に呼び出す必要があります。たとえば、
この関数を EssMDXExecuteQuery を呼び出す前に呼び出す必要があります。

EssMdxSetDataLess

セル・データを取得しないクエリー実行モードに切り替えます。**EssMdxGetCellAtOffset**()および **EssMdxGetCellAtIndices**()はクエリーに対して呼び出せません。デフォルトでは、セル・データを取得します。

構文

```
  ESS_FUNC_M EssMdxSetDataLess(
ESS_MDX_QRYHDL_T  hQry);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル

EssMDXSetHideData

#NOACCESS セルを#MISSING に変換します。

構文

```
  ESS_FUNC_M EssMDXSetHideData(
ESS_MDX_QRYHDL_T  hQry);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル

EssMdxSetMbrIdType

結果で必要なメンバー識別子のタイプを設定します。デフォルトは ESS_MDX_MEMBERIDTYPE_NAME です。

構文

```
  ESS_FUNC_M EssMdxSetMbrIdType(
ESS_MDX_QRYHDL_T     hQry,
ESS_MDX_MEMBERIDTYPE_T  mbrIdType);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
mbrIdType	入力	必要なメンバー識別子(名前または別名): ● ESS_MDX_MEMBERIDTYPE_NAME ● ESS_MDX_MEMBERIDTYPE_ALIAS

EssMdxSetNeedCellStatus

セル・ステータス情報の取得を有効にします。デフォルトでは、セル・ステータス情報は取得されません。

構文

```
  ESS_FUNC_M EssMdxSetNeedCellStatus(
ESS_MDX_QRYHDL_T  hQry);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル

EssMdxSetQueryCellProperties

各セルに対してサーバーから送信されるセル・プロパティを指定します。デフォルトで、セルの 1 つの値のみ送信されます。ulProp に渡されたオプションは、既存のクエリー・セル・プロパティを上書きします。つまり、EssMdxSetQueryCellProperties が複数回呼び出されると、最後の呼出しの ulProp 値のみ処理対象になります。

構文

```
  ESS_FUNC_M    EssMdxSetQueryCellProperties(
ESS_MDX_QRYHDL_T   hQry,
ESS_MDX_CELLPROPS_T  ulProp);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
ulProp	入力	送信するセル・プロパティを指定するビットマスク。値: ● ESS_MDX_CELLPROP_FORMATTED_VALUE ● ESS_MDX_CELLPROP_FORMAT_STRING ● ESS_MDX_CELLPROP_SMLIST_NAME

関連項目

1775 ページの「EssMdxGetQueryCellProperties」

EssMdxSetQueryOptions

ulOpt の値に基づいてクエリー・オプションを設定します。ulOpt に渡されたオプションは、既存のクエリー・オプションを上書きします。つまり、EssMdxSetQueryOptions が複数回呼び出されると、最後の呼出しの ulOpt 値のみが適用されます。

構文

```
  ESS_FUNC_M    EssMdxSetQueryOptions(
ESS_MDX_QRYHDL_T  hQry,
ESS_MDX_QRYOPT_T  ulOpt);
```

パラメータ	タイプ	説明
hQry	入力	クエリー・ハンドル
ulOpt	入力	クエリー・オプション。ビットマスク値: ● ESS_MDX_QRYOPT_GET_MI_CELLS このオプションは、#Missing セルに対してもフォーマットした値を生成する必要があることを示します。デフォルトでは、#Missing セルはサーバーでフォーマットされません。 ● ESS_MDX_QRYOPT_GET_ME_CELLS このオプションはサーバーに#Missing 値と#ME (無意味な値)とを区別させます。#ME は #Missing 値の特殊な場合です。基本メンバーと属性メンバーの組合せがそのセルのコンテキストで意味がないことを示します。デフォルトで、このオプションはオフに設定されています。

関連項目

1775 ページの「EssMdxGetQueryOptions」

MDX サンプル・クライアント・プログラム

```
   #if defined _WIN32 || defined _WINDOWS
#include <windows.h>
#endif

#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>
#if defined _WIN32 || defined _WINDOWS
#pragma pack(push,localid,1)
#endif
#include <essapi.h>
#if defined _WIN32 || defined _WINDOWS
#pragma pack(pop,localid)
#endif

ESS_HINST_T  hInst;
ESS_HCTX_T hCtx;
#define MAXQRYLEN 65536
ESS_CHAR_T qry[MAXQRYLEN];
ESS_STR_T AppName = "Sample";
ESS_STR_T DbName = "Basic";
```

```
static ESS_CHAR_T *axisnames[] =
{
  "COLUMNS", "ROWS", "PAGES", "CHAPTERS", "SECTIONS"
};

void ESS_Init()
{
  ESS_STS_T  sts;
  ESS_INIT_T InitStruct = {ESS_API_VERSION,
              NULL,
              0L,
              255,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              0L
             };
 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {
  printf("EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssInit sts: %ld\n", sts);
}

void ESS_Login ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;
 ESS_CHAR_T SvrName[ESS_SVRNAMELEN];
 ESS_CHAR_T UserName[ESS_USERNAMELEN];
 ESS_CHAR_T Password[ESS_PASSWORDLEN];

 /* Initialize parameters */
 strcpy(SvrName,"localhost");
 strcpy(UserName,"essexer");
 strcpy(Password,"password");
 sts = EssLogin(hInst, SvrName, UserName, Password, &Items,
       &pAppsDbs, &hCtx);
 if ( (sts != 0) && (sts != 1051093L) && (sts != 1051090L) )
 {
  printf("EssLogin failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssLogin sts: %ld\n", sts);
}

void ESS_MdxAxis(ESS_MDX_QRYHDL_T hQry,
       ESS_MDX_AXISHDL_T hAxis,
       ESS_STR_T pszAxisName
       )
```

```
{
 ESS_STS_T  sts;
 ESS_ULONG_T ulNAxisDims, ulAxisSize;
 ESS_ULONG_T ulNClusters, ulClusterSize, ulNClusterDims;
 ESS_ULONG_T ulAxisDimCnt, ulIndex, ulPropCnt;
 ESS_ULONG_T ulClusterCnt, ulClusterDimCnt;
 ESS_PULONG_T ulaDimSizes;
 ESS_MDX_PCLUSTERHDL_T haClusters;
 ESS_MDX_CLUSTERHDL_T hCluster;
 ESS_MDX_PDIMHDL_T haDims;
 ESS_STR_T pszDimName, pszMbrIdentifier, pszPropName;
 ESS_MDX_PMBRHDL_T haMbrs;
 ESS_PULONG_T ulaNProps = NULL;
 ESS_MDX_PPPROPHDL_T haaProps = NULL;
 ESS_MDX_PPROPHDL_T haProps;
 ESS_MDX_PROPHDL_T hProp;
 ESS_MDX_PROPTYPE_T propType;
 ESS_MDX_PROPVALUE_T propval;

 if ((sts = EssMdxGetAxisInfo(hAxis, &ulAxisSize, &ulNAxisDims,
             &haDims)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetAxisInfo failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetAxisInfo sts: %ld\n", sts);
 printf("%s Size %ld Num dims %ld\n", pszAxisName,
    ulAxisSize, ulNAxisDims);
 if (ulAxisSize == 0)
 {
  return;
 }
 if ((sts = EssAlloc(hInst,
          ulNAxisDims * sizeof(ESS_ULONG_T),
          (ESS_PPVOID_T) &ulaNProps)) != ESS_STS_NOERR)
 {
  printf("EssAlloc failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssAlloc sts: %ld\n", sts);
 if ((sts = EssAlloc(hInst,
          ulNAxisDims * sizeof(ESS_MDX_PPROPHDL_T),
          (ESS_PPVOID_T) &haaProps)) != ESS_STS_NOERR)
 {
  printf("EssAlloc failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssAlloc sts: %ld\n", sts);
 for (ulAxisDimCnt = 0; ulAxisDimCnt < ulNAxisDims;
   ulAxisDimCnt++)
 {
  if ((sts = EssMdxGetDimInfo(haDims[ulAxisDimCnt],
              &pszDimName,
              &ulaNProps[ulAxisDimCnt],
              &haaProps[ulAxisDimCnt])) != ESS_STS_NOERR)
  {
   printf("EssMdxGetDimInfo failure: %ld\n", sts);
```

```
   exit ((int) sts);
  }
  printf("EssMdxGetDimInfo sts: %ld\n", sts);
  printf("Dim %ld name %s #props %ld\n", ulAxisDimCnt,
     pszDimName, ulaNProps[ulAxisDimCnt]);
  haProps = haaProps[ulAxisDimCnt];
  for (ulPropCnt = 0; ulPropCnt < ulaNProps[ulAxisDimCnt]; ulPropCnt++)
  {
   hProp = haProps[ulPropCnt];
   if ((sts = EssMdxGetPropertyInfo(hProp, &pszPropName,
                 &propType)) != ESS_STS_NOERR)
   {
    printf("EssMdxGetPropertyInfo failure: %ld\n", sts);
    exit ((int) sts);
   }
   printf("EssMdxGetPropertyInfo sts: %ld\n", sts);
   printf("Property %ld type %ld name %s\n", ulPropCnt,
      propType, pszPropName);
  }
}
if ((sts = EssMdxGetClusters(hAxis, &ulNClusters,
             &haClusters)) != ESS_STS_NOERR)
{
 printf("EssMdxGetClusters failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxGetClusters sts: %ld\n", sts);
printf("Num clusters %ld\n", ulNClusters);
for (ulClusterCnt = 0; ulClusterCnt < ulNClusters;
  ulClusterCnt++)
{
 hCluster = haClusters[ulClusterCnt];
 if ((sts = EssMdxGetClusterInfo(hCluster, &ulClusterSize,
               &ulNClusterDims,
               &ulaDimSizes)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetClusterInfo failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetClusterInfo sts: %ld\n", sts);
 printf("Cluster %ld Size %ld\n", ulClusterCnt, ulClusterSize);
 for (ulClusterDimCnt = 0; ulClusterDimCnt < ulNClusterDims;
   ulClusterDimCnt++)
 {
  printf("Cluster Dim %ld Size %ld\n", ulClusterDimCnt,
     ulaDimSizes[ulClusterDimCnt]);
 }
 for (ulIndex = 0; ulIndex < ulClusterSize; ulIndex++)
 {
  if ((sts = EssMdxGetClusterMembers(hCluster, ulIndex,
                &haMbrs)) != ESS_STS_NOERR)
  {
   printf("EssMdxGetClusterMembers failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssMdxGetClusterMembers sts: %ld\n", sts);
  for (ulClusterDimCnt = 0; ulClusterDimCnt < ulNClusterDims;
```

```
  ulClusterDimCnt++)
{
 if ((sts = EssMdxGetMbrIdentifier(haMbrs[ulClusterDimCnt],
                &pszMbrIdentifier)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetMbrIdentifier failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetMbrIdentifier sts: %ld\n", sts);
 printf("Mbr %ld identifier %s\n", ulClusterDimCnt,
    pszMbrIdentifier);
 haProps = haaProps[ulClusterDimCnt];
 for (ulPropCnt = 0;
   ulPropCnt < ulaNProps[ulClusterDimCnt];
   ulPropCnt++)
 {
  if ((sts = EssMdxGetMbrProperty(haMbrs[ulClusterDimCnt],
                haProps[ulPropCnt],
                &propval)) != ESS_STS_NOERR)
  {
   printf("EssMdxGetMbrProperty failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssMdxGetMbrProperty sts: %ld\n", sts);
  printf("Property %ld Type ", ulPropCnt);
  switch (propval.ulPropType)
  {
   case ESS_MDX_PROPTYPE_ULONG:
   {
    printf("Ulong Value: %ld\n",
       propval.value.ulData);
    break;
   }
   case ESS_MDX_PROPTYPE_STRING:
   {
    printf("String Value: %s\n",
       propval.value.strData);
    break;
   }
   case ESS_MDX_PROPTYPE_BOOL:
   {
    printf("Bool Value: %s\n",
       propval.value.bData ? "TRUE" : "FALSE");
    break;
   }
   case ESS_MDX_PROPTYPE_DOUBLE:
   {
    printf("Double Value: %lf\n",
       propval.value.dblData);
    break;
   }
   case ESS_MDX_PROPTYPE_DATETIME:
   {
    ESS_CHAR_T tmpbuf[80];
    struct tm* pTime;
    pTime =  gmtime((time_t*)&(propval.value.dtData));
    sprintf(tmpbuf, "%02i-%02i-%04i",
```

```
            pTime->tm_mon+1, pTime->tm_mday,pTime->tm_year+1900);
          printf("DateTime Value: %s\n", tmpbuf);
          break;
         }
         case ESS_MDX_PROPTYPE_NONE:
         {
          printf("NULL Value\n");
          break;
         }
       }
     }
    }
    if ((sts = EssFree(hInst, (ESS_PVOID_T) haMbrs)) != ESS_STS_NOERR)
    {
     printf("EssFree failure: %ld\n", sts);
     exit ((int) sts);
    }
    printf("EssFree sts: %ld\n", sts);
  }
  for (ulClusterDimCnt = 0; ulClusterDimCnt < ulNClusterDims;
    ulClusterDimCnt++)
  {
   if ((sts = EssMdxGetClusterDimMembers(hCluster, ulClusterDimCnt,
                   &haMbrs)) != ESS_STS_NOERR)
   {
    printf("EssMdxGetClusterDimMembers failure: %ld\n", sts);
    exit ((int) sts);
   }
   printf("EssMdxGetClusterDimMembers sts: %ld\n", sts);
   for (ulIndex = 0; ulIndex < ulaDimSizes[ulClusterDimCnt];
     ulIndex++)
   {
    if ((sts = EssMdxGetMbrIdentifier(haMbrs[ulIndex],
                   &pszMbrIdentifier)) != ESS_STS_NOERR)
    {
     printf("EssMdxGetMbrIdentifier failure: %ld\n", sts);
     exit ((int) sts);
    }
    printf("EssMdxGetMbrIdentifier sts: %ld\n", sts);
    printf("Dim %ld Mbr %ld identifier %s\n", ulClusterDimCnt,
       ulIndex, pszMbrIdentifier);
   }
  }
 }
 for (ulIndex = 0; ulIndex < ulAxisSize; ulIndex++)
 {
  if ((sts = EssMdxGetAxisMembers(hAxis, ulIndex,
                &haMbrs)) != ESS_STS_NOERR)
  {
   printf("EssMdxGetAxisMembers failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssMdxGetAxisMembers sts: %ld\n", sts);
  for (ulAxisDimCnt = 0; ulAxisDimCnt < ulNAxisDims;
    ulAxisDimCnt++)
  {
   if ((sts = EssMdxGetMbrIdentifier(haMbrs[ulAxisDimCnt],
```

```
                &pszMbrIdentifier)) != ESS_STS_NOERR)
{
 printf("EssMdxGetMbrIdentifier failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxGetMbrIdentifier sts: %ld\n", sts);
printf("Mbr %ld identifier %s\n", ulAxisDimCnt, pszMbrIdentifier);
haProps = haaProps[ulAxisDimCnt];
for (ulPropCnt = 0;
  ulPropCnt < ulaNProps[ulAxisDimCnt];
  ulPropCnt++)
{
 hProp = haProps[ulPropCnt];
 if ((sts = EssMdxGetPropertyInfo(hProp, &pszPropName,
                &propType)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetPropertyInfo failure: %ld\n", sts);
  exit ((int) sts);
 }
 if ((sts = EssMdxGetMbrProperty(haMbrs[ulAxisDimCnt],
                hProp,
                &propval)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetMbrProperty failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetMbrProperty sts: %ld\n", sts);
 printf("Property %ld Type ", ulPropCnt);
 switch (propval.ulPropType)
 {
  case ESS_MDX_PROPTYPE_ULONG:
  {
   printf("Ulong Value: %ld\n",
      propval.value.ulData);
   break;
  }
  case ESS_MDX_PROPTYPE_STRING:
  {
   printf("String Value: %s\n",
      propval.value.strData);
   break;
  }
  case ESS_MDX_PROPTYPE_BOOL:
  {
   printf("Bool Value: %s\n",
      propval.value.bData ? "TRUE" : "FALSE");
   break;
  }
  case ESS_MDX_PROPTYPE_DOUBLE:
  {
   printf("Double Value: %lf\n",
      propval.value.dblData);
   break;
  }
  case ESS_MDX_PROPTYPE_DATETIME:
  {
   ESS_CHAR_T tmpbuf[80];
```

```
        struct tm* pTime;
        pTime =  gmtime((time_t*)&(propval.value.dtData));
        sprintf(tmpbuf, "%02i-%02i-%04i",
            pTime->tm_mon+1, pTime->tm_mday,pTime->tm_year+1900);
        printf("DateTime Value: %s\n", tmpbuf);
        break;
       }
       case ESS_MDX_PROPTYPE_NONE:
       {
        printf("NULL Value\n");
        break;
       }
      }
     }
   }
   if ((sts = EssFree(hInst, (ESS_PVOID_T) haMbrs)) != ESS_STS_NOERR)
   {
    printf("EssFree failure: %ld\n", sts);
    exit ((int) sts);
   }
   printf("EssFree sts: %ld\n", sts);
  }
  if ((sts = EssFree(hInst, (ESS_PVOID_T) ulaNProps)) != ESS_STS_NOERR)
  {
   printf("EssFree failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssFree sts: %ld\n", sts);
  if ((sts = EssFree(hInst, (ESS_PVOID_T) haaProps)) != ESS_STS_NOERR)
  {
   printf("EssFree failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssFree sts: %ld\n", sts);
}

void ESS_MdxQry()
{
  ESS_STS_T  sts;
  ESS_MDX_QRYHDL_T hQry;
  ESS_ULONG_T ulNAxes, ulNAxisDims, ulAxisSize, ulResultSize;
  ESS_ULONG_T ulNClusters, ulClusterSize, ulNClusterDims;
  ESS_ULONG_T ulAxisCnt, ulAxisDimCnt, ulIndex, ulPropCnt;
  ESS_ULONG_T ulCellOffset, ulClusterCnt, ulClusterDimCnt;
  ESS_MDX_CELLSTATUS_T ulCellStatus;
  ESS_PULONG_T ulaDimSizes;
  ESS_MDX_PCLUSTERHDL_T haClusters;
  ESS_MDX_CLUSTERHDL_T hCluster;
  ESS_MDX_PAXISHDL_T haAxes;
  ESS_MDX_PDIMHDL_T haDims;
  ESS_STR_T pszDimName, pszMbrIdentifier, pszPropName;
  ESS_MDX_AXISHDL_T hAxis, hSlicerAxis;
  ESS_MDX_PMBRHDL_T haMbrs;
  ESS_MDX_CELLHDL_T hCell;
  ESS_DOUBLE_T dValue;
  ESS_BOOL_T bIsMissing, bNoAccess;
  ESS_PULONG_T ulaNProps;
```

```
ESS_MDX_PPPROPHDL_T haaProps;
ESS_MDX_PPROPHDL_T haProps;
ESS_MDX_PROPHDL_T hProp;
ESS_MDX_PROPTYPE_T propType;
ESS_MDX_PROPVALUE_T propval;

if ((sts = EssMdxNewQuery(hCtx, qry, &hQry)) != ESS_STS_NOERR)
{
 printf("EssMdxNewQuery failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxNewQuery sts: %ld\n", sts);

if ((sts = EssMdxSetMbrIdType(hQry, ESS_MDX_MEMBERIDTYPE_ALIAS)) !=
      ESS_STS_NOERR)
{
 printf("EssMdxSetMbrIdType failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxSetMbrIdType sts: %ld\n", sts);

if ((sts = EssMdxSetNeedCellStatus(hQry)) != ESS_STS_NOERR)
{
 printf("EssMdxSetNeedCellStatus failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxSetNeedCellStatus sts: %ld\n", sts);

if ((sts = EssMdxExecuteQuery(hQry)) != ESS_STS_NOERR)
{
 printf("EssMdxExecuteQuery failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxExecuteQuery sts: %ld\n", sts);

if ((sts = EssMdxGetAxes(hQry, &ulNAxes, &haAxes,
           &hSlicerAxis)) != ESS_STS_NOERR)
{
 printf("EssMdxGetAxes failure: %ld\n", sts);
 exit ((int) sts);
}
printf("EssMdxGetAxes sts: %ld\n", sts);
printf("Number of axes: %ld\n", ulNAxes);

ulResultSize = 1;
for (ulAxisCnt = 0; ulAxisCnt < ulNAxes; ulAxisCnt++)
{
 hAxis = haAxes[ulAxisCnt];
 if ((sts = EssMdxGetAxisInfo(hAxis, &ulAxisSize, &ulNAxisDims,
             &haDims)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetAxisInfo failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetAxisInfo sts: %ld\n", sts);
 printf("Axis %ld Size %ld Num dims %ld\n", ulAxisCnt,
    ulAxisSize, ulNAxisDims);
```

```
 ulResultSize *= ulAxisSize;
}

if (hSlicerAxis)
{
 ESS_MdxAxis(hQry, hSlicerAxis, "SLICER");
}
else
{
 printf("Slicer Axis is empty\n");
}

for (ulAxisCnt = 0; ulAxisCnt < ulNAxes; ulAxisCnt++)
{
 hAxis = haAxes[ulAxisCnt];
 ESS_MdxAxis(hQry, hAxis, axisnames[ulAxisCnt]);
}
for (ulCellOffset = 0; ulCellOffset < ulResultSize;
  ulCellOffset++)
{
 if ((sts = EssMdxGetCellAtOffset(hQry, ulCellOffset,
               &hCell)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetCellAtOffset failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetCellAtOffset sts: %ld\n", sts);
 if ((sts = EssMdxGetValue(hCell, &bIsMissing, &bNoAccess,
            &dValue)) != ESS_STS_NOERR)
 {
  printf("EssMdxGetValue failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssMdxGetValue sts: %ld\n", sts);
 if (bIsMissing)
 {
  printf("CellOffset %ld Value #Missing\n", ulCellOffset);
 }
 else if (bNoAccess)
 {
  printf("CellOffset %ld Value #NoAccess\n", ulCellOffset);
 }
 else
 {
  printf("CellOffset %ld Value %lf\n", ulCellOffset,
     dValue);
 }
 if (!bNoAccess)
 {
  if ((sts = EssMdxGetCellStatus(hQry, hCell,
                &ulCellStatus)) != ESS_STS_NOERR)
  {
   printf("EssMdxGetCellStatus failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssMdxGetCellStatus sts: %ld\n", sts);
  if (ulCellStatus & ESS_MDX_CELLSTATUS_LINKEDOBJS)
```

```
    {
     printf("Cell status: LINKEDOBJS\n");
    }
    if (ulCellStatus & ESS_MDX_CELLSTATUS_DYNCALC)
    {
     printf("Cell status: DYNCALC\n");
    }
    if (ulCellStatus & ESS_MDX_CELLSTATUS_CALCEDMBR)
    {
     printf("Cell status: CALCEDMBR\n");
    }
    if (ulCellStatus & ESS_MDX_CELLSTATUS_READONLY)
    {
     printf("Cell status: READONLY\n");
    }
   }
  }

  if ((sts = EssMdxFreeQuery(hQry)) != ESS_STS_NOERR)
  {
   printf("EssMdxFreeQuery failure: %ld\n", sts);
   exit ((int) sts);
  }
  printf("EssMdxFreeQuery sts: %ld\n", sts);

}

void ESS_Term()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
  {
   /* error terminating API */
   exit((ESS_USHORT_T) sts);
  }
  printf("EssTerm sts: %ld\n", sts);
}

void ESS_Logout()
{
  ESS_STS_T sts = ESS_STS_NOERR;

  sts = EssLogout(hCtx);
  printf("\n\nEssLogout sts: %ld\n",sts);
}

void ESS_SetActive()
{
  ESS_STS_T sts = ESS_STS_NOERR;
  ESS_ACCESS_T Access;

  sts = EssSetActive(hCtx, AppName, DbName, &Access);
  printf("EssSetActive sts: %ld\n",sts);
}

int main(int argc, char *argv[])
{
```

```
 FILE *f;
 char *s, *sout;
 int n, l, e;

 assert(argc > 1);
 f = fopen(argv[1], "r");
 assert(f != NULL);
 s = qry;
 n = MAXQRYLEN;
 while (n > 0 && !feof(f) && fgets(s, n, f) != NULL)
 {
  l = strlen(s);
  s += l;
  n -= l;
 }
 if ((e = ferror(f)) != 0)
 {
  printf("fgets error %d\n", e);
  exit((int) e);
 }
 fclose(f);
 printf("The query is\n%s\n", qry);
 if (argc > 2)
 {
  AppName = argv[2];
 }
 if (argc > 3)
 {
  DbName = argv[3];
 }

 ESS_Init();
 ESS_Login();
 ESS_SetActive();

 ESS_MdxQry();

 ESS_Logout();
 ESS_Term();

 return 0;
}
```

27 XMLAリファレンスへようこそ

XML for Analysis (XMLA) API を使用するには、Provider Services をインストールする必要があります。『Oracle Hyperion Enterprise Performance Management System インストール概要』および『Oracle Hyperion Enterprise Performance Management System インストールおよび構成ガイド』を参照してください。このヘルプでは、XMLA メソッドについて説明し、rowset のサンプル・コードを提供します。XMLA クライアントは Provider Services を介してのみ Essbase と通信できます。

詳細は、左側のフレームの「目次」、「インデックス」または「検索」をクリックしてください。

28 XMLAの操作

この章の内容

主な機能

XML for Analysis (XMLA)はアウトライン分析処理向けに設計されたオープンな業界標準 Web サービスインタフェースです。XMLA は HTTP、XML、Simple Object Access Protocol (SOAP)のオープン・スタンダードで作成された、XML メッセージ・インタフェースのセットです。XMLA はどの言語、プラットフォーム、オペレーティング・システムにも依存せず、クライアント・アプリケーションと Web 上の多次元データ・ソース間で標準化されたデータ・アクセスを提供します。

主な機能:

- フラット化された行セットのサポート
- ステートフル・セッションのサポート
- 下位 XMLA レベル表現(レベル 1 がトップ・レベル)
- 基本 HTTP 認証によるユーザー認証
- Oracle Hyperion Provider Services による XMLA 高可用性機能
- Oracle Essbase Administration Services による XMLA 管理および監視

注： XMLA は Essbase とのみ併用可能です。

メソッド

次のメソッドは、XML アプリケーションがサーバーの基本情報にアクセスする際の標準的な方法です。この種のメソッドは SOAP を使用して呼び出されるため、XML での入力を受け入れ、出力も XML で戻します。デフォルトではメソッドはステートレスなため、サーバー・コンテキストはあらゆるコマンドの完了時に終了します。

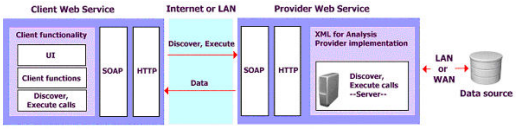
簡略化されたインタフェース・モデルには 2 つのメソッドがあります。

- Discover

- Execute

Discover では、Web サービスからの情報とメタデータを入手します。この種の情報には、使用可能なデータ・ソースと、データ・ソースのプロバイダのリストが含まれます。プロパティによって、入手するデータが定義されます。Discover によって、クライアント・アプリケーションが必要としている情報のタイプを指定できます。汎用インタフェースとプロパティの使用により、既存の関数を書き換えずに拡張性を実現できます。

Execute では Multidimensional Expressions (MDX)やその他のプロバイダ固有のコマンドを XMLA データ・ソースに実行できます。次の図は、n 層のアプリケーションの実装例を示したものです。



Web サービスのホスティングを行っているサーバーの URL で提供されている SOAP および HTTP プロトコルを使用して、クライアントはサーバーに Discover および Execute 呼出しを送信します。サーバーは XMLA プロバイダのインスタンス化を行い、そこで呼出しが処理されます。XMLA プロバイダはデータをフェッチして XML にパッケージ化し、データをクライアントに送信します。

Discover および Execute メソッドによって、ユーザーはサーバーでのクエリー内容の判断が可能になり、それに基づいて実行するコマンドを送信します。

この 2 つのメソッドの XML 名前は"urn:schemas-microsoft-com:xml-analysis"の形式で指定します。接続情報は、各呼出しで接続プロパティによって提供されます。

Discover

Discover メソッドは、サーバー上のデータ・ソースのリストまたはデータ・ソースに関する詳細などの情報を取得します。Discover メソッドで取得されるデータは、メソッドに渡されたパラメータの値により異なります。

ネームスペース

```
urn:schemas-microsoft-com:xml-analysis
```

SOAP アクション

```
"urn:schemas-microsoft-com:xml-analysis:Discover"
```

構文

```
  Discover (
[in] RequestType As EnumString,
[in] Restrictions As Restrictions,
[in] Properties As Properties,
[out] Result As Rowset)
```

パラメータ

RequestType [in]

この必須パラメータは、RequestType 列挙値で構成されます。これは、戻される情報のタイプを特定します。RequestType 列挙は、Discover メソッドによって使用され、Result パラメータで戻された行セットの構造およびコンテンツを特定します。また、Restrictions パラメータ・フォーマットおよび XML 結果セットも、このパラメータで指定された値に依存します。この列挙は、プロバイダ特有の列挙文字列をサポートするために拡張できます。

各 RequestType 列挙値は、戻り行セットに対応します。行セットの定義は、1801 ページの「XMLA 行セット」の説明を参照してください。次の明示的に名前を付けられた RequestType 列挙値には、サポートが必要です。

列挙値	説明
DISCOVER_DATASOURCES	サーバーまたは Web サービスで利用可能な XMLA データ・ソースのリストを戻します。
DISCOVER_PROPERTIES	指定されたデータ・ソース(プロバイダ)がサポートする、要求されたプロパティに関する情報および値のリストを戻します。
DISCOVER_SCHEMA_ROWSETS	(ここに記載されたものを含む)サポートされているすべての RequestType 列挙値の名前、値、他の情報および追加のプロバイダ特有の列挙値を戻します。
DISCOVER_ENUMERATORS	特定のデータ・ソースのプロバイダがサポートする列挙子の名前、データ型および列挙値のリストを戻します。
DISCOVER_KEYWORDS	プロバイダが予約しているキーワードのリストを含む行セットを戻します。
DISCOVER_LITERALS	データ・ソース・プロバイダがサポートするリテラルに関する情報を戻します。スキーマ行セット定数が指定されている場合、MDSCHEMA_CUBES など OLE DB によって定義されたスキーマ行セット名の 1 つに対応する定数は、XML フォーマットで OLE DB スキーマ行セットを戻します。追加のプロバイダ特有のスキーマ行セットの提供により、プロバイダも OLEDB を拡張できることに注意してください。表データ・プロバイダ(TDP)および多次元データ・プロバイダ(MDP)がサポートする必要があるスキーマ行セットは、セクション「DISCOVER_SCHEMA_ROWSETS 行セット」にリストされています。

Restrictions [in]

Restrictions データ型のこのパラメータによって、Result で戻されるデータを制限できます。Result 列は、RequestType パラメータで指定された行セットによって定義されます。Result のいくつかの列は、戻された行をフィルタできます。これらの列および制限できる列については、1801 ページの「XMLA 行セット」の説明にある行セット表を参照してください。プロバイダ特有のスキーマ行セットの制限情報を取得するには、DISCOVER_SCHEMA_ROWSETS 要求タイプを使用します。このパラメータは空でもかまいませんが、含まれている必要があります。

Properties [in]

Properties データ型のこのパラメータは、XMLA プロパティの集合で構成されます。各プロパティによって、結果セットの戻りフォーマット、タイムアウトまたはデータがフォーマットされるロケールを指定するなど、Discover メソッドのある側面を制御できます。

Discover メソッドと DISCOVER_PROPERTIES 要求タイプを使用することにより、利用可能なプロパティを取得できます。

Properties パラメータ内のプロパティには、必須の順序がありません。このパラメータは空でもかまいませんが、含まれている必要があります。

Result [out]

この必須パラメータは、Rowset オブジェクトとしてプロバイダが戻す結果セットを含んでいます。結果セットの列およびコンテンツは、RequestType および Restrictions パラメータの値によって指定されます。戻された結果セットの列レイアウトは、RequestType で指定された値によっても特定されます。各 RequestType 値に対応する行セットのレイアウトについては、1801 ページの「XMLA 行セット」を参照してください。

例

次のサンプルでは、クライアントは、Demo カタログからキューブのリストを要求する XML Discover コールを送信します:

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_CUBES</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>Demo</CATALOG_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

プロバイダはクライアントに次の結果を戻します:

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
  <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sql="urn:schemas-microsoft-com:xml-sql"
    elementFormDefault="qualified">
    <xsd:element name="root">
     <xsd:complexType>
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
       <xsd:element name="row" type="row"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="row">
     <xsd:sequence maxOccurs="unbounded" minOccurs="0">
      <xsd:element name="CATALOG_NAME" type="xsd:string"
       sql:field="CATALOG_NAME"/>
      <xsd:element name="CUBE_NAME" type="xsd:string"
       sql:field="CUBE_NAME"/>
      <xsd:element name="CUBE_TYPE" type="xsd:string"
       sql:field="CUBE_TYPE"/>
      <xsd:element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime"
       sql:field="LAST_SCHEMA_UPDATE" minOccurs="0"/>
      <xsd:element name="DESCRIPTION" type="xsd:string"
       sql:field="DESCRIPTION" minOccurs="0"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:schema>
   <row>
    <CATALOG_NAME>Demo</CATALOG_NAME>
    <CUBE_NAME>Demo.Basic</CUBE_NAME>
    <CUBE_TYPE>CUBE</CUBE_TYPE>
   </row>
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Execute

Execute メソッドは、サーバー上のデータ取得や更新など、サーバーへのデータ転送を含むアクションの要求を送信します。

ネームスペース

```
urn:schemas-microsoft-com:xml-analysis
```

SOAP アクション

```
"urn:schemas-microsoft-com:xml-analysis:Execute"
```

構文

```
  Execute (
 [in] Command As Command,
 [in] Properties As Properties,
 [out] Result As Resultset)
```

パラメータ

Command [in]

この必須パラメータは Command データ型で、実行する MDX ステートメントで構成されています。

Properties [in]

このパラメータは Parameter データ型で、XMLA プロパティの集合から構成されます。各プロパティによって、ユーザーは接続に必要な情報の定義、結果セットの戻りフォーマットの指定、データのフォーマット用のロケール指定など、Execute メソッドの特定の側面を制御できます。

使用可能なプロパティとその値は、Discover メソッドで DISCOVER_PROPERTIES 要求タイプを使用して取得できます。

Properties パラメータ内のプロパティには、必須の順序がありません。このパラメータは空でもかまいませんが、含まれている必要があります。

Result [out]

このパラメータにはプロバイダから戻された Resultset の結果が含まれています。Command パラメータと Properties パラメータの値によって、結果セットの形状が定義されます。形状を定義するプロパティが渡されない場合は、XMLA プロバイダはデフォルトの形状を使用できます。この種の指定によって定義される 2 つの結果セット・フォーマットとして、Tabular (タブ区切り)と Multidimensional (多次元)があり、Format プロパティによってクライアントが指定します。OLAP データは多次元フォーマットで提供されます(ただしタブ区切りフォーマットも使用できます)。プロバイダは追加の行セット・タイプをサポートしていることがあるため、特殊なタイプが必要なクライアントは要求できます。

例

Execute メソッドの呼出しで、<Statement>を MDX SELECT ステートメントに設定した例:

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<SOAP-ENV:Body>
 <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <Command>
  <Statement>
  SELECT  CrossJoin([Measures].CHILDREN , [Market].CHILDREN)
  on columns,  [Product].Members on rows
  from Sample.Basic
```

```
 </Statement>
 </Command>
 <Properties>
  <PropertyList>
   <DataSourceInfo>
    Provider=Essbase;Data Source=localhost
   </DataSourceInfo>
   <Catalog>Sample</Catalog>
   <Format>Multidimensional</Format>
   <AxisFormat>TupleFormat</AxisFormat>
   <Content>SchemaData</Content>
  </PropertyList></Properties>
 </Execute>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

前のメソッドの呼出しに対する簡略化された応答:

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:ExecuteResponse
  xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:return
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:mddataset">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xars="urn:schemas-microsoft-com:xars">
     ...<!-The schema for the data goes here. -- >
    </xsd:schema>
    ... <!-The data in MDDataSet format goes here. -- >
   </root>
  </m:return>
  </m:ExecuteResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

XMLA 行セット

Discover メソッドの Result パラメータで戻される情報は、このセクションで説明する行セットの列レイアウトに従って構造化されます。

CATALOGS rowset

CATALOGS rowset は、Analytic Services からアクセスできるカタログと関連付けられた物理属性を特定します。

GUID: DBSCHEMA_CATALOGS

フラット化した rowset の例項で、この rowset 構造体について説明します。

表 32　CATALOGS rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
DESCRIPTION	常に NULL

要求の例

```
   <SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>DBSCHEMA_CATALOGS</RequestType>
  <Restrictions>
   <RestrictionList></RestrictionList>
  </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
```

```
   <xsd:complexType>
    <xsd:sequence minOccurs="0" maxOccurs="unbounded">
     <xsd:element name="row" type="row"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="row">
   <xsd:sequence maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="CATALOG_NAME" type="xsd:string"
     sql:field="CATALOG_NAME"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
     sql:field="DESCRIPTION" minOccurs="0"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
 <row>
  <CATALOG_NAME>Demo</CATALOG_NAME>
 </row>
 < ..............More Rows..... >
 </root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_CUBES Rowset

CUBES rowset にはスキーマ内(またはプロバイダがスキーマをサポートしない場合はカタログ)で使用できるキューブについての情報が含まれています。

GUID: MDSCHEMA_CUBES

表 33 で、この rowset 構造体について説明します。

表 33　MDSCHEMA_CUBES rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
CUBE_TYPE	"CUBE"
LAST_SCHEMA_UPDATE	最後のアウトライン更新のタイム・スタンプ
DESCRIPTION	データベースの説明

要求の例

```
  <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
```

```
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_CUBES</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>Demo</CATALOG_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="row">
       <xsd:sequence maxOccurs="unbounded" minOccurs="0">
        <xsd:element name="CATALOG_NAME" type="xsd:string"
         sql:field="CATALOG_NAME"/>
        <xsd:element name="CUBE_NAME" type="xsd:string"
         sql:field="CUBE_NAME"/>
        <xsd:element name="CUBE_TYPE" type="xsd:string"
         sql:field="CUBE_TYPE"/>
        <xsd:element name="LAST_SCHEMA_UPDATE" type="xsd:dateTime"
```

```
    sql:field="LAST_SCHEMA_UPDATE" minOccurs="0"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
    sql:field="DESCRIPTION" minOccurs="0"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
 <row>
  <CATALOG_NAME>Demo</CATALOG_NAME>
  <CUBE_NAME>Demo.Basic</CUBE_NAME>
  <CUBE_TYPE>CUBE</CUBE_TYPE>
 </row>
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_DIMENSIONS rowset

DIMENSIONS rowset には指定されたキューブの次元に関する情報が含まれます。各次元には 1 つの行があります。

GUID: MDSCHEMA_DIMENSIONS

表 34 で、この rowset 構造体について説明します。

表 34　MDSCHEMA_DIMENSIONS rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
DIMENSION_NAME	次元名
DIMENSION_UNIQUE_NAME	次元名
DIMENSION_CAPTION	次元名
DIMENSION_ORDINAL	次元番号。最初の次元が 1、次が 2 と続きます
DIMENSION_TYPE	Essbase 次元タイプ: ● 時間: MD_DIMTYPE_TIME ● 勘定科目: MD_DIMTYPE_MEASURE ● その他: MD_DIMTYPE_OTHER
DIMENSION_CARDINALITY	次元内のメンバー数
DEFAULT_HIERARCHY	次元名
DESCRIPTION	次元に追加されたコメント
DIMENSION_UNIQUE_SETTINGS	2

列名	Essbase マッピング
DIMENSION_IS_VISIBLE	常に TRUE

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_DIMENSIONS</RequestType>
  <Restrictions>
   <RestrictionList>
    <CATALOG_NAME>Sample</CATALOG_NAME>
    <CUBE_NAME>Basic</CUBE_NAME>
   </RestrictionList>
  </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
```

```
   </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="row">
   <xsd:sequence maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="CATALOG_NAME" type="xsd:string"
    sql:field="CATALOG_NAME"/>
    <xsd:element name="CUBE_NAME" type="xsd:string"
    sql:field="CUBE_NAME"/>
    <xsd:element name="DIMENSION_NAME" type="xsd:string"
    sql:field="DIMENSION_NAME"/>
    <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
    sql:field="DIMENSION_UNIQUE_NAME"/>
    <xsd:element name="DIMENSION_CAPTION" type="xsd:string"
    sql:field="DIMENSION_CAPTION"/>
    <xsd:element name="DIMENSION_ORDINAL" type="xsd:unsignedInt"
    sql:field="DIMENSION_ORDINAL"/>
    <xsd:element name="DIMENSION_TYPE" type="xsd:short"
    sql:field="DIMENSION_TYPE"/>
    <xsd:element name="DIMENSION_CARDINALITY" type="xsd:unsignedInt"
    sql:field="DIMENSION_CARDINALITY"/>
    <xsd:element name="DEFAULT_HIERARCHY" type="xsd:string"
    sql:field="DEFAULT_HIERARCHY"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
    sql:field="DESCRIPTION" minOccurs="0"/>
    <xsd:element name="DIMENSION_UNIQUE_SETTINGS" type="xsd:int"
    sql:field="DIMENSION_UNIQUE_SETTINGS"/>
    <xsd:element name="DIMENSION_IS_VISIBLE" type="xsd:boolean"
    sql:field="DIMENSION_IS_VISIBLE"/>
   </xsd:sequence>
  </xsd:complexType>
  </xsd:schema>
  <row>
   <CATALOG_NAME>Sample</CATALOG_NAME>
   <CUBE_NAME>Sample.Basic</CUBE_NAME>
   <DIMENSION_NAME>Year</DIMENSION_NAME>
   <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
   <DIMENSION_CAPTION>Year</DIMENSION_CAPTION>
   <DIMENSION_ORDINAL>1</DIMENSION_ORDINAL>
   <DIMENSION_TYPE>1</DIMENSION_TYPE>
   <DIMENSION_CARDINALITY>19</DIMENSION_CARDINALITY>
   <DEFAULT_HIERARCHY>[Year]</DEFAULT_HIERARCHY>
   <DIMENSION_UNIQUE_SETTINGS>2</DIMENSION_UNIQUE_SETTINGS>
   <DIMENSION_IS_VISIBLE>true</DIMENSION_IS_VISIBLE>
  </row>
  < ..............More Rows..... >
 </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_FUNCTIONS 行セット

FUNCTIONS 行セットは MDP がサポートするすべての関数を提供します。デフォルトのソート順: ORIGIN、INTERFACE_NAME、FUNCTION_NAME。

GUID: MDSCHEMA_FUNCTIONS

表 35 で、この rowset 構造体について説明します。

表 35　MDSCHEMA_FUNCTIONS rowset 構造体

列名	Essbase マッピング
FUNCTION_NAME	関数の名前
DESCRIPTION	関数の説明
PARAM_LIST	カンマ区切りのパラメータ・リスト
RETURN_TYPE	常時 12
ORIGIN	1(常時: MDX 関数)
INTERFACE_NAME	メンバー、セット、タプル、数値、次元、レベル、ブールのうちのいずれか
OBJECT	セット、メンバー、タプル、レベル、階層、次元のうちのいずれか
HELP_CONTEXT	関数のヘルプ・コンテキスト ID
CAPTION	関数の表示キャプション

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_FUNCTIONS</RequestType>
  <Restrictions><RestrictionList></RestrictionList></Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
   <?xml version="1.0"?>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="FUNCTION_NAME" type="xsd:string"
        sql:field="FUNCTION_NAME"/>
       <xsd:element name="DESCRIPTION" type="xsd:string"
        sql:field="DESCRIPTION"/>
       <xsd:element name="PARAMETER_LIST" type="xsd:string"
        sql:field="PARAMETER_LIST"/>
       <xsd:element name="RETURN_TYPE" type="xsd:int"
        sql:field="RETURN_TYPE"/>
       <xsd:element name="ORIGIN" type="xsd:int"
        sql:field="ORIGIN"/>
       <xsd:element name="INTERFACE_NAME" type="xsd:string"
        sql:field="INTERFACE_NAME"/>
       <xsd:element name="OBJECT" type="xsd:string"
        sql:field="OBJECT" minOccurs="0"/>
       <xsd:element name="HELP_CONTEXT" type="xsd:int"
        sql:field="HELP_CONTEXT" minOccurs="0"/>
       <xsd:element name="CAPTION" type="xsd:string"
        sql:field="CAPTION"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
    <!-- Begin: All MDX functions that return a Member
     (INTERFACE_NAME=Member) -->
    <row>
     <FUNCTION_NAME>Ancestor</FUNCTION_NAME>
     <DESCRIPTION>Given the input member, returns the ancestor
      at the specified level.</DESCRIPTION>
     <PARAMETER_LIST>Member, Level | Numeric Expression</PARAMETER_LIST>
     <RETURN_TYPE>12</RETURN_TYPE>
     <ORIGIN>1</ORIGIN>
     <INTERFACE_NAME>Member</INTERFACE_NAME>
     <HELP_CONTEXT>9142</HELP_CONTEXT>
```

```
   <CAPTION>Ancestor</CAPTION>
   </row>
   < ..............More Rows..... >
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_HIERARCHIES 行セット

HIERARCHIES 行セットは次元で使用できる階層に関する情報を含んでいます。

GUID: MDSCHEMA_HIERARCHIES

表 36 で、この rowset 構造体について説明します。

表 36　MDSCHEMA_HIERARCHIES rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
DIMENSION_UNIQUE_NAME	次元名
HIERARCHY_NAME	次元名
HIERARCHY_UNIQUE_NAME	次元名
HIERARCHY_CAPTION	次元名
DIMENSION_TYPE	Essbase 次元タイプ: ● 時間: MD_DIMTYPE_TIME ● ACCOUNTS: MD_DIMTYPE_MEASSURE ● その他: MD_DIMTYPE_OTHER
HIERARCHY_CARDINALITY	次元内のメンバー数
DEFAULT_MEMBER	次元名
ALL_MEMBER	次元名
DESCRIPTION	次元のコメント
STRUCTURE	MD_STRUCTURE_UNBALANCED(2)
HIERARCHY_UNIQUE_SETTINGS	2
HIERARCHY_IS_VISIBLE	TRUE

要求の例

```
   <SOAP-ENV:Envelope
```

```
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>MDSCHEMA_HIERARCHIES</RequestType>
   <Restrictions>
    <RestrictionList>
     <CUBE_NAME>Sample.Basic</CUBE_NAME>
     <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
 <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:return xsi:type="xsd:string"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="CATALOG_NAME" type="xsd:string"
        sql:field="CATALOG_NAME"/>
```

```
    <xsd:element name="CUBE_NAME" type="xsd:string"
     sql:field="CUBE_NAME"/>
    <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
     sql:field="DIMENSION_UNIQUE_NAME"/>
    <xsd:element name="HIERARCHY_NAME" type="xsd:string"
     sql:field="HIERARCHY_NAME"/>
    <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
     sql:field="HIERARCHY_UNIQUE_NAME"/>
    <xsd:element name="HIERARCHY_CAPTION" type="xsd:string"
     sql:field="HIERARCHY_CAPTION"/>
    <xsd:element name="DIMENSION_TYPE" type="xsd:short"
     sql:field="DIMENSION_TYPE"/>
    <xsd:element name="HIERARCHY_CARDINALITY" type="xsd:unsignedInt"
     sql:field="HIERARCHY_CARDINALITY"/>
    <xsd:element name="DEFAULT_MEMBER" type="xsd:string"
     sql:field="DEFAULT_MEMBER"/>
    <xsd:element name="ALL_MEMBER" type="xsd:string"
     sql:field="ALL_MEMBER"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
     sql:field="DESCRIPTION" minOccurs="0"/>
    <xsd:element name="STRUCTURE" type="xsd:int"
     sql:field="STRUCTURE"/>
    <xsd:element name="HIERARCHY_UNIQUE_SETTINGS" type="xsd:int"
     sql:field="HIERARCHY_UNIQUE_SETTINGS"/>
    <xsd:element name="HIERARCHY_IS_VISIBLE" type="xsd:boolean"
     sql:field="HIERARCHY_IS_VISIBLE"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
 <row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
  <HIERARCHY_NAME>Year</HIERARCHY_NAME>
  <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
  <HIERARCHY_CAPTION>Year</HIERARCHY_CAPTION>
  <DIMENSION_TYPE>1</DIMENSION_TYPE>
  <HIERARCHY_CARDINALITY>19</HIERARCHY_CARDINALITY>
  <DEFAULT_MEMBER>[Year]</DEFAULT_MEMBER>
  <ALL_MEMBER>[Year]</ALL_MEMBER>
  <STRUCTURE>2</STRUCTURE>
  <HIERARCHY_UNIQUE_SETTINGS>2</HIERARCHY_UNIQUE_SETTINGS>
  <HIERARCHY_IS_VISIBLE>true</HIERARCHY_IS_VISIBLE>
 </row>
</root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_MEASURES Rowset

MEASURES rowset は使用できるメジャーに関する情報を含んでいます。

GUID: MDSCHEMA_MEASURES

表 37 で、この rowset 構造体について説明します。

表 37　MDSCHEMA_MEASURES rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
MEASURE_NAME	会計次元メンバー名
MEASURE_UNIQUE_NAME	前述のメンバー名
MEASURE_CAPTION	前述のメンバー名
MEASURE_AGGREGATOR	Essbase ADDITION: 1 Essbase SUBSTRACTION: 17 Essbase MULTIPLICATION: 18 Essbase DIVISION: 19 Essbase PERCENT: 20 Essbase NOOP: 21
DESCRIPTION	メンバーのコメント
DATA_TYPE	5
EXPRESSION	メンバー式
MEASURE_IS_VISIBLE	TRUE

要求の例

```
  <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <RequestType>MDSCHEMA_MEASURES</RequestType>
  <Restrictions>
  <RestrictionList>
   <CATALOG_NAME>Sample</CATALOG_NAME>
   <CUBE_NAME>Basic</CUBE_NAME>
  </RestrictionList>
 </Restrictions>
 <Properties>
  <PropertyList>
   <DataSourceInfo>Provider=Essbase;Data Source=localhost
   </DataSourceInfo>
   <Format>Tabular</Format>
  </PropertyList>
 </Properties>
</Discover>
```

```
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="CATALOG_NAME" type="xsd:string"
        sql:field="CATALOG_NAME"/>
       <xsd:element name="CUBE_NAME" type="xsd:string"
        sql:field="CUBE_NAME"/>
       <xsd:element name="MEASURE_NAME" type="xsd:string"
        sql:field="MEASURE_NAME"/>
       <xsd:element name="MEASURE_UNIQUE_NAME" type="xsd:string"
        sql:field="MEASURE_UNIQUE_NAME"/>
       <xsd:element name="MEASURE_CAPTION" type="xsd:string"
        sql:field="MEASURE_CAPTION"/>
       <xsd:element name="MEASURE_AGGREGATOR" type="xsd:int"
        sql:field="MEASURE_AGGREGATOR"/>
       <xsd:element name="DESCRIPTION" type="xsd:string"
        sql:field="DESCRIPTION" minOccurs="0"/>
       <xsd:element name="DATA_TYPE" type="xsd:unsignedShort"
        sql:field="DATA_TYPE"/>
       <xsd:element name="NUMERIC_PRECISION" type="xsd:unsignedShort"
        sql:field="NUMERIC_PRECISION"/>
       <xsd:element name="NUMERIC_SCALE" type="xsd:short"
        sql:field="NUMERIC_SCALE"/>
       <xsd:element name="EXPRESSION" type="xsd:string"
        sql:field="EXPRESSION" minOccurs="0"/>
       <xsd:element name="MEASURE_IS_VISIBLE" type="xsd:boolean"
        sql:field="MEASURE_IS_VISIBLE"/>
```

```
   </xsd:sequence>
  </xsd:complexType>
 </xsd:schema>
 <row>
  <CATALOG_NAME>Sample</CATALOG_NAME>
  <CUBE_NAME>Sample.Basic</CUBE_NAME>
  <MEASURE_NAME>Measures</MEASURE_NAME>
  <MEASURE_UNIQUE_NAME>[Measures]</MEASURE_UNIQUE_NAME>
  <MEASURE_CAPTION>Measures</MEASURE_CAPTION>
  <MEASURE_AGGREGATOR>0</MEASURE_AGGREGATOR>
  <DATA_TYPE>5</DATA_TYPE>
  <NUMERIC_PRECISION>0</NUMERIC_PRECISION>
  <NUMERIC_SCALE>0</NUMERIC_SCALE>
  <MEASURE_IS_VISIBLE>true</MEASURE_IS_VISIBLE>
 </row>
 < ..............More Rows..... >
 </root>
 </m:return>
 </m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_MEMBERS rowset

MEMBERS rowset は使用できるメンバーに関する情報を含んでいます。

GUID: MDSCHEMA_MEMBERS

表 38 で、この rowset 構造体について説明します。

表 38　MDSCHEMA_MEMBERS rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
DIMENSION_UNIQUE_NAME	次元名
HIERARCHY_UNIQUE_NAME	次元名
LEVEL_UNIQUE_NAME	レベル名
LEVEL_NUMBER	レベル番号
GENERATION_NUMBER	世代番号
MEMBER_ORDINAL	メンバー番号
MEMBER_NAME	メンバー名
MEMBER_UNIQUE_NAME	一意のメンバー名
MEMBER_TYPE	1(通常)

列名	Essbase マッピング
MEMBER_CAPTION	メンバー名
MEMBER_ALIAS	デフォルトの別名
CHILDREN_CARDINALITY	子の数
PARENT_LEVEL	親のレベル番号。次元については、次元レベル番号と同じレベル番号
PARENT_UNIQUE_NAME	親の名前。次元については、次元名と同じ名前
PARENT_COUNT	常に 1
DESCRIPTION	メンバーのコメント

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
secext-1.0.xsd">
<SOAP-ENV:Header>
 <wsse:Security>
  <wsse:UsernameToken>
   <wsse:Username>system</wsse:Username>
   <wsse:Password>password</wsse:Password>
  </wsse:UsernameToken>
 </wsse:Security>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <RequestType>MDSCHEMA_MEMBERS</RequestType>
 <Restrictions>
  <RestrictionList>
   <CATALOG_NAME>Sample</CATALOG_NAME>
   <CUBE_NAME>Basic</CUBE_NAME>
   <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
  </RestrictionList>
 </Restrictions>
 <Properties>
  <PropertyList>
   <DataSourceInfo>
    Provider=Essbase;Data Source=localhost
   </DataSourceInfo>
   <Format>Tabular</Format>
  </PropertyList>
 </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
  <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
       targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:sql="urn:schemas-microsoft-com:xml-sql"
       elementFormDefault="qualified">
       <xsd:element name="root">
        <xsd:complexType>
         <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="row" type="row"/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
       <xsd:complexType name="row">
        <xsd:sequence maxOccurs="unbounded" minOccurs="0">
         <xsd:element name="CATALOG_NAME" type="xsd:string"
          sql:field="CATALOG_NAME"/>
         <xsd:element name="CUBE_NAME" type="xsd:string"
          sql:field="CUBE_NAME"/>
         <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
          sql:field="DIMENSION_UNIQUE_NAME"/>
         <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
          sql:field="HIERARCHY_UNIQUE_NAME"/>
         <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
          sql:field="LEVEL_UNIQUE_NAME"/>
         <xsd:element name="LEVEL_NUMBER" type="xsd:unsignedInt"
          sql:field="LEVEL_NUMBER"/>
         <xsd:element name="GENERATION_NUMBER" type="xsd:unsignedInt"
          sql:field="GENERATION_NUMBER"/>
         <xsd:element name="MEMBER_ORDINAL" type="xsd:unsignedInt"
          sql:field="MEMBER_ORDINAL"/>
         <xsd:element name="MEMBER_NAME" type="xsd:string"
          sql:field="MEMBER_NAME"/>
         <xsd:element name="MEMBER_UNIQUE_NAME" type="xsd:string"
          sql:field="MEMBER_UNIQUE_NAME"/>
         <xsd:element name="MEMBER_TYPE" type="xsd:int"
          sql:field="MEMBER_TYPE"/>
         <xsd:element name="MEMBER_CAPTION" type="xsd:string"
          sql:field="MEMBER_CAPTION"/>
         <xsd:element name="MEMBER_ALIAS" type="xsd:string"
          sql:field="MEMBER_ALIAS" minOccurs="0"/>
         <xsd:element name="CHILDREN_CARDINALITY" type="xsd:unsignedInt"
          sql:field="CHILDREN_CARDINALITY"/>
         <xsd:element name="PARENT_LEVEL" type="xsd:unsignedInt"
          sql:field="PARENT_LEVEL"/>
```

```
      <xsd:element name="PARENT_UNIQUE_NAME" type="xsd:string"
       sql:field="PARENT_UNIQUE_NAME"/>
      <xsd:element name="PARENT_COUNT" type="xsd:unsignedInt"
       sql:field="PARENT_COUNT"/>
      <xsd:element name="DESCRIPTION" type="xsd:string"
       sql:field="DESCRIPTION" minOccurs="0" />
     </xsd:sequence>
    </xsd:complexType>
   </xsd:schema>
   <row>
    <CATALOG_NAME>Sample</CATALOG_NAME>
    <CUBE_NAME>Sample.Basic</CUBE_NAME>
    <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
    <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
    <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
    <LEVEL_NUMBER>2</LEVEL_NUMBER>
    <GENERATION_NUMBER>1</GENERATION_NUMBER>
    <MEMBER_ORDINAL>1</MEMBER_ORDINAL>
    <MEMBER_NAME>Jan</MEMBER_NAME>
    <MEMBER_UNIQUE_NAME>[Jan]</MEMBER_UNIQUE_NAME>
    <MEMBER_TYPE>1</MEMBER_TYPE>
    <MEMBER_CAPTION>Jan</MEMBER_CAPTION>
    <CHILDREN_CARDINALITY>0</CHILDREN_CARDINALITY>
    <PARENT_LEVEL>1</PARENT_LEVEL>
    <PARENT_UNIQUE_NAME>[Qtr1]</PARENT_UNIQUE_NAME>
    <PARENT_COUNT>1</PARENT_COUNT>
   </row>
   < ..............More Rows..... >
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_PROPERTIES 行セット

PROPERTIES 行セットには、次元の各レベルで使用可能なプロパティの情報が含まれています。ここでは各レベルにメンバー・クラスが定義されていることが前提となっています。このクラス内のすべてのメンバーのプロパティは同一です。名前の付いたレベルをサポートしていないデータ・ストアの場合は、ダミー・レベルに次元内のすべてのメンバーが含まれます。このレベルの名前は、次元の名前と同じです。

デフォルトのソート順序: PROPERTY_TYPE、CATALOG_NAME、SCHEMA_NAME、CUBE_NAME、DIMENSION_UNIQUE_NAME、HIERARCHY_UNIQUE_NAME および LEVEL_UNIQUE_NAME。

GUID: MDSCHEMA_PROPERTIES

表 39 で、この rowset 構造体について説明します。

表 39　MDSCHEMA_PROPERTIES rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
HIERARCHY_UNIQUE_NAME	次元名
LEVEL_UNIQUE_NAME	次元名
PROPERTY_TYPE	1 (MDPROP_MEMBER)
PROPERTY_NAME	次のいずれかです: ● 属性次元の場合は、次元名はプロパティ名と同じです ● UDA の場合は UDA 名 ● 別名の場合は別名
PROPERTY_CAPTION	次のいずれかです: ● 属性次元の場合は属性次元名 ● UDA の場合は UDA 名 ● 別名の場合は別名
DATA_TYPE	1 (倍精度) - 属性次元 2 (ブール値) - 属性次元 3 (文字列) - 属性次元、UDA または別名 4(整数) - 属性次元
CHARACTER_MAXIMUM_LENGTH	80 (UDA または属性次元の場合) 30 (別名の場合)
CHARACTER_OCTET_LENGTH	320 (UDA または属性次元の場合) 120 (別名の場合)
PROPERTY_CONTENT_TYPE	0 (MD_PROPTYPE_REGULAR)
SQL_COLUMN_NAME	次のいずれかです: ● 属性次元の場合は属性次元名 ● UDA の場合は UDA 名 ● 別名の場合は別名
PROPERTY_ORIGIN	1 (MD_USER_DEFINED)
PROPERTY_ATTRIBUTE_HIERARCHY_NAME	属性次元の場合は属性次元名
PROPERTY_CARDINALITY	ONE (UDA または別名の場合) MANY (属性次元の場合)
PROPERTY_IS_VISIBLE	TRUE

要求の例

```
   <SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <SOAP-ENV:Body>
  <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <RequestType>MDSCHEMA_PROPERTIES</RequestType>
   <Restrictions>
    <RestrictionList>
     <CATALOG_NAME>Sample</CATALOG_NAME>
     <CUBE_NAME>Basic</CUBE_NAME>
     <DIMENSION_UNIQUE_NAME>Product</DIMENSION_UNIQUE_NAME>
     <LEVEL_UNIQUE_NAME>SKU</LEVEL_UNIQUE_NAME>
    </RestrictionList>
   </Restrictions>
   <Properties>
    <PropertyList>
     <DataSourceInfo>
      Provider=Essbase;Data Source=localhost
     </DataSourceInfo>
     <Format>Tabular</Format>
    </PropertyList>
   </Properties>
  </Discover>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例(抜粋)

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
   <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
    <m:return xsi:type="xsd:string"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
      targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:sql="urn:schemas-microsoft-com:xml-sql"
      elementFormDefault="qualified">
      <xsd:element name="root">
       <xsd:complexType>
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="row" type="row"/>
        </xsd:sequence>
       </xsd:complexType>
      </xsd:element>
```

```
<xsd:complexType name="row">
 <xsd:sequence maxOccurs="unbounded" minOccurs="0">
  <xsd:element name="CATALOG_NAME" type="xsd:string"
   sql:field="CATALOG_NAME"/>
  <xsd:element name="CUBE_NAME" type="xsd:string"
   sql:field="CUBE_NAME"/>
  <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
   sql:field="DIMENSION_UNIQUE_NAME"/>
  <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
   sql:field="HIERARCHY_UNIQUE_NAME"/>
  <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
   sql:field="LEVEL_UNIQUE_NAME" minOccurs ="0"/>
  <xsd:element name="MEMBER_UNIQUE_NAME" type="xsd:string"
   sql:field="MEMBER_UNIQUE_NAME" minOccurs ="0"/>
  <xsd:element name="PROPERTY_TYPE" type="xsd:short"
   sql:field="PROPERTY_TYPE" minOccurs ="0"/>
  <xsd:element name="PROPERTY_NAME" type="xsd:string"
   sql:field="PROPERTY_NAME" minOccurs ="0"/>
  <xsd:element name="PROPERTY_CAPTION" type="xsd:string"
   sql:field="PROPERTY_CAPTION" minOccurs ="0"/>
  <xsd:element name="DATA_TYPE" type="xsd:unsignedShort"
   sql:field="DATA_TYPE" minOccurs ="0"/>
  <xsd:element name="CHARACTER_MAXIMUM_LENGTH"
   type="xsd:unsignedInt"
   sql:field="CHARACTER_MAXIMUM_LENGTH" minOccurs ="0"/>
  <xsd:element name="CHARACTER_OCTET_LENGTH" type="xsd:unsignedInt"
   sql:field="CHARACTER_OCTET_LENGTH" minOccurs ="0"/>
  <xsd:element name="NUMERIC_PRECISION" type="xsd:unsignedShort"
   sql:field="NUMERIC_PRECISION" minOccurs ="0"/>
  <xsd:element name="NUMERIC_SCALE" type="xsd:short"
   sql:field="NUMERIC_SCALE" minOccurs ="0"/>
  <xsd:element name="DESCRIPTION" type="xsd:string"
   sql:field="DESCRIPTION" minOccurs ="0"/>
  <xsd:element name="PROPERTY_CONTENT_TYPE" type="xsd:short"
   sql:field="PROPERTY_CONTENT_TYPE" minOccurs ="0"/>
  <xsd:element name="SQL_COLUMN_NAME" type="xsd:string"
   sql:field="SQL_COLUMN_NAME" minOccurs ="0"/>
  <xsd:element name="LANGUAGE" type="xsd:unsignedShort"
   sql:field="LANGUAGE" minOccurs ="0"/>
  <xsd:element name="PROPERTY_ORIGIN" type="xsd:unsignedShort"
   sql:field="PROPERTY_ORIGIN" minOccurs ="0"/>
  <xsd:element name="PROPERTY_ATTRIBUTE_HIERARCHY_NAME"
   type="xsd:string"
   sql:field="PROPERTY_ATTRIBUTE_HIERARCHY_NAME" minOccurs ="0"/>
  <xsd:element name="PROPERTY_CARDINALITY" type="xsd:string"
   sql:field="PROPERTY_CARDINALITY" minOccurs ="0"/>
  <xsd:element name="MIME_TYPE" type="xsd:string"
   sql:field="MIME_TYPE" minOccurs ="0"/>
  <xsd:element name="PROPERTY_IS_VISIBLE" type="xsd:boolean"
   sql:field="PROPERTY_IS_VISIBLE" minOccurs ="0"/>
 </xsd:sequence>
</xsd:complexType>
</xsd:schema>
<row>
 <CATALOG_NAME>Sample</CATALOG_NAME>
 <CUBE_NAME>Sample.Basic</CUBE_NAME>
 <DIMENSION_UNIQUE_NAME>[Product]</DIMENSION_UNIQUE_NAME>
```

```
    <HIERARCHY_UNIQUE_NAME>[Product]</HIERARCHY_UNIQUE_NAME>
    <LEVEL_UNIQUE_NAME>[Product]</LEVEL_UNIQUE_NAME>
    <PROPERTY_TYPE>1</PROPERTY_TYPE>
    <PROPERTY_NAME>Caffeinated</PROPERTY_NAME>
    <PROPERTY_CAPTION>Caffeinated</PROPERTY_CAPTION>
    <DATA_TYPE>2</DATA_TYPE>
    <PROPERTY_CONTENT_TYPE>0</PROPERTY_CONTENT_TYPE>
    <SQL_COLUMN_NAME>Caffeinated</SQL_COLUMN_NAME>
    <PROPERTY_ORIGIN>1</PROPERTY_ORIGIN>
    <PROPERTY_ATTRIBUTE_HIERARCHY_NAME>Caffeinated
    </PROPERTY_ATTRIBUTE_HIERARCHY_NAME>
    <PROPERTY_CARDINALITY>MANY</PROPERTY_CARDINALITY>
    <PROPERTY_IS_VISIBLE>true</PROPERTY_IS_VISIBLE>
   </row>
   < ................More Rows..............  >
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

MDSCHEMA_SETS Rowset

SETS rowset には、スキーマ(またはプロバイダがスキーマをサポートしない場合は、カタログ)内のセットに関する情報が含まれています。

GUID: MDSCHEMA_SETS

表 40 で、この rowset 構造体について説明します。

表 40　MDSCHEMA_SETS rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
SET_NAME	セットの名前
SCOPE	セッション

MDSCHEMA_LEVELS 行セット

LEVELS 行セットは次元で使用できるレベルに関する情報を含んでいます。

GUID: MDSCHEMA_LEVELS

表 41 で、この rowset 構造体について説明します。

表 41　MDSCHEMA_LEVELS rowset 構造体

列名	Essbase マッピング
CATALOG_NAME	アプリケーション名
CUBE_NAME	データベース名
DIMENSION_UNIQUE_NAME	レベルが属する次元の名前
HIERARCHY_UNIQUE_NAME	レベルが属する次元の名前
LEVEL_NAME	一意のレベル名
LEVEL_UNIQUE_NAME	一意のレベル名
LEVEL_CAPTION	レベル名
LEVEL_NUMBER	レベル番号
LEVEL_CARDINALITY	レベル内のメンバー数
LEVEL_TYPE	MDLEVEL_TYPE_ALL (次元レベル) MDLEVEL_TYPE_TIME (次元タイプ TIME) MDLEVEL_TYPE_REGULAR (その他すべて)
LEVEL_UNIQUE_SETTINGS	2 (MDDIMENSIONS_MEMBER_NAME_UNIQUE)
LEVEL_IS_VISIBLE	TRUE
ESSBASE_GEN_UNIQUE_NAME	世代の一意の名前
ESSBASE_GEN_CAPTION	世代のキャプション

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <RequestType>MDSCHEMA_LEVELS</RequestType>
 <Restrictions>
  <RestrictionList>
   <CATALOG_NAME>Sample</CATALOG_NAME>
    <CUBE_NAME>Basic</CUBE_NAME>
    <DIMENSION_UNIQUE_NAME>Year</DIMENSION_UNIQUE_NAME>
   </RestrictionList>
 </Restrictions>
 <Properties>
  <PropertyList>
   <DataSourceInfo>Provider=Essbase;Data Source=localhost
   </DataSourceInfo>
   <Format>Tabular</Format>
  </PropertyList>
```

```
 </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return xsi:type="xsd:string"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="CATALOG_NAME" type="xsd:string"
        sql:field="CATALOG_NAME"/>
       <xsd:element name="CUBE_NAME" type="xsd:string"
        sql:field="CUBE_NAME"/>
       <xsd:element name="DIMENSION_UNIQUE_NAME" type="xsd:string"
        sql:field="DIMENSION_UNIQUE_NAME"/>
       <xsd:element name="HIERARCHY_UNIQUE_NAME" type="xsd:string"
        sql:field="HIERARCHY_UNIQUE_NAME"/>
       <xsd:element name="LEVEL_NAME" type="xsd:string"
        sql:field="LEVEL_NAME"/>
       <xsd:element name="LEVEL_UNIQUE_NAME" type="xsd:string"
        sql:field="LEVEL_UNIQUE_NAME"/>
       <xsd:element name="LEVEL_CAPTION" type="xsd:string"
        sql:field="LEVEL_CAPTION"/>
       <xsd:element name="LEVEL_NUMBER" type="xsd:unsignedInt"
        sql:field="LEVEL_NUMBER"/>
       <xsd:element name="LEVEL_CARDINALITY" type="xsd:unsignedInt"
        sql:field="LEVEL_CARDINALITY"/>
       <xsd:element name="LEVEL_TYPE" type="xsd:int"
        sql:field="LEVEL_TYPE"/>
       <xsd:element name="LEVEL_UNIQUE_SETTINGS" type="xsd:int"
        sql:field="LEVEL_UNIQUE_SETTINGS"/>
```

```
    <xsd:element name="LEVEL_IS_VISIBLE" type="xsd:boolean"
     sql:field="LEVEL_IS_VISIBLE"/>
    <xsd:element name="DESCRIPTION" type="xsd:string"
     sql:field="DESCRIPTION" minOccurs="0"/>
              <xsd:element name="ESSBASE_GEN_UNIQUE_NAME" type="xsd:string"
                  sql:field="ESSBASE_GEN_UNIQUE_NAME"/>
              <xsd:element name="ESSBASE_GEN_CAPTION" type="xsd:string"
                  sql:field="ESSBASE_GEN_CAPTION"/>
    </xsd:sequence>
   </xsd:complexType>
  </xsd:schema>
  <row>
   <CATALOG_NAME>Sample</CATALOG_NAME>
   <CUBE_NAME>Sample.Basic</CUBE_NAME>
   <DIMENSION_UNIQUE_NAME>[Year]</DIMENSION_UNIQUE_NAME>
   <HIERARCHY_UNIQUE_NAME>[Year]</HIERARCHY_UNIQUE_NAME>
   <LEVEL_NAME>[Year].Levels(2)</LEVEL_NAME>
   <LEVEL_UNIQUE_NAME>[Year].Levels(2)</LEVEL_UNIQUE_NAME>
   <LEVEL_CAPTION>[Year].Level 2</LEVEL_CAPTION>
   <LEVEL_NUMBER>2</LEVEL_NUMBER>
   <LEVEL_CARDINALITY>12</LEVEL_CARDINALITY>
   <LEVEL_TYPE>4</LEVEL_TYPE>
   <LEVEL_UNIQUE_SETTINGS>2</LEVEL_UNIQUE_SETTINGS>
   <LEVEL_IS_VISIBLE>true</LEVEL_IS_VISIBLE>
              <ESSBASE_GEN_UNIQUE_NAME>[Year].[Months]</ESSBASE_GEN_UNIQUE_NAME>
              <ESSBASE_GEN_CAPTION>[Year].Months</ESSBASE_GEN_CAPTION>
  </row>
  < ..............More Rows............. >
 </root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DISCOVER_SCHEMA_ROWSETS Rowset

GUID: DISCOVER_SCHEMA_ROWSETS

表 42 で、この rowset 構造体について説明します。

表 42　DISCOVER_SCHEMA rowset 構造体

列名	Essbase マッピング
SchemaName	スキーマ/要求の名前。RequestTypes 列挙値、およびプロバイダがサポートするその他の型で値を戻します。プロバイダは、その他の型の rowset 構造体を定義します。
Restrictions	許可された制限のリスト
Description	スキーマの説明

DISCOVER_DATASOURCES 行セット

GUID: DISCOVER_DATASOURCES

表 43 で、この rowset 構造体について説明します。

表 43　DISCOVER_DATASOURCES rowset 構造体

列名	Essbase マッピング
DataSourceName	データ・ソースの名前
DataSourceDescription	データ・ソースの説明
DataSourceInfo	プロバイダ=Essbase データ・ソース=Analytic Server の名前
ProviderName	Essbase の XMLA
ProviderType	MDP
AuthenticationMode	認証済

DISCOVER_PROPERTIES Rowset

GUID: DISCOVER_PROPERTIES

表 44 で、この rowset 構造体について説明します。

表 44　DISCOVER_PROPERTIES rowset 構造体

列名	Essbase マッピング
PropertyName	プロパティの名前
PropertyDescription	プロパティの説明
PropertyType	プロパティの XML データ型。
PropertyAccessType	プロパティのアクセス。値は、Read、Write、ReadWrite です
IsRequired	プロパティが必要な場合は TRUE、不要の場合は FALSE
Value	プロパティの現在の値

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>DISCOVER_PROPERTIES</RequestType>
   <Restrictions>
    <RestrictionList></RestrictionList>
   </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>Provider=Essbase;Data Source=localhost
```

```
  </DataSourceInfo>
  <Format>Tabular</Format>
 </PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
  <?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
<m:return xsi:type="xsd:string"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:sql="urn:schemas-microsoft-com:xml-sql"
elementFormDefault="qualified">
<xsd:element name="root">
<xsd:complexType>
<xsd:sequence minOccurs="0" maxOccurs="unbounded">
<xsd:element name="row" type="row"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="row">
<xsd:sequence maxOccurs="unbounded" minOccurs="0">
<xsd:element name="PropertyName" type="xsd:string"
sql:field="PropertyName"/>
<xsd:element name="PropertyDescription" type="xsd:string"
sql:field="PropertyDescription"/>
<xsd:element name="PropertyType" type="xsd:string"
sql:field="PropertyType"/>
<xsd:element name="PropertyAccessType" type="xsd:string"
sql:field="PropertyAccessType"/>
<xsd:element name="IsRequired" type="xsd:boolean"
sql:field="IsRequired"/>
<xsd:element name="Value" type="xsd:string"
sql:field="Value"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
<row>
<PropertyName>ProviderName</PropertyName>
<PropertyDescription>The name of the Analytic Services Provider
```

```
   </PropertyDescription>
   <PropertyType>string</PropertyType>
   <PropertyAccessType>Read</PropertyAccessType>
   <IsRequired>false</IsRequired>
   <Value>Analytic Services XML for Analysis Provider</Value>
  </row>
  < ..............More Rows..... >
 </root>
</m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DISCOVER_ENUMERATORS Rowset

GUID: DISCOVER_ENUMERATORS

表 45 で、この rowset 構造体について説明します。

表 45　DISCOVER_ENUMERATORS rowset 構造体

列名	Essbase マッピング
EnumName	値のセットを含む列挙子の名前
EnumDescription	列挙子の説明
ElementName	列挙子セットの値要素の名前 例: TDP
ElementDescription	要素の説明
EnumType	Enum 値のデータ型
ElementValue	要素の値 例: 01

要求の例

```
  <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <RequestType>DISCOVER_ENUMERATORS</RequestType>
 <Restrictions>
  <RestrictionList></RestrictionList>
 </Restrictions>
 <Properties>
  <PropertyList>
   <DataSourceInfo>
    Provider=Essbase;Data Source=localhost
```

```
  </DataSourceInfo>
  <Format>Tabular</Format>
 </PropertyList>
</Properties>
</Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
 <m:DiscoverResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
  <m:return xsi:type="xsd:string"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType  name="row">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
       <xsd:element name="EnumName" type="xsd:string"
        sql:field="EnumName"/>
       <xsd:element name="EnumDescription" type="xsd:string"
        sql:field="EnumDescription" minOccurs="0"/>
       <xsd:element name="ElementName" type="xsd:string"
        sql:field="ElementName"/>
       <xsd:element name="ElementDescription" type="xsd:string"
        sql:field="ElementDescription" minOccurs="0"/>
       <xsd:element name="ElementValue" type="xsd:string"
        sql:field="ElementValue" minOccurs="0"/>
       <xsd:element name="EnumType" type="xsd:string"
        sql:field="EnumType"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
    <row>
     <EnumName>ProviderType</EnumName>
     <ElementName>TDP</ElementName>
```

```
   <EnumType>string</EnumType>
   </row>
   < ..............More Rows..... >
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DISCOVER_KEYWORDS rowset

GUID: DISCOVER_KEYWORDS

表 46 で、この rowset 構造体について説明します。

表 46　DISCOVER_KEYWORDS rowset 構造体

列名	Essbase マッピング
Keyword	プロバイダによって予約済のキーワードのリスト 例: AND

要求の例

```
   <SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Discover xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <RequestType>DISCOVER_KEYWORDS</RequestType>
  <Restrictions>
   <RestrictionList></RestrictionList>
  </Restrictions>
  <Properties>
   <PropertyList>
    <DataSourceInfo>
     Provider=Essbase;Data Source=localhost
    </DataSourceInfo>
    <Format>Tabular</Format>
   </PropertyList>
  </Properties>
 </Discover>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

応答の例

```
   <?xml version="1.0"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

```
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
 <m:DiscoverResponse
 xmlns:m="urn:schemas-microsoft-com:xml-analysis">
 <m:return xsi:type="xsd:string"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
    targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:sql="urn:schemas-microsoft-com:xml-sql"
    elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row"/>
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
    <xsd:complexType name="row">
     <xsd:sequence maxOccurs="unbounded" minOccurs="0">
      <xsd:element name="Keyword" type="xsd:string"
       sql:field="Keyword"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:schema>
   <row><Keyword>aggregate</Keyword></row>
   <row><Keyword>ancestors</Keyword></row>
   < ..............More Rows..... >
  </root>
 </m:return>
</m:DiscoverResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DISCOVER_LITERALS 行セット

GUID: DISCOVER_LITERALS

例 1 項で、この rowset 構造体について説明します。

表 47　DISCOVER_LITERALS rowset 構造体

列名	Essbase マッピング
LiteralName	行で説明されるリテルの名前 例: DBLITERAL_LIKE_PERCENT

列名	Essbase マッピング
LiteralValue	リテラル値を含みます たとえば、LiteralName が DBLITERAL_LIKE_PERCENT で、パーセント文字(%)で LIKE 句内のゼロ個以上の文字の一致に使用される場合、この列の値は"%"になります。
LiteralInvalidChars	リテラル内の有効でない文字 例: テーブル名に数字以外の文字を含める場合、この文字列は"0123456789"となります
LiteralInvalidStartingChars	リテラルの先頭文字として有効でない文字。リテラルの先頭文字がどの有効な文字でもよい場合は、これは NULL になります。
LiteralMaxLength	リテラル内の最大文字数。最大値がない場合、または最大値が不明な場合は、値は-1になります。

フラット化した rowset の例

rowset のフラット化は、多次元データをグリッドで表現する方法です。このデータの 2 次元テーブル表示で、多次元 XMLA 要求の出力の理解が容易になります。

MDX の例

次の例は、MDX クエリーおよび結果のフラット化した rowset を示しています。MDX は表現を簡単にするために使用されています。ただし、この例のクエリーは XMLA SOAP 要求の面で解釈するよう作成されています。XMLA ではレベル 0 は次元を示し、MDX のようにリーフ・メンバーを示すものでないことに注意してください。このため、これらの例は MDX ですが、レベルは XMLA と同様に反転されています。

例 1

次のクエリーでは、レベル 1 のすべてのメンバーを要求します。

```
   SELECT NON EMPTY {[Profit]} ON COLUMNS,
NON EMPTY [Product].Levels(1).ALLMEMBERS ON ROWS
FROM Sample.Basic
```

このクエリーには、次の結果があります:

[Product].[Family].[MEMBER_CAPTION]	[Profit]
100	30468
200	27954
300	25799
400	21301

[Product].[Family].[MEMBER_CAPTION]	[Profit]
Diet	28826

例 2

次のクエリーは、最大 2 つのレベルを要求します。rowset のフラット化には、このレベル(2)へのこの要求にレベル 1 が含まれます。フラット化した rowset を使用する場合、レベル N にクエリーを行うと N からレベル 1 が戻されます。

```
   SELECT NON EMPTY {[Profit] } ON COLUMNS,
NON EMPTY [Product].Levels(2).ALLMEMBERS ON ROWS
FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Product].[Family].[MEMBER_CAPTION]	[Product].[SKU].[MEMBER_CAPTION]	[Profit]
100	**100-10**	22777
100	**100-20**	5708
100	**100-30**	1983
200	**200-10**	7201
200	**200-20**	12025
200	**200-30**	4636
200	**200-40**	4092
...	**...**	...

例 3

次のクエリーは以前のクエリーを基に作成され、レベルが 1 から N (ここで N=2) のセットのメンバー固有の名前およびレベル番号プロパティを含める結果セットを要求します。各メンバーと各プロパティには行が割り当てられます。

```
   SELECT NON EMPTY {[Profit]} ON COLUMNS,
NON EMPTY [Product].Levels(2).ALLMEMBERS
 DIMENSION PROPERTIES MEMBER_UNIQUE_NAME, LEVEL_NUMBER
ON ROWS
FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Product]。 [Family]。 [MEMBER_ UNIQUE_ NAME]	[Product]. [Family]. LEVEL_ NUMBER	[Product]。 [SKU]。 [MEMBER_ UNIQUE_ NAME]	[Product]. [SKU]. LEVEL_ NUMBER	[Profit]
[100]	1	[100-10]	2	22777
[100]	1	[100-20]	2	5708
[100]	1	[100-30]	2	1983
[200]	1	[200-10]	2	7201
[200]	1	[200-20]	2	12025
[200]	1	[200-30]	2	4636
[200]	1	[200-40]	2	4092
[300]	1	[300-10]	2	12195
[300]	1	[300-20]	2	2511
[300]	1	[300-30]	2	2511
...	...	...	...	...

例 4

フラット化された行セットクエリーに CrossJoin を実装することで、複数次元(少なくとも 2 つ)を使用できます。この例では、Market および Product 次元が要求されます。各次元に対して、以前の例と同じ論理が適用されます。各次元、レベル、プロパティには 1 列が割り当てられます(この場合、1 つのレベルと 1 つのプロパティが要求されます)。

```
   SELECT NON EMPTY {[Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(1).AllMembers,[Product].Levels(1).ALLMEMBERS)
 DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS
FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Market].Levels(1). [MEMBER_CAPTION]	[Product]. [Family]. [MEMBER_CAPTION]	[Profit]
East	**Colas**	12656
East	**Root Beer**	2534
East	**Cream Soda**	2627
East	**Fruit Soda**	6344
East	**Diet Drinks**	2408
West	**Colas**	3549

[Market].Levels(1). [MEMBER_CAPTION]	[Product]. [Family]. [MEMBER_CAPTION]	[Profit]
West	**Root Beer**	9727
West	**Cream Soda**	10731
West	**Fruit Soda**	5854
West	**Diet Drinks**	8087
...	**...**	...

例 5

この例では、CrossJoin を使用して市場および製品のレベル 1-2 を要求しています。

```
   SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(2).AllMembers,[Product].Levels(2).ALLMEMBERS)
 DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS
FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Market].Levels(1). [MEMBER_ CAPTION]	[Market].Levels(2). [MEMBER_ CAPTION]	[Product]. [Family]. [MEMBER_ CAPTION]	[Product]. [SKU]. [MEMBER_ CAPTION]	[Profit]
East	**New York**	**Colas**	**Cola**	3498
East	**New York**	**Root Beer**	**Old Fashioned**	-2594
East	**New York**	**Root Beer**	**Birch Beer**	3086
East	**New York**	**Cream Soda**	**Dark Cream**	2496
East	**New York**	**Cream Drinks**	**Vanilla Cream**	-1952
East	**New York**	**Fruit Soda**	**Grape**	1329
East	**New York**	**Fruit Soda**	**Orange**	1388
East	**New York**	**Fruit Soda**	**Strawberry**	951
...	**...**	**...**	**...**	...

例 6

次の例では、CrossJoin を使用して複数次元を表現し、各次元の異なる数のレベルを要求し、複数のプロパティを要求します。

```
   SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY Crossjoin ([Market].Levels(1).AllMembers,[Product].Levels(2).ALLMEMBERS)
 DIMENSION PROPERTIES MEMBER_CAPTION, LEVEL_NUMBER
ON ROWS
```

```
FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Market]. Levels(1). [MEMBER_ CAPTION]	[Market]. Levels(1). [LEVEL_ NUMBER]	[Product]. [Family]. [MEMBER_ CAPTION]	[Market]. Levels(1). [LEVEL_ NUMBER]	[Product]. [SKU]. [MEMBER_ CAPTION]	[Market]. Levels(1). [LEVEL_ NUMBER]	[Profit]
East	1	**Colas**	1	**Cola**	2	11129
East	1	**Colas**	1	**Diet Cola**	2	1114
East	1	**Colas**	1	**Caffeine Free Cola**	2	413
East	1	**Root Beer**	1	**Old Fashioned**	2	-2540
East	1	**Root Beer**	1	**Diet Root Beer**	2	982
East	1	**Root Beer**	1	**Birch Beer**	2	4092
East	1	**Cream Soda**	1	**Dark Cream**	2	3233
East	1	**Cream Soda**	1	**Vanilla Cream**	2	-918
...	...	**...**	...	**...**	...	...

例 7

次の例では、複数のネストした CrossJoin が使用されています。

```
   SELECT NON EMPTY { [Profit] } ON COLUMNS,
NON EMPTY {CROSSJOIN
          (
           CROSSJOIN( [Market].Levels(1).ALLMEMBERS,
                [Product].[Family].ALLMEMBERS
               ),
           [Year].Levels(1).ALLMEMBERS
          )
     } DIMENSION PROPERTIES MEMBER_CAPTION
ON ROWS FROM Sample.Basic
```

このクエリーには、次の結果があります(抜粋):

[Market].Levels(1). [MEMBER_ CAPTION]	[Product]. [Family]. [MEMBER_ CAPTION]	[Year].Levels(1). [MEMBER_ CAPTION]	[Profit]
East	Colas	Qtr1	2747
East	Colas	Qtr2	3352
East	Colas	Qtr3	3740
East	Colas	Qtr4	2817

[Market].Levels(1). [MEMBER_ CAPTION]	[Product]. [Family]. [MEMBER_ CAPTION]	[Year].Levels(1). [MEMBER_ CAPTION]	[Profit]
East	Root Beer	Qtr1	562
East	Root Beer	Qtr2	610
East	Root Beer	Qtr3	372
East	Root Beer	Qtr4	990
...	...	...	...

XMLA の例

次の例では、XMLA の応答と要求を示しています。

これはフラット化した行セット要求の例です。結果をフラット化するには、例で示すように PropertyList 要素のテーブル・フォーマットを使用する必要があります。

```
  <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
 <Execute xmlns="urn:schemas-microsoft-com:xml-analysis"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <Command>
   <Statement>
   WITH MEMBER [Year].[calctest] AS '4'
   SELECT NON EMPTY { [Profit] } ON COLUMNS,
   NON EMPTY {[Year].ALLMEMBERS } ON ROWS
   FROM Sample.Basic
   </Statement>
  </Command>
  <Properties>
   <PropertyList>
   <DataSourceInfo>Provider=Essbase;Data Source=localhost
   </DataSourceInfo>
   <Catalog>Sample</Catalog>
   <Format>Tabular</Format>
   <AxisFormat>TupleFormat</AxisFormat>
   </PropertyList>
  </Properties>
 </Execute>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

フラット化した行セット応答の例:

```
  <?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
```

```
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <SOAP-ENV:Body>
  <m:ExecuteResponse xmlns:m="urn:schemas-microsoft-com:xml-analysis">
   <m:return
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <root xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:schema xmlns="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     targetNamespace="urn:schemas-microsoft-com:xml-analysis:rowset"
     xmlns:sql="urn:schemas-microsoft-com:xml-sql"
     elementFormDefault="qualified">
     <xsd:element name="root">
      <xsd:complexType>
       <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="row" type="row" />
       </xsd:sequence>
      </xsd:complexType>
     </xsd:element>
     <xsd:complexType name="row">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
       <xsd:element name="column1" type="xsd:string"
        sql:field="[Year].Levels(1).[MEMBER_CAPTION]" minOccurs="0"/>
       <xsd:element name="column2" type="xsd:string"
        sql:field="[Year].Levels(2).[MEMBER_CAPTION]" minOccurs="0"/>
       <xsd:element name= "column3" type="xsd:double"
        sql:field= "[Profit]" minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>

    <row>
     <column3>105522.000000</column3>
    </row>
    <row>
     <column1>Qtr1</column1>
     <column3>24703.000000</column3>
    </row>
    <row>
     <column1>Qtr1</column1>
     <column2>Jan</column2>
     <column3>8024.000000</column3>
    </row>
    <row>
     <column1>Qtr1</column1>
     <column2>Feb</column2>
     <column3>8346.000000</column3>
    </row>
    <row>
     <column1>Qtr1</column1>
     <column2>Mar</column2>
     <column3>8333.000000</column3>
    </row>
    <row>
     <column1>Qtr2</column1>
```

```
 <column3>27107.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
 <column2>Apr</column2>
 <column3>8644.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
 <column2>May</column2>
 <column3>8929.000000</column3>
</row>
<row>
 <column1>Qtr2</column1>
 <column2>Jun</column2>
 <column3>9534.000000</column3>
</row>
<row>
 <column1>Qtr3</column1>
 <column3>27912.000000</column3>
</row>
<row>
 <column1>Qtr3</column1>
 <column2>Jul</column2>
 <column3>9878.000000</column3>
</row>
<row>
 <column1>Qtr3</column1>
 <column2>Aug</column2>
 <column3>9545.000000</column3>
</row>
<row>
 <column1>Qtr3</column1>
 <column2>Sep</column2>
 <column3>8489.000000</column3>
</row>
<row>
 <column1>Qtr4</column1>
 <column3>25800.000000</column3>
</row>
<row>
 <column1>Qtr4</column1>
 <column2>Oct</column2>
 <column3>8653.000000</column3>
</row>
<row>
 <column1>Qtr4</column1>
 <column2>Nov</column2>
 <column3>8367.000000</column3>
</row>
<row>
 <column1>Qtr4</column1>
 <column2>Dec</column2>
 <column3>8780.000000</column3>
</row>
<row>
 <column1>calctest</column1>
```

```
    <column3>4.000000</column3>
   </row>
  </root>
 </m:return>
</m:ExecuteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A APIサンプル・プログラム

この付録の内容

C API のサンプル・プログラム 1(cs1.c)

このファイルには注釈付きの Essbase C API プログラムが含まれています。この基本的なサンプル・プログラムは、C++プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の C コード・ファイル一式も、Essbase API に含まれています。このドキュメントの samples ディレクトリにある*.c ファイル、実行可能ファイル、プロジェクト、ワークスペースを参照してください。

```
  /*
Copyright 1992-2008 Oracle Corporation. All Rights Reserved.

NAME
cs1.c

DEPENDENCIES
You must add ESSAPIN.LIB to your project.
You must also identify the /API/Include and /API/Lib
directories to the compiler/linker.

DESCRIPTION
This file is used for testing of the Main API and
describing the most fundamental aspects of the Essbase API.
This simple application program is intended as a starting
point for more complex programs. This program performs only
the most basic initialization and login functions. It
connects to a server/application/database, performs only
the most basic of tasks (lists connected users), disconnects,
logs out and terminates. Because all Essbase API programs
must do these things, this program represents the
most simple API program possible. It is applicable in the
```

```
 most general sense to being used as a starting point for
 more useful and complex production-oriented programs.

 NOTES
 This program has three sections:
  1 - The includes and function definitions
  2 - The function declarations
  3 - The main flow

 MODIFIED
 * Created  26 Aug 1999   publications

*/
/*********************************************************/
/*********************************************************/
/*********************************************************/

/*
Declaration of Include files
*/

#if defined _WIN32 || defined _WINDOWS
#include <windows.h>
#endif

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#pragma pack (1)
#include <essapi.h>
#include <essotl.h>
#pragma pack ()

/*
Declaration of handles and connection information variables
*/

ESS_HINST_T  hInst;
ESS_HCTX_T hCtx;
ESS_SVRNAME_T   srvrName  =  "";
ESS_USERNAME_T  userName  =  "";
ESS_PASSWORD_T  pswd    =  "";


/*
Declaration of all the Essbase API functions used in this
program. You could declare all the functions here, and have
them available for the prototype section. This program only
uses a few functions.
*/

/* Initialization and Login functions */
void ESS_Init();
void ESS_AutoLogin();
void ESS_Login();        //This app uses EssAutoLogin().
void ESS_LoginSetPassword();   //I declared these other loginvoid
ESS_AutoLoginSetPassword(); //functions for future use.
```

```
void ESS_Logout();
void ESS_Term();
void ESS_GetVersion();
void ESS_GetAPIVersion();
void ESS_SetActive();
void ESS_ListDatabases();
void ESS_ListUsers();
void ESS_Free();


/************* START FUNCTION DECLARATIONS  ***************/
/**********************************************************/

void ESS_Init()
{
  ESS_STS_T  sts;
  ESS_INIT_T InitStruct = {ESS_API_VERSION,
              NULL,
              0L,
              255,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              0L
             };
 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {
  printf("EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssInit sts: %ld\n", sts);
}

/****************************************************/
/****************************************************/

void ESS_Login ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssLogin (hInst, srvrName, userName,
        pswd, &Items, &pAppsDbs, &hCtx);
 printf("EssLogin sts: %ld\r\n", sts);
 if ( (sts == 1051093L) || (sts == 1051090L) )
 {  ESS_LoginSetPassword(); }
 else
 if ( (sts != 0) && (sts != 1051093L) && (sts != 1051090L) )
 {
  printf("\n\tUsage: ");
  printf("MAINAPI servername username password\n");
  printf("\tDefault: \n\tserver name: local\n\t");
```

```
  printf("user name: admin\n\tpassword: password\n");

  exit ((int) sts);
 }
}

/*****************************************************/
/*****************************************************/

void ESS_AutoLogin ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_CHAR_T SvrName[ESS_SVRNAMELEN];   //this is different in VC++6
 ESS_CHAR_T UserName[ESS_USERNAMELEN];
 ESS_CHAR_T Password[ESS_PASSWORDLEN];
 ESS_CHAR_T AppName[ESS_APPNAMELEN];
 ESS_CHAR_T DbName[ESS_DBNAMELEN];

 ESS_USHORT_T  Option;
 ESS_ACCESS_T  Access ;
//  ESS_HCTX_T   hCtx; Don't set this again, it is set in EssInit

 /* Initialize parameters */
 strcpy(SvrName,"localhost");
 strcpy(UserName,"Admin");
 strcpy(Password,"Password");
 strcpy(AppName,"");
 strcpy(DbName,"");
 Option = AUTO_DEFAULT;

 /* Login to Essbase Server */
 sts = EssAutoLogin (hInst, SvrName, UserName, Password,
       AppName, DbName, Option, &Access, &hCtx);
 printf("EssAutoLogin sts: %ld\r\n", sts);
}

/*****************************************************/
/*****************************************************/

void ESS_LoginSetPassword()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 ESS_PASSWORD_T newPswd = "password2";

 sts = EssLoginSetPassword (hInst, srvrName, userName, pswd, newPswd,
             &Items, &pAppsDbs, &hCtx);
 printf("EssLoginSetPassword sts: %ld\r\n", sts);
 if (sts)
 {  printf("\n\tEssLoginSetPassword sts: %ld\n",sts);
  exit ((int) sts);
 }
}

/*****************************************************/
```

```
/*****************************************************/

void ESS_GetAPIVersion()
{
 ESS_STS_T  sts = ESS_STS_NOERR;
 ESS_ULONG_T  Version;

 sts = EssGetAPIVersion(&Version);

 if(!sts)
  printf("API Version %#x\n",Version);
}

/*****************************************************/
/*****************************************************/

void ESS_Term()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
 {
  /* error terminating API */
  exit((ESS_USHORT_T) sts);
 }
 printf("EssTerm sts: %ld\r\n", sts);
}

/*****************************************************/
/*****************************************************/

void ESS_Logout()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 sts = EssLogout (hCtx);
 printf("\n\nEssLogout sts: %ld\n",sts);
}

/*****************************************************/
/*****************************************************/

void ESS_GetVersion()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Release;
 ESS_USHORT_T Version;
 ESS_USHORT_T Revision;

 sts = EssGetVersion (hCtx, &Release, &Version, &Revision);
 printf("EssGetVersion sts: %ld\r\n", sts);

 if(!sts)
 {
  printf("\r\nEssbase Application Server - ");
  printf("Version %d.%d.%d\r\n", Release, Version, Revision);

 }
```

```
}

/*****************************************************/
/*****************************************************/

void ESS_SetActive()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_ACCESS_T Access;
 ESS_STR_T AppName;
 ESS_STR_T DbName;

 AppName = "sample";
 DbName = "basic";
 sts = EssSetActive(hCtx, AppName, DbName, &Access);
 printf("EssSetActive sts: %ld\r\n",sts);
}

/*****************************************************/
/*****************************************************/

void ESS_ListDatabases()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssListDatabases(hCtx, NULL, &Items, &pAppsDbs);
 printf("EssListDatabases sts: %ld\r\n",sts);

 if(!sts)
 {
  if(Items && pAppsDbs)
  {
   printf("\r\n--Applications/databases available--\r\n");
   for (ind = 0; ind<Items; ind++)
   {
    if((pAppsDbs+ind) !=NULL)
    {
     if((pAppsDbs[ind].AppName != NULL)
      && (pAppsDbs[ind].DbName != NULL))
     {
      printf("%s",pAppsDbs[ind].AppName);
      printf(" ==> ");
      printf("%s",pAppsDbs[ind].DbName);
      printf("\n\r");
     }
    }
   }
   EssFree(hInst, pAppsDbs);
  }
  else
   printf("\r\nDatabase List is Empty\r\n\r\n");
 }
}
```

```
/*****************************************************/
/*****************************************************/

void ESS_ListUsers()
{

 ESS_STS_T    sts;
 ESS_USHORT_T   Count;
 ESS_PUSERINFO_T  Users = NULL;
 ESS_USHORT_T   ind;

 sts = EssListUsers (hCtx, NULL, NULL, &Count, &Users);
 if (!sts)
 {
  if (Count && Users)
  {
   printf ("\r\n-----User List from EssListUsers()-----\r\n\r\n");
   for (ind = 0; ind < Count; ind++)
   {
    printf ("Name->%s\tApplication->%s\tdatabase->%s\r\n",
        Users[ind].Name, Users[ind].AppName,
         Users[ind].DbName);
    // printf("Login %d\r\n",Users[ind].Login);
    // printf("Type %d\r\n",Users[ind].Type);
    // printf("Access %d\r\n",Users[ind].Access);
    // printf("MaxAccess %d\r\n",Users[ind].MaxAccess);
    // printf("Expiration %d\r\n",Users[ind].Expiration);
    // printf("LastLogin %d\r\n",Users[ind].LastLogin);
    // printf("FailCount %d\r\n",Users[ind].FailCount);
    // printf("LoginId %ld\r\n",Users[ind].LoginId);
   }
    // printf("end of userlist %d\r\", count);
   printf ("\r\n----User List from EssListUsers()-----\r\n\r\n");
   EssFree (hInst, Users);
   printf("\r\n");
  }
  else
   printf ("\r\nUsers list is empty\r\n\r\n");
 }
}

/*****************************************************/
/*************** MAIN FUNCTION ***********************/

/*
This is the actual program. It initializes and logs with EssAutoLogin,
then gets the Essbase Server version and the version of the API. It
sets the active application and lists the users connected to the
application. The output consists of simple printf statements.
*/
main()
{
 ESS_Init();
 ESS_AutoLogin();

/*
Every Essbase API program must issue EssInit to get the context
```

```
handle (hCtx). The EssLogin is required to connect to a
database/application. Almost any functions can follow the Init and Login.
We used EssAutoLogin to display the Connect dialog box, but this
program could have used EssLogin and retrieve the Username and
Password as command line arguments. Following sample programs will
illustrate the use of command line arguments.
*/

/*
The following statements perform some of the most simple actions. The
output, in the form of printf statements, is done by the individual
functions. The EssFree functions that release allocated memory are
also in the individual functions. More complex programs will not free
memory in the individual functions because the allocated structures
and handles are needed until the end.

These simple actions can easily be more complex.  Additional operations
would be added in this section.  Following sample programs will do more,
but this program merely retrieves some basic information and displays it.
*/

 ESS_GetVersion();
 ESS_GetAPIVersion();
 ESS_SetActive();
 ESS_ListDatabases();
 ESS_ListUsers();
/*
The EssLogout disconnects the user from the Essbase Server, application,
and database.  The EssTerm ends the program and frees allocated memory,
such as the context handle.
*/
 ESS_Logout();
 ESS_Term();
}

/*
End of program
*/
```

C API のサンプル・プログラム 2(cs2.c)

このファイルには注釈付きの Essbase C API プログラムが含まれています。この基本的なサンプル・プログラムは、C++プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の C コード・ファイル一式も、Essbase API に含まれています。samples ディレクトリにある*.c ファイル、実行可能ファイル、プロジェクト、ワークスペースを参照してください。

```
  /*
 Copyright 1992-2008 Oracle Corporation. All Rights Reserved.
```

```
 NAME
 cs2.c

 DEPENDENCIES


 DESCRIPTION
 This file is used as an example of a simple
 applications program. This program performs basic initialization and
 login and queries the active application/database. It then manipulates
 the user list, adding, renaming, and deleting a new user.

 NOTES
 This program has three sections:
  1 - The includes and function definitions
  2 - The function declarations
  3 - The main flow


 MODIFIED
 * Modified  03 Sep 1999   Publications

*/
/*********************************************************************/
/*****************  START FUNCTION DEFINITIONS  **********************/
/*********************************************************************/

#if defined _WIN32 || defined _WINDOWS
#include <windows.h>
#endif

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#pragma pack (1)
#include <essapi.h>
#include <essotl.h>
#pragma pack ()


ESS_HINST_T  hInst;
ESS_HCTX_T hCtx;
ESS_SVRNAME_T   srvrName  =  "";
ESS_USERNAME_T  userName  =  "";
ESS_PASSWORD_T  pswd    =  "";

/* Initialization and Login functions */
void ESS_Init();
// void ESS_Login();      /* Requires command line arguments */
void ESS_Logout();
void ESS_Term();
void ESS_AutoLogin();     /* Displays the login dialog box */
void ESS_LoginSetPassword();  /* Called if EssAutoLogin returns error */
void ESS_GetVersion();
void ESS_GetAPIVersion();

/* Application functions */
```

```
void ESS_SetActive();
// void ESS_GetActive();
void ESS_ListApplications();
void ESS_ListDatabases();
void ESS_GetDatabaseInfo();

void ESS_ListUsers();  /* These functions will be called repeatedly */
void ESS_CreateUser (); /* to create a user, list users, rename the */
void ESS_RenameUser(); /* new user, list users again, then delete */
void ESS_DeleteUser(); /* the new users and list users again    */
void ESS_GetUserInfo ();

/*********************************************************************/
/***************** START FUNCTION DECLARATIONS ********************/
/*********************************************************************/
void ESS_Init()
{
  ESS_STS_T  sts;
  ESS_INIT_T InitStruct = {ESS_API_VERSION,
             NULL,
             0L,
             255,
             NULL,
             NULL,
             NULL,
             NULL,
             NULL,
             NULL,
             NULL,
             0L
             };
 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {    printf("EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssInit sts: %ld\n", sts);
}

/*********************************************************************/
void ESS_Login ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssLogin (hInst, srvrName, userName, pswd, &Items,
        &pAppsDbs, &hCtx);
 printf("EssLogin sts: %ld\r\n", sts);
 if ( (sts == 1051093L) || (sts == 1051090L) )
 {  ESS_LoginSetPassword(); }
 else
 if ( (sts != 0) && (sts != 1051093L) && (sts != 1051090L) )
 {  printf("\n\tUsage: MAINAPI servername username password\n");
  printf("\tDefault: \n\tserver name: local\n\t");
  printf("user name: admin\n\tpassword: password\n");

  exit ((int) sts);
```

```
 }
}

/*********************************************************************/
void ESS_AutoLogin ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_CHAR_T SvrName[ESS_SVRNAMELEN];   //this is different in VC++6
 ESS_CHAR_T UserName[ESS_USERNAMELEN];
 ESS_CHAR_T Password[ESS_PASSWORDLEN];
 ESS_CHAR_T AppName[ESS_APPNAMELEN];
 ESS_CHAR_T DbName[ESS_DBNAMELEN];

 ESS_USHORT_T  Option;
 ESS_ACCESS_T  Access ;
//  ESS_HCTX_T   hCtx;  Don't set this again, it is set at the top

 /* Initialize parameters */
 strcpy(SvrName,"localhost");
 strcpy(UserName,"Admin");
 strcpy(Password,"Password");
 strcpy(AppName,"");
 strcpy(DbName,"");
 Option = AUTO_DEFAULT;

 /* Login to Essbase Server */
 sts = EssAutoLogin (hInst, SvrName, UserName, Password,
       AppName, DbName, Option, &Access, &hCtx);
 printf("EssAutoLogin sts: %ld\r\n", sts);
}

/*********************************************************************/
void ESS_LoginSetPassword()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 ESS_PASSWORD_T newPswd = "password2";

 sts = EssLoginSetPassword (hInst, srvrName, userName, pswd, newPswd,
             &Items, &pAppsDbs, &hCtx);
 printf("EssLoginSetPassword sts: %ld\r\n", sts);
 if (sts)
 {  printf("\n\tEssLoginSetPassword sts: %ld\n",sts);
  exit ((int) sts);
 }
}

/*********************************************************************/
void ESS_GetAPIVersion()
{
 ESS_STS_T   sts = ESS_STS_NOERR;
 ESS_ULONG_T  Version;

 sts = EssGetAPIVersion(&Version);
```

```
 if(!sts)
  printf("API Version %#x\n",Version);
}

/*********************************************************************/
void ESS_Term()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
 {
  /* error terminating API */
  exit((ESS_USHORT_T) sts);
 }
 printf("EssTerm sts: %ld\r\n", sts);
}

/*********************************************************************/
void ESS_Logout()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 sts = EssLogout (hCtx);
 printf("\n\nEssLogout sts: %ld\n",sts);
}

/*********************************************************************/
void ESS_GetVersion()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Release;
 ESS_USHORT_T Version;
 ESS_USHORT_T Revision;

 sts = EssGetVersion (hCtx, &Release, &Version, &Revision);
 printf("EssGetVersion sts: %ld\r\n", sts);

 if(!sts)
 {
  printf("\r\nEssbase Application Server -  ");
  printf("Version %d.%d.%d\r\n", Release, Version, Revision);
 }
}

/*********************************************************************/
void ESS_GetActive()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T pDbName;
 ESS_STR_T pAppName;
 ESS_ACCESS_T Access;

 if((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pAppName)) == 0)
 {
  if((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pDbName)) == 0)
  {
   if((sts =
     EssGetActive(hCtx, &pAppName, &pDbName, &Access)) == 0)
```

```
    {
     if(pAppName)
     {
      if(*pAppName)
       printf("Current active app: [%s]\r\n",pAppName);
      else
       printf("No active Application is set\r\n");
     }
     EssFree(hInst, pDbName);
    }
    EssFree(hInst, pAppName);
   }
  }
}

/*********************************************************************/
void ESS_SetActive()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_ACCESS_T Access;
 ESS_STR_T AppName;
 ESS_STR_T DbName;

 AppName = "sample";
 DbName = "basic";
 sts = EssSetActive(hCtx, AppName, DbName, &Access);
 printf("EssSetActive sts: %ld\r\n",sts);
}

/*********************************************************************/
void ESS_ListApplications()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_PAPPNAME_T strp = NULL;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;

 sts = EssListApplications(hCtx, &Items, &strp);
 if(!sts)
 {
  if(Items && strp)
  {
   printf("Applications availables\r\n");
   for(ind = 0; ind <Items; ind++)
   {
    if(strp[ind] != NULL)
     printf("%s\r\n", strp[ind]);
   }
   EssFree(hInst, strp);
  }
  else
   printf("\r\nApplication List is Empty\r\n\r\n");
 }
}

/*********************************************************************/
void ESS_ListDatabases()
```

```
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssListDatabases(hCtx, NULL, &Items, &pAppsDbs);
 printf("EssListDatabases sts: %ld\r\n",sts);

 if(!sts)
 {
  if(Items && pAppsDbs)
  {
   printf("\r\n--Applications/databases available--\r\n");
   for (ind = 0; ind<Items; ind++)
   {
    if((pAppsDbs+ind) !=NULL)
    {
     if((pAppsDbs[ind].AppName != NULL)
       && (pAppsDbs[ind].DbName != NULL))
     {
      printf("%s",pAppsDbs[ind].AppName);
      printf(" ==> ");
      printf("%s",pAppsDbs[ind].DbName);
      printf("\n\r");
     }
    }
   }
   EssFree(hInst, pAppsDbs);
  }
  else
   printf("\r\nDatabase List is Empty\r\n\r\n");
 }
}

/*********************************************************************/
void ESS_GetDatabaseInfo()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_PDBINFO_T DbInfo;
 ESS_STR_T AppName;
 ESS_STR_T DbName;

 AppName = "Sample";
 DbName = "Basic";

 sts = EssGetDatabaseInfo(hCtx, AppName, DbName, &DbInfo);
 if(!sts)
 {
  printf("\r\n----- Results of EssGetDatabaseInfo -----\r\n");
  printf("AppName: %s\r\n",DbInfo->AppName);
  printf("DbName: %s\r\n",DbInfo->Name);
  printf("DbType: %d\r\n",DbInfo->DbType);
  printf("Status: %d\r\n",DbInfo->Status);
  printf("nConnects: %d\r\n",DbInfo->nConnects);
  printf("nLocks: %d\r\n",DbInfo->nLocks);
  printf("nDims: %d\r\n",DbInfo->Data);
```

```
   printf("Country: %s\r\n",DbInfo->Country);
   printf("Time: %s\r\n",DbInfo->Time);
   printf("Category: %s\r\n",DbInfo->Category);
   printf("Type: %s\r\n",DbInfo->Type);
   printf("CrPartition: %s\r\n",DbInfo->CrPartition);
   printf("\r\n----- Results of EssGetDatabaseInfo -----\r\n");

   if(DbInfo)
   {
    EssFree(hInst, DbInfo);
   }
  }
}

/*********************************************************************/
void ESS_ListUsers()
{

 ESS_STS_T    sts;
 ESS_USHORT_T   Count;
 ESS_PUSERINFO_T  Users = NULL;
 ESS_USHORT_T   ind;

 sts = EssListUsers (hCtx, NULL, NULL, &Count, &Users);
 if (!sts)
 {
  if (Count && Users)
  {
   printf ("\r\n-----User List from EssListUsers()-----\r\n\r\n");
   for (ind = 0; ind < Count; ind++)
   {
    printf ("Name->%s\tApplication->%s\tdatabase->%s\r\n",
        Users[ind].Name, Users[ind].AppName,
         Users[ind].DbName);
   }
   printf ("\r\n-----User List from EssListUsers()-----\r\n\r\n");
   EssFree (hInst, Users);
  printf("\r\n");
  }
  else
   printf ("\r\nUsers list is empty\r\n\r\n");
 }
}

/*********************************************************************/
void ESS_CreateUser()
{
 ESS_STS_T   sts = ESS_STS_NOERR;

 ESS_CHAR_T UserName[] = "newuser";  //this is different in VC++6
 ESS_CHAR_T Password[] = "password"; //compare to API reference example

 printf("Begin EssCreateUser Function");

 sts = EssCreateUser (hCtx, UserName, Password);
 printf("EssCreateUser sts: %ld",sts);
}
```

```
/*********************************************************************/
void ESS_RenameUser()
{
 ESS_STS_T   sts = ESS_STS_NOERR;
 ESS_CHAR_T OldName[] = "newuser";
 ESS_CHAR_T NewName[] = "user4";

 sts = EssRenameUser (hCtx, OldName, NewName);
}

/*********************************************************************/
void ESS_DeleteUser()
{
 ESS_STS_T   sts = ESS_STS_NOERR;
 ESS_CHAR_T UserName[] = "user4";

 sts = EssDeleteUser (hCtx, UserName);
 printf("EssDeleteUser sts: %ld",sts);
}

/*********************************************************************/
void ESS_GetUserInfo ()
{
 ESS_STS_T    sts  = ESS_STS_NOERR;
 ESS_PUSERINFO_T  User = NULL;

 sts = EssGetUser (hCtx, "Jim Smith", &User);
 printf("EssGetUserInfo %ld\r\n",sts);

 if (!sts)
 {
   printf ("Name->%s Application->%s database->%s\r\n",
         User->Name, User->AppName, User->DbName);
   printf("Login %d\r\n",User->Login);
   printf("Type %d\r\n",User->Type);
   printf("Access %d\r\n",User->Access);
   printf("MaxAccess %d\r\n",User->MaxAccess);
   printf("Expiration %d\r\n",User->Expiration);
   printf("LastLogin %d\r\n",User->LastLogin);
   printf("FailCount %d\r\n",User->FailCount);
   printf("LoginId %ld\r\n",User->LoginId);

  if (User)
   EssFree (hInst, User);
 }
}

/*********************************************************************/
void getCmdLineArgs(int argc, char *argv[])
{
 if (argc>1)
 strcpy(srvrName,argv[1]);

 if (argc>2)
  strcpy(userName,argv[2]);
```

```
  if (argc>3)
   strcpy(pswd,argv[3]);

  printf("Server name:  %s\n",srvrName);
  printf("User name:  %s\n",userName);
  printf("Password:  %s\n",pswd);
}

/*****************************************************/
/*************** MAIN FUNCTION ***********************/
/*****************************************************/

void main(int argc, char *argv[])
{

  getCmdLineArgs(argc,argv);

  ESS_Init();
  ESS_AutoLogin();
  ESS_GetVersion();
  ESS_GetAPIVersion();

  ESS_SetActive();
  ESS_ListApplications();
  ESS_ListDatabases();
  ESS_GetDatabaseInfo();

  ESS_ListUsers();
  //ESS_CreateUser();
  //ESS_RenameUser();
  //ESS_DeleteUser();
  //ESS_GetUserInfo ();

  ESS_Logout();
  ESS_Term();
}
/*
End of program
*/
```

C API のサンプル・プログラム 3(cs3.c)

このファイルには注釈付きの Essbase C API プログラムが含まれています。この基本的なサンプル・プログラムは、C++プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の C コード・ファイル一式も、Essbase API に含まれています。samples ディレクトリにある*.c ファイル、実行可能ファイル、プロジェクト、ワークスペースを参照してください。

```
   /*
  Copyright 1992-2008 Oracle Corporation. All Rights Reserved.
```

```
 NAME
 cs3.c

 DEPENDENCIES
 You must add ESSAPIN.LIB to your project.
 You must also identify the API/Include and API/Lib
 directories to the compiler/linker.

 DESCRIPTION
 This file is used as an extended example of
 API programming techniques.  This program illustrates the sequence of
 function call expected by the Essbase Server and shows the syntax of
 actual API function calls in an actual working program.

 NOTES
 This program has three sections:
  1 - the includes and function definitions
  2 - the function declarations
  3 - the main program flow

 MODIFIED
  * Created  26July99  Publications
*/

/******************  Includes and Definitions  ***********************/
/*****************************************************************************/

#if defined _WIN32 || defined _WINDOWS
#include <windows.h>
#endif

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#pragma pack (1)
#include <essapi.h>
#include <essotl.h>
#pragma pack ()


ESS_HINST_T  hInst;
ESS_HCTX_T hCtx;
ESS_SVRNAME_T   srvrName  =  "";
ESS_USERNAME_T  userName  =  "";
ESS_PASSWORD_T  pswd    =  "";

/* Initialization and Login functions */
void ESS_Init();
void ESS_Login();
void ESS_Logout();
void ESS_Term();
void ESS_AutoLogin();
void ESS_GetVersion();
void ESS_GetAPIVersion();
void ESS_LoginSetPassword();
```

```
void ESS_SetActive();
// void ESS_GetActive();

void ESS_ListDatabases();
void ESS_UnloadDb();
void ESS_ClearDatabase();

/* Report - updating - Calculation */
void ESS_Report();
void ESS_RunRept ();
void ESS_ReportFile ();

void ESS_Update();
void ESS_UpdateFile();

void ESS_Calc();
void ESS_CalcLine();
void ESS_RunCalc ();
void ESS_CalcFile();
void ESS_Import ();

void ESS_Free();

/*****************  START FUNCTION DECLARATIONS  ******************/
/*********************************************************************/
void ESS_Init()
{
  ESS_STS_T   sts;
  ESS_INIT_T InitStruct = {ESS_API_VERSION,
              NULL,
              0L,
              255,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              NULL,
              0L
             };
 if ((sts = EssInit(&InitStruct, &hInst)) != ESS_STS_NOERR)
 {    printf("EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
 printf("EssInit sts: %ld\n", sts);
}

/*********************************************************************/
void ESS_Login ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssLogin (hInst, srvrName, userName, pswd, &Items, &pAppsDbs,
        &hCtx);
```

```
 printf("EssLogin sts: %ld\r\n", sts);
 if ( (sts == 1051093L) || (sts == 1051090L) )
 {  ESS_LoginSetPassword(); }
 else
 if ( (sts != 0) && (sts != 1051093L) && (sts != 1051090L) )
 {
  printf("\n\tUsage:  MAINAPI servername username password\n");
  printf("\tDefault: \n\tserver name: local\n\t");
  printf("user name:  admin\n\tpassword:  password\n");
  exit ((int) sts);
 }
}

/*********************************************************************/
void ESS_AutoLogin ()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_CHAR_T SvrName[ESS_SVRNAMELEN];   //this is different in VC++6
 ESS_CHAR_T UserName[ESS_USERNAMELEN];
 ESS_CHAR_T Password[ESS_PASSWORDLEN];
 ESS_CHAR_T AppName[ESS_APPNAMELEN];
 ESS_CHAR_T DbName[ESS_DBNAMELEN];

 ESS_USHORT_T  Option;
 ESS_ACCESS_T  Access ;
//  ESS_HCTX_T   hCtx;  Don't set this again, it is set at the top

 /* Initialize parameters */
 strcpy(SvrName,"localhost");
 strcpy(UserName,"Admin");
 strcpy(Password,"Password");
 strcpy(AppName,"");
 strcpy(DbName,"");
 Option = AUTO_DEFAULT;

 /* Login to Essbase Server */
 sts = EssAutoLogin (hInst, SvrName, UserName, Password,
       AppName, DbName, Option, &Access, &hCtx);
 printf("EssAutoLogin sts: %ld\r\n", sts);
}

/*********************************************************************/
void ESS_LoginSetPassword()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_PAPPDB_T pAppsDbs = NULL;
 ESS_PASSWORD_T newPswd = "password2";

 sts = EssLoginSetPassword (hInst, srvrName, userName, pswd, newPswd, &Items,
&pAppsDbs, &hCtx);
 printf("EssLoginSetPassword sts: %ld\r\n", sts);
 if (sts)
 {  printf("\n\tEssLoginSetPassword sts: %ld\n",sts);
  exit ((int) sts);
 }
}
```

```
/*********************************************************************/
void ESS_GetAPIVersion()
{
 ESS_STS_T  sts = ESS_STS_NOERR;
 ESS_ULONG_T Version;

 sts = EssGetAPIVersion(&Version);

 if(!sts)
  printf("API Version %#x\n",Version);
}

/*********************************************************************/
void ESS_Term()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 if ((sts = EssTerm(hInst)) != ESS_STS_NOERR)
 {
  /* error terminating API */
  exit((ESS_USHORT_T) sts);
 }
 printf("EssTerm sts: %ld\r\n", sts);
}

/*********************************************************************/
void ESS_Logout()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 sts = EssLogout (hCtx);
 printf("\n\nEssLogout sts: %ld\n",sts);
}

/*********************************************************************/
void ESS_GetVersion()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Release;
 ESS_USHORT_T Version;
 ESS_USHORT_T Revision;

 sts = EssGetVersion (hCtx, &Release, &Version, &Revision);
 printf("EssGetVersion sts: %ld\r\n", sts);

 if(!sts)
 {
  printf("\r\nEssbase Application Server -  ");
  printf("Version %d.%d.%d\r\n", Release, Version, Revision);
 }
}

/*********************************************************************/
void ESS_GetActive()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T pDbName;
```

```
 ESS_STR_T pAppName;
 ESS_ACCESS_T Access;

 if((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pAppName)) == 0)
 {
  if((sts = EssAlloc (hInst, 80, (ESS_PPVOID_T)&pDbName)) == 0)
  {
   if((sts =
    EssGetActive(hCtx, &pAppName, &pDbName, &Access)) == 0)
   {
    if(pAppName)
    {
     if(*pAppName)
      printf("Current active app: [%s]\r\n",pAppName);
     else
      printf("No active Application is set\r\n");
    }
    EssFree(hInst, pDbName);
   }
   EssFree(hInst, pAppName);
  }
 }
}

/*********************************************************************/
void ESS_SetActive()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_ACCESS_T Access;
 ESS_STR_T AppName;
 ESS_STR_T DbName;

 AppName = "sample";
 DbName = "basic";
 sts = EssSetActive(hCtx, AppName, DbName, &Access);
 printf("EssSetActive sts: %ld\r\n",sts);
}

/*********************************************************************/
void ESS_ListDatabases()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_USHORT_T Items;
 ESS_USHORT_T ind;
 ESS_PAPPDB_T pAppsDbs = NULL;

 sts = EssListDatabases(hCtx, NULL, &Items, &pAppsDbs);
 printf("EssListDatabases sts: %ld\r\n",sts);

 if(!sts)
 {
  if(Items && pAppsDbs)
  {
   printf("\r\n--Applications/databases available--\r\n");
   for (ind = 0; ind<Items; ind++)
   {
    if((pAppsDbs+ind) !=NULL)
```

```
     {
      if((pAppsDbs[ind].AppName != NULL)
        && (pAppsDbs[ind].DbName != NULL))
       {
        printf("%s",pAppsDbs[ind].AppName);
        printf(" ==> ");
        printf("%s",pAppsDbs[ind].DbName);
        printf("\n\r");
       }
     }
    }
    EssFree(hInst, pAppsDbs);
   }
   else
    printf("\r\nDatabase List is Empty\r\n\r\n");
 }
}

/*********************************************************************/
void ESS_ClearDatabase()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 sts = EssClearDatabase(hCtx);
 printf("EssClearDatabase sts:%ld\r\n",sts);
 printf("The database is now empty\n");
}

/
*************************************************************************************/
void ESS_Report()
{
 ESS_STS_T sts = ESS_STS_NOERR;
 ESS_STR_T rString;
 ESS_CHAR_T  pszReportIn[512];
  strcpy(pszReportIn,
 " {TABDELIMIT} \
{SUPALL COLHEADING NAMESON BLOCKHEADERS PAGEHEAD INDENTGEN 2 DECIMALS \
VARIABLE} \
 {BRACKET} \
<SINGLECOLUMN \
<QUOTEMBRNAMES \
 {SUPMISSING} \
<BOTTOM ( 4, @DATACOL(1) ) \
<SYM \
<PAGE( 'Measures') \
'Measures' \
<COL( 'Market','Scenario') \
 { OUTALTNAMES } \
<ICHILDREN 'Market' \
'Actual' \
'Budget' \
<ROW( 'Year','Product') \
<ICHILDREN 'Year' \
<DIMBOTTOM 'Product' \
! " );
```

```
 sts = EssReport (hCtx, ESS_TRUE, ESS_FALSE, pszReportIn);
 //sts = EssReport (hCtx, ESS_TRUE, ESS_FALSE, "<Desc &ThisMonth !");
 //sts = EssReport (hCtx, ESS_TRUE, ESS_FALSE, "<Desc Year !");
 printf("EssReport sts: %ld\r\n",sts);

 if(!sts)
  sts = EssGetString(hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
  printf("%s", rString);
  EssFree (hInst, rString);
  sts = EssGetString (hCtx, &rString);
 }
 printf("\r\n");
}

/*********************************************************************/
void ESS_Update()
{
 ESS_STS_T sts = ESS_STS_NOERR;

 sts = EssUpdate(hCtx, ESS_TRUE, ESS_FALSE,
        "Year Market Scenario Measures Product 123456");
 printf("EssUpdate sts: %ld\r\n",sts);
}

/*********************************************************************/
void ESS_CalcLine()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_STR_T    Script;
 ESS_PROCSTATE_T  pState;

 Script = "CALC DIM (Measures, Product, Market, Year, Scenario);";

 sts = EssCalc(hCtx, ESS_TRUE, Script);
 printf("EssCalc sts: %ld\r\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
     sts = EssGetProcessState (hCtx, &pState);
 }
}

/*********************************************************************/
void ESS_Calc()
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     Script;
 ESS_PROCSTATE_T  pState;
 Script = "CALC ALL;";

 sts = EssBeginCalc (hCtx,ESS_TRUE);
 printf("EssBeginCalc sts: %ld\r\n",sts);
 if (!sts)
```

```
 {
  sts = EssSendString (hCtx, Script);
  printf("EssSendString sts: %ld\r\n",sts);
 }
 if (!sts)
 {
  sts = EssEndCalc (hCtx);
  printf("EssEndCalc sts: %ld\r\n",sts);
 }
 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while(!sts && (pState.State != ESS_STATE_DONE))
    sts = EssGetProcessState (hCtx, &pState);
 }
}

/*********************************************************************/
void ESS_ReportFile ()
{
 ESS_STS_T  sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_STR_T  rString;
 ESS_STR_T  AppName;
 ESS_STR_T  DbName;
 ESS_STR_T  FileName;

 hSrcCtx = hCtx;
 AppName = "Sample";
 DbName  = "Basic";
 FileName = "cdlockdb";

 sts = EssReportFile (hCtx, hSrcCtx, AppName, DbName, FileName,
          ESS_TRUE, ESS_FALSE);
 printf("EssReportFile sts: %ld\r\n",sts);

 if (!sts)
  sts = EssGetString (hCtx, &rString);
 while ((!sts) && (rString != NULL))
 {
  printf ("%s", rString);
  EssFree (hInst, rString);
  sts = EssGetString (hCtx,&rString);
 }
}

/*********************************************************************/
void ESS_UpdateFile ()
{
 ESS_STS_T  sts = ESS_STS_NOERR;
 ESS_HCTX_T  hSrcCtx;
 ESS_BOOL_T  isStore;
 ESS_BOOL_T  isUnlock;
 ESS_STR_T  AppName;
 ESS_STR_T  DbName;
 ESS_STR_T  FileName;
```

```
 AppName  = "Sample";
 DbName   = "Basic";
 hSrcCtx  = hCtx;
 FileName = "cdupdtdb.txt";
 isStore  = ESS_TRUE;
 isUnlock = ESS_FALSE;

 sts = EssUpdateFile (hCtx, hSrcCtx, AppName, DbName, FileName,
          isStore, isUnlock);
 printf("EssUpdateFile sts: %ld\r\n",sts);
}

/*********************************************************************/
void ESS_RunCalc ()
{
 ESS_STS_T    sts = ESS_STS_NOERR;
 ESS_HCTX_T    hSrcCtx;
 ESS_BOOL_T    isObject = ESS_FALSE;
 ESS_STR_T    AppName;
 ESS_STR_T    DbName;
 ESS_STR_T    FileName;
 ESS_PROCSTATE_T  pState;

 hSrcCtx  = hCtx;
 AppName  = "Sample";
 DbName   = "Basic";
 FileName = "calc5dim";

  sts = EssCalcFile (hCtx, hSrcCtx, AppName, DbName, FileName,
           ESS_TRUE);
  printf("EssCalcFile sts: %ld\r\n",sts);

 if (!sts)
 {
  sts = EssGetProcessState (hCtx, &pState);
  while (!sts || (pState.State != ESS_STATE_DONE))
     sts = EssGetProcessState (hCtx, &pState);
 }
}

/*********************************************************************/
void ESS_Import ()
{
 eSS_STS_T    sts = eSS_STS_NOERR;
 eSS_SHORT_T   isAbortOnError;
 eSS_OBJDEF_T  Rules;
 eSS_OBJDEF_T  Data;
 eSS_PMBRERR_T  pMbrErr = NULL;

 Data.hCtx    = hCtx;
 Data.AppName  = "Sample";
 Data.DbName   = "Basic";
 Data.ObjType  = ESS_OBJTYPE_TEXT;
 Data.FileName  = "calcdat.txt";

 Rules.hCtx   = hCtx;
 Rules.AppName  = "Olap";
```

```
 Rules.DbName  = "Demo";
 Rules.ObjType  = eSS_OBJTYPE_RULES;
 Rules.FileName = "Actmap";

  //* Running conditions *

 isAbortOnError = eSS_TRUE;

 sts = EssImport (hCtx, NULL, &Data, &pMbrErr, NULL, isAbortOnError);
 printf("EssImport sts: %ld\r\n",sts);

 if(pMbrErr)
   EssFreeMbrErr(hCtx, pMbrErr);
}


/*********************************************************************/
/*
 This routine gets arguments from the command line.  The routine under
 stands a number of arguments will be present up to 3 arguments total
 The first parameter, argc, is the number of arguments present
 following the command to start (csamp3). The second parameter, argv,
 is the array of arguments. This program (csamp3) has been built to
 override the command line arguments, but could be easily modified to
 use them. In other words, this routine is not used.
*/

void getCmdLineArgs(int argc, char *argv[])
{
 if (argc>1)
 strcpy(srvrName,argv[1]);
 if (argc>2)
   strcpy(userName,argv[2]);
 if (argc>3)
   strcpy(pswd,argv[3]);

  printf("Server name:  %s\n",srvrName);
  printf("User name:  %s\n",userName);
  printf("Password:  %s\n",pswd);
}

/******************  Program Main Flow  ***************************/
/*********************************************************************/

void main(int argc, char *argv[])
{
 getCmdLineArgs(argc,argv);

 /**** Initialization and Login Functions ****/

 ESS_Init();
 ESS_AutoLogin();
 ESS_GetVersion();
 ESS_GetAPIVersion();
 ESS_SetActive();
 ESS_ListDatabases();
```

```
 /**** Report and Updating Calculation ****/
 /*
 This section issues a report to show what is in the database, then
 clears all the data, runs another report to show that the database
 is empty, then imports data from calcdat.txt, then finally, issues
 another report to show that the database now has data.
 */

  ESS_Report();
  ESS_ClearDatabase();
  ESS_Report();
  ESS_Import ();
  ESS_Report();

 /*
 This section runs a calculation from a file. (ESS_RunCalc calls
 EssCalcFile, which specifies the calculation script in the file
 calc5dim.csc.) Then issues yet another report to show the results.
 */

  ESS_CalcLine();
  ESS_Report();

  ESS_ReportFile();
  ESS_UpdateFile();
  ESS_ReportFile();

  ESS_Logout();
  ESS_Term();
}
/*
End of program
*/
```

Visual Basic API サンプル・プログラム 1(initialize.vbp)

このファイルには注釈付きの Essbase Visual Basic API プログラムが含まれています。この基本的なサンプル・プログラムは、Visual Basic プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の VB コード・ファイル一式は、このドキュメントの samples ディレクトリにもあります。

```
   Option Explicit

'*** Always obtain and process the return error status
 Dim lngStatus As Long  ' Return error status

'*** EsbGetAPIVersion()needs the following
 Dim lngAPIVersion As Long
```

```
'*** EsbInit() accepts an initialization structure
'***  and returns an instance handle
 Dim structInit  As ESB_INIT_T  ' Create an instance of the initialization structure
 Dim lngInstHndl As Long    ' Instance handle for program (returned by EsbInit())

'*** EsbGetMessage() (enabled by cmdInit()) needs the following
'*** (see EsbListMessages() for intMsgLen and strMsg)
 Dim intMsgLev  As Integer  ' Whether information/warning/serious error/fatal error
 Dim lngMsgNmbr As Long   ' Message number in Essbase.mdb

'*** EsbAutoLogin()needs the following
 Dim strServer  As String * ESB_SVRNAMELEN  ' Empty string okay
 Dim strUser   As String * ESB_USERNAMELEN  ' Empty string okay
 Dim strPassword As String * ESB_PASSWORDLEN  ' Empty string okay
 Dim strAppName  As String * ESB_APPNAMELEN  ' Empty string okay
 Dim strDbName  As String * ESB_DBNAMELEN   ' Empty string okay
 Dim intOption  As Integer  ' Flags whether to display dialog box, allow user to log
             '  in without selecting the application/database, or
             '  allow user to interact with dialog box to log in and
             '  select the application/database
 Dim intAccess  As Integer  ' User's access level to application/database
 Dim lngCtxHndl As Long   ' Context handle for login (returned by EsbAutoLogin()

'***
' Initialized, logged in, able to log out, able to terminate
' Ready to work with databases, users, objects
'***

'*** MORE DECLARATIONS HERE OR IN SUB PROCEDURES
 Dim intArrayIndex As Integer ' Declare an integer, for example


Private Sub ESB_ListErrorStackMsgs()

'*** EsbGetMessage() needs the following
'*** (see Declarations for intMsgLev and lngMsgNmbr)
 Const intMsgLen = 256       ' Set maximum message length as a constant,
 Dim strMsg As String * intMsgLen  '  then Dim message string at that length

'*** Get all messages from error stack and display them in list box
 lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, strMsg, intMsgLen)
            ' Retrieves strMsg from stack and decrements stack pointer

 Dim intStackNmbr As Integer  ' To track the number of messages on the error stack
 intStackNmbr = 1

 Do While Mid$(strMsg, 1, 1) <> Chr$(0)  ' Do while the error stack has messages
  MsgBox "Error stack #" & (intStackNmbr) & " is level #" & (intMsgLev) _
     & "/message #" & (lngMsgNmbr)
  intStackNmbr = intStackNmbr + 1    ' Increment the stack number displayed
  lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, strMsg, intMsgLen)
 Loop
End Sub

Private Sub cmdAutoLogin_Click()
```

```
 intOption = ESB_AUTO_DEFAULT  ' Allows user to interact with login dialog box

'*** Call EsbAutoLogin() and obtain the return error status
 lngStatus = EsbAutoLogin(lngInstHndl, _
            strServer, strUser, strPassword, _
            strAppName, strDbName, _
            intOption, _
            intAccess, _
            lngCtxHndl)  ' EsbAutoLogin() returns a unique
                  '  context handle for each login, even if the
                  '  user and server are the same

'*** Display the return error status
 If lngStatus = 0 Then
  MsgBox "This login ID (context handle) is logged in: " & (lngCtxHndl)
  Call ESB_ListErrorStackMsgs  ' Even successful logins return useful messages
  cmdAutoLogin.Enabled = False  ' True would allow other login IDs (context handles)
cmdLogout.Enabled = True
  cmdLogout.Enabled = True    ' Log out;
  cmdTerm.Enabled = False    '  then terminate the API
 Else
  MsgBox "Login failed: " & (lngStatus)
  Call ESB_ListErrorStackMsgs  ' Always handle messages if function call fails
 End If
End Sub

Private Sub cmdGetAPIVers_Click()

'***
' You can call EsbGetAPIVersion() before or after you call EsbInit()
'***

'*** Call EsbGetAPIVersion() and obtain the return error status
 lngStatus = EsbGetAPIVersion(lngAPIVersion)

'*** Display the API version or that the call failed
 If lngStatus = 0 Then
  MsgBox "The API version is " & (lngAPIVersion)
 Else
  MsgBox "EsbGetAPIVersion() failed: " & (lngStatus)
 End If
End Sub

Private Sub cmdInit_Click()

'*** Initialize the structure before you call EsbInit()
 structInit.Version = ESB_API_VERSION
 structInit.MaxHandles = 10
 structInit.LocalPath = "C:\\Hyperion\\products\\Essbase\\EssbaseClient"  '
<ARBORPATH>\Client is the default
 structInit.MessageFile = ""        ' The default message file
 structInit.ClientError = ESB_TRUE      ' Enables EsbGetMessage() to retrieve
                     '  top message in stack
 structInit.ErrorStack = 100        ' No. of messages allowed in stack;
                     '  stack initialized on each call

'*** Call EsbInit() to initialize the API; obtain the return error status
```

```
 lngStatus = EsbInit(structInit, lngInstHndl)

'*** Display the return error status
 If lngStatus = 0 Then
  MsgBox "The API is initialized: " & (lngInstHndl)
  cmdAutoLogin.Enabled = True  ' You can log in only after you initialize the API
  cmdInit.Enabled = False    ' Initialization endures until you terminate the API
  cmdTerm.Enabled = True
 Else
  MsgBox "The API failed to initialize: " & (lngStatus)
 End If
End Sub

Private Sub cmdLogout_Click()

'*** Call EsbLogout() and obtain return error status
 lngStatus = EsbLogout(lngCtxHndl)  ' Logs user out for the specified login context

'*** Display whether the logout succeeded or failed
 If lngStatus = 0 Then  ' Should test that all login IDs (contexts) are logged out
  MsgBox "This login ID (context handle) is logged out: " & (lngCtxHndl)
  cmdLogout.Enabled = False  ' Log out;
  cmdTerm.Enabled = True   '  then terminate the API
 Else
  MsgBox "EsbLogout() failed: " & (lngStatus)
 End If
End Sub

Private Sub cmdTerm_Click()

'*** Call EsbTerm() after all other calls are completed
 EsbTerm (lngInstHndl)

'*** Display whether the API terminated
 If lngStatus = 0 Then
  MsgBox "The API is terminated"
  cmdGetAPIVers.Enabled = True  ' After you terminate the API,
  cmdInit.Enabled = True    ' you can call only EsbInit() and EsbGetVersion()
  cmdTerm.Enabled = False
  cmdAutoLogin.Enabled = False
 Else
  MsgBox "EsbTerm() failed: " & (lngStatus)
 End If
End Sub

Private Sub Form_Load()

' *** Must set boolean values in the form
 ESB_TRUE = 1  ' ESB_TRUE
 ESB_FALSE = 0  '  and ESB_FALSE are variables, not constants
End Sub
```

Visual Basic API サンプル・プログラム 2(appdb.vbp)

このファイルには注釈付きの Essbase Visual Basic API プログラムが含まれています。この基本的なサンプル・プログラムは、Visual Basic プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の VB コード・ファイル一式は、このドキュメントの samples ディレクトリにもあります。

フォーム内のコード

このコードはフォーム自体に付属しています。Code.bas(後続)の関数を呼び出します。Code.bas は他のプロジェクトに含めることもできます。

```
   Private Sub Form_Load()
 Call SetBeforeStart
End Sub

Private Sub SetBeforeStart()

 cmdStart.Enabled = True
 cmdStop.Enabled = False
 cmdClearMsg.Enabled = False
 lstMessages.Enabled = False

 cmdListApps.Enabled = False
 cmdListDbs.Enabled = False
 cmdGetActive.Enabled = False
 cmdSetActive.Enabled = False
 cmdGetDbInfo.Enabled = False

End Sub

Private Sub SetAfterLogin()

 cmdStart.Enabled = False
 cmdStop.Enabled = True
 cmdClearMsg.Enabled = True
 lstMessages.Enabled = True

 cmdListApps.Enabled = True
 cmdListDbs.Enabled = True
 cmdGetActive.Enabled = True
 cmdSetActive.Enabled = True
 cmdGetDbInfo.Enabled = True

End Sub

Private Sub cmdClearMsg_Click()
 lstMessages.Clear
End Sub
```

Code.bas モジュールのコード

このコードは、code.bas に含まれています。

```
   Option Explicit

'*******************
'RETURN ERROR STATUS
'*******************
 Dim lngStatus As Long

'************
'INIT GLOBAL
'************
 Dim structInit  As ESB_INIT_T
 Dim lngInstHndl As Long

'**********************
'ESB_GetMESSAGE GLOBAL
'**********************
 Dim intMsgLev  As Integer
 Dim lngMsgNmbr As Long

'*****************
'ESB_LOGIN GLOBAL
'*****************

 Dim lngCtxHndl  As Long

'****************************************
'ESB_SetACTIVE and ESB_ClearDATABASE GLOBAL
'****************************************
 Dim strActiveApp  As String
 Dim strActiveDb  As String


'*****************************************
'Init and turn error handle turned off
'*****************************************
Sub ESB_Init()

 ESB_TRUE = 1     ' ESB_TRUE
 ESB_FALSE = 0     ' and ESB_FALSE are variables, not constants

 '***********************
 ' Define init structure
 '***********************
 structInit.Version = ESB_API_VERSION
 structInit.MaxHandles = 10
 structInit.LocalPath = "C:\Hyperion\products\Essbase\EssbaseClient"
 structInit.MessageFile = ""
 structInit.ClientError = ESB_TRUE
 structInit.ErrorStack = 100
```

```
 '******************
 'Initialize the API
 '******************
 lngStatus = EsbInit(structInit, lngInstHndl)
If lngStatus = 0 Then
  MsgBox "The API is initialized: " & (lngInstHndl)
 Else
  MsgBox "The API failed to initialize: " & (lngStatus)
 End If

End Sub

'******************************************************
'Login in user Admin. All login parameters are hardcoded
'******************************************************
Sub ESB_Login()

 Dim strServer  As String * ESB_SVRNAMELEN
 Dim strUser   As String * ESB_USERNAMELEN
 Dim strPassword As String * ESB_PASSWORDLEN
 Dim intNumAppDb As Integer


 strServer = "Localhost"
 strUser = "Admin"
 strPassword = "password"


 lngStatus = EsbLogin(lngInstHndl, _
           strServer, strUser, strPassword, _
           intNumAppDb, _
           lngCtxHndl)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "Admin is logged in, with login ID (context handle) " & (lngCtxHndl)

  Call ESB_ListErrorStackMsgs  ' Even successful logins return useful messages
 Else
  MsgBox "Login failed: " & (lngStatus)
 End If

End Sub

'*******
' Logout
'*******
Sub ESB_Logout()

 lngStatus = EsbLogout(lngCtxHndl)

'******************************************
'Display whether the logout succeeded or failed
'******************************************
```

```
 If lngStatus = 0 Then
  MsgBox "Admin, with login ID (context handle) " & (lngCtxHndl) _
      & ", is logged out"
 Else
  MsgBox "EsbLogout() failed: " & (lngStatus)
 End If

End Sub

'*****************************
' Terminate the VB API
'*****************************
Sub ESB_Term()

EsbTerm (lngInstHndl)

'*********************************
'Display whether the API terminated
'*********************************
 If lngStatus = 0 Then
  MsgBox "The API is terminated"
 Else
  MsgBox "EsbTerm() failed: " & (lngStatus)
 End If

End Sub

'************************************************************
'This is an error checking subroutine that uses EsbGetMessage
'************************************************************
Sub ESB_ListErrorStackMsgs()

 Const intMsgLen = 256
 Dim strMsg As String * intMsgLen


 lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, _
 strMsg, intMsgLen)

 Dim intStackNmbr As Integer

 intStackNmbr = 1

'*********************************************************************
'Do while the error stack has messages and drop messages in a ListBox
'*********************************************************************
 Do While Mid$(strMsg, 1, 1) <> Chr$(0)
  lstMessages "MESSAGE ON ERROR STACK:"
  lstMessages "Stack #" & (intStackNmbr)
  lstMessages "Level #" & (intMsgLev)
  lstMessages "Message #" & (lngMsgNmbr)
  lstMessages (strMsg)
  intStackNmbr = intStackNmbr + 1
  lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, strMsg, intMsgLen)
 Loop

End Sub
```

```
'*********************************************************************
'Gets the names of the caller's current active application and database
'*********************************************************************
Sub ESB_GetActive()

 Const intAppNameSize = ESB_APPNAMELEN
 Const intDbNameSize = ESB_DBNAMELEN

 Dim strAppName   As String * intAppNameSize
 Dim strDbName    As String * intDbNameSize
 Dim intUserAccess  As Integer

 lngStatus = EsbGetActive(lngCtxHndl, strAppName, intAppNameSize, _
             strDbName, intDbNameSize, intUserAccess)

'*********************************
'Error Checking and Message display
'*********************************
 If lngStatus = 0 Then
  MsgBox "EsbGetActive() succeeded"

  If Mid$(strAppName, 1, 1) = Chr$(0) Then
   lstMessages "No active application/database is set"
  Else
   lstMessages (strAppName)
   lstMessages "/ " & (strDbName)
  End If
 Else
  MsgBox "EsbGetActive() failed: " & (lngStatus)
 End If

End Sub

'*********************************************************************
'Gets a database's information structure, which contains non
'user-configurable parameters for the database. Sample Basic Hardcoded.
'*********************************************************************
Sub Esb_GetDbInfo()

 Dim strAppName  As String
 Dim strDbName   As String
 Dim structDbInfo As ESB_DBINFO_T
 Dim structDbReqInfo As ESB_DBREQINFO_T
 Dim intI As Integer

 'Number of database info structures;
 'Applies where database is an empty string
 Dim intNumDbInfo As Integer

 strAppName = "Sample"
 strDbName = "Basic"

 lngStatus = EsbGetDatabaseInfo(lngCtxHndl, strAppName, strDbName, _
               structDbInfo, intNumDbInfo)

'*********************************
```

```
'Error Checking and Message display
'*********************************
 If lngStatus = 0 Then
  MsgBox "You have retrieved a list of database info structures" & Chr(10) _
     & "EsbGetNextItem() will now generate a list"
 Else
  MsgBox "EsbGetDatabaseInfo() failed: " & (lngStatus)
  MsgBox "Note: Sample / Basic are Hardcoded for this Example"
 End If

'*********************************************
'Get database information and display in list box
'*********************************************
 For intI = 1 To intNumDbInfo
   lngStatus = EsbGetNextItem(lngCtxHndl, ESB_DBREQINFO_TYPE, structDbReqInfo)
  If lngStatus = 0 Then
    MsgBox "EsbGetNextItem() succeeded"
   'Return values for the structDbReqInfo.DbReqType:
   ' 0 = Data load
   ' 1 = Calculation
   ' 2 = Outline update
   lstMessages "Type of request is: " & (structDbReqInfo.DbReqType)
   lstMessages "User is: " & (structDbReqInfo.User)
   ' User does not display - none is loading, calculating, or updating outline
   ' BUT, cannot display structDbInfo fields, which is reason for call
  Else
   MsgBox "EsbGetNextItem() failed: " & (lngStatus)

  End If
 Next

End Sub

'*********************************************************
'Lists all applications which are accessible to the caller
'*********************************************************
Sub Esb_ListApps()

 Dim intNumApps As Integer
 Dim strAppName As String * ESB_APPNAMELEN
 Dim intI As Integer  ' Index for loop

 lngStatus = EsbListApplications(lngCtxHndl, intNumApps)


'*********************************
'Error Checking and Message display
'*********************************
 If lngStatus = 0 Then
  MsgBox "You have retrieved the application names" & Chr(10) _
     & "EsbGetNextItem() will now generate a list"
 Else
  MsgBox "EsbListApplications() failed: " & (lngStatus)
 End If


'*******************************************
```

```
'Get list of applications and display in list box
'************************************************
  For intI = 1 To intNumApps

   lngStatus = EsbGetNextItem(lngCtxHndl, ESB_APPNAME_TYPE, ByVal strAppName)

  If lngStatus = 0 Then
   MsgBox "EsbGetNextItem() succeeded"
   lstMessages (strAppName)
  Else
   MsgBox "EsbGetNextItem() failed: " & (lngStatus)
  End If

 Next

End Sub

'************************************************************
'Lists all databases which are accessible to the caller,
'either within a specific application, or on an entire server.
'************************************************************
Sub Esb_ListDbs()

 Dim strAppName  As String
 Dim intNumDbs  As Integer
 Dim structAppDb As ESB_APPDB_T
 Dim intI As Integer  ' Index for loop

 lngStatus = EsbListDatabases(lngCtxHndl, strAppName, intNumDbs)

'*********************************
'Error Checking and Message display
'*********************************
 If lngStatus = 0 Then
  MsgBox "You have retrieved a list of application/database structures" & Chr(10) _
    & "EsbGetNextItem() will now generate a list"
 Else
  MsgBox "EsbListDatabases() failed: " & (lngStatus)
 End If

'*********************************************************
'Get list of applications/databases and display in list box
'*********************************************************
 For intI = 1 To intNumDbs
  lngStatus = EsbGetNextItem(lngCtxHndl, ESB_APPDB_TYPE, structAppDb)

  If lngStatus = 0 Then
   MsgBox "EsbGetNextItem() succeeded"
   lstMessages (structAppDb.AppName)
   lstMessages "/ " & (structAppDb.DbName)
  Else
   MsgBox "EsbGetNextItem() failed: " & (lngStatus)
  End If
 Next

End Sub
```

```
'************************************************
'Sets the caller's active application and database
'************************************************
Sub Esb_SetActive()

 Dim strAppAnswer  As String
 Dim strDbAnswer  As String
 Dim intUserAccess As Integer

'****************************************
'Input boxes allow users to select an app/db
'****************************************
 strAppAnswer = InputBox("Type the Application Name to Set Active. (May be case
sensitive)")

 '
 strDbAnswer = InputBox("Type the Database Name to Set Active. (May be case
sensitive)")


 lngStatus = EsbSetActive(lngCtxHndl, strAppAnswer, strDbAnswer, intUserAccess)


'*********************************
'Error Checking and Message display
'*********************************
 If lngStatus = 0 Then
  MsgBox strAppAnswer & "/" & strDbAnswer & " is now active"
 Else
  MsgBox "EsbSetActive() failed: " & (lngStatus)
 End If

End Sub

Sub lstMessages(strItem As String)
  frmAppDb.lstMessages.AddItem (strItem)
End Sub

Sub lstMessagesClear()
  frmAppDb.lstMessages.Clear
End Sub
```

Visual Basic API サンプル・プログラム 3(reports.vbp)

このファイルには注釈付きの Essbase Visual Basic API プログラムが含まれています。この基本的なサンプル・プログラムは、Visual Basic プログラミング環境でより多機能なプログラムを作成するための開始点として使用できます。

このファイルは『Oracle Essbase API リファレンス』と併用して、API プログラミングの基本的な点を示します。実際の VB コード・ファイル一式は、このドキュメントの samples ディレクトリにもあります。

注： このサンプル・プログラムでは更新、レポート、および計算スクリプトが使用されます。デフォルト設定では、Essbase サーバーはこの種のスクリプトが接続先のデータベースのアプリケーション/データベース・ディレクトリにあると仮定しています。このサンプル・プログラムでは、これが $ARBORPATH/App/Sample/Basic ディレクトリです。サーバーが次にスクリプト・ファイルを検索するのが、プログラムが実行中のディレクトリです。標準的な Oracle Essbase のインストールには、calcdat.txt データ・ロード・ファイルがありますが、他のスクリプト・ファイルは$ARBORPATH/Docs/Api/Samples/vbexecs/V3Report から$ARBORPATH/App/Sample/Basic にコピーする必要があります。他の場所にファイルを配置することもできますが、その場合はプログラム内で絶対パス名を指定する必要があります。

フォーム内のコード

このコードはフォーム自体に付属しています。Code.bas(後続)の関数を呼び出します。Code.bas は他のプロジェクトに含めることもできます。

```
   Sub cmdStart_Click()
 Call Code.ESB_Init  '  Initializes ESB_INIT_T and calls EsbInit()
 Call ESB_Login    '  EsbLogin() sets server, user and password
 Call SetAfterLogin
End Sub

Sub cmdStop_Click()
 Call ESB_Logout    ' Should logout all login IDs (context handles)
 Call ESB_Term     ' EsbTerm() terminates the API
 Call lstMessagesClear
 Call SetBeforeStart
End Sub

Sub cmdClearMsg_Click()
 lstMessages.Clear   'Clear Messages
End Sub

Sub cmdCalcFile_Click()
 Call ESB_CalcFile  'Calculate
End Sub

Sub cmdClrData_Click()
 Call ESB_SetActive  'Set the active database before calling EsbClearDatabase()
 Call ESB_ClrData   'Clear data
End Sub

Sub cmdLdData_Click()
  MsgBox "WAIT!! Don't do anything until this process completes. Click OK and wait
about 15 seconds. "
 Call ESB_LdData    'Import Data
End Sub

Sub cmdQryFile_Click()
 Call ESB_QryFile
End Sub
```

```
Sub cmdQryStr_Click()
 Call ESB_QryStr
End Sub

Sub cmdQryStrs_Click()
'   Call QryStrs
 Call ESB_BeginReport  ' 1. EsbBeginReport()
 Call ESB_SendString  ' 2. EsbSendString() - for each string in the report spec
 Call ESB_EndReport   ' 3. EsbEndReport()

'*** Display returned data strings; assumes EsbBeginReport()'s ouput flag is TRUE
 If lngStatus = 0 Then  ' If EsbEndReport() succeeded, call EsbGetString()
  Call ESB_GetString  ' Server outputs data if intWhetherOutput = ESB_TRUE;
            '  ESB_GetString calls EsbGetString() to read the returned
            '  data until an empty string is returned
 End If

End Sub

Sub cmdUpdFile_Click()
 Call ESB_UpdFile
End Sub

Sub Form_Load()
 Call SetBeforeStart
End Sub

Sub SetBeforeStart()

'*** Enable cmdStart
 cmdStart.Enabled = True

'*** Disable everything else
 cmdStop.Enabled = False
 cmdClearMsg.Enabled = False
 lstMessages.Enabled = False

 cmdCalcFile.Enabled = False
 cmdClrData.Enabled = False
 cmdLdData.Enabled = False
 cmdQryStr.Enabled = False
 cmdQryStrs.Enabled = False
 cmdQryFile.Enabled = False
 cmdUpdFile.Enabled = False

End Sub

Sub SetAfterLogin()

'*** Disable cmdStart
 cmdStart.Enabled = False

'*** Enable everything else
 cmdStop.Enabled = True
 cmdClearMsg.Enabled = True
 lstMessages.Enabled = True
```

```
 cmdCalcFile.Enabled = True
 cmdClrData.Enabled = True
 cmdLdData.Enabled = True
 cmdQryStr.Enabled = True
 cmdQryStrs.Enabled = True
 cmdQryFile.Enabled = True
 cmdUpdFile.Enabled = True

End Sub
```

Code.bas モジュールのコード

このコードは、code.bas に含まれています。

```
Option Explicit

'*******************
'RETURN ERROR STATUS
'*******************
 Dim lngStatus As Long

'************
'INIT GLOBAL
'************
 Dim structInit  As ESB_INIT_T
 Dim lngInstHndl As Long
 Dim lngCtxHndl  As Long

'**********************
'ESB_GetMESSAGE GLOBAL
'**********************
 Dim intMsgLev  As Integer
 Dim lngMsgNmbr As Long


'****************************************
'ESB_SetACTIVE and ESB_ClearDATABASE GLOBAL
'****************************************
 Dim strActiveApp  As String
 Dim strActiveDb  As String

'*********************************************
'Init and turn error handle turned off
'*********************************************
Sub ESB_Init()

 ESB_TRUE = 1     ' ESB_TRUE
 ESB_FALSE = 0     ' and ESB_FALSE are variables, not constants

 '***********************
 ' Define init structure
 '***********************
 structInit.Version = ESB_API_VERSION
```

```
 structInit.MaxHandles = 10
 structInit.LocalPath = "$ARBORPATH"
 structInit.MessageFile = ""
 structInit.ClientError = ESB_TRUE
 structInit.ErrorStack = 100

 '******************
 'Initialize the API
 '******************
 lngStatus = EsbInit(structInit, lngInstHndl)

 '**************
 'Error Checking
 '**************
 If lngStatus = 0 Then
  MsgBox "The API is initialized: " & (lngInstHndl)
 Else
  MsgBox "The API failed to initialize: " & (lngStatus)
 End If

End Sub

'*******************************************************
'Login in user Admin. All login parameters are hardcoded
'*******************************************************
Sub ESB_Login()

 Dim strServer  As String * ESB_SVRNAMELEN
 Dim strUser   As String * ESB_USERNAMELEN
 Dim strPassword As String * ESB_PASSWORDLEN
 Dim intNumAppDb As Integer


 strServer = "Localhost"
 strUser = "Admin"
 strPassword = "password"


 lngStatus = EsbLogin(lngInstHndl, _
           strServer, strUser, strPassword, _
           intNumAppDb, _
           lngCtxHndl)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "Admin is logged in, with login ID (context handle) " & (lngCtxHndl) _
    & Chr$(10) & "WAIT! DO NOTHING!" _
    & Chr$(10) & "Retrieving login status; setting Sample/Basic as active"


  '*********************************************
  'Call the SetActive routine to select Sample Basic
  '*********************************************
  Call ESB_ListErrorStackMsgs  ' Even successful logins return useful messages
  Call ESB_SetActive
```

```
 Else
  MsgBox "Login failed: " & (lngStatus)
 End If

End Sub

'************************************************
'Sets the caller's active application and database.
'************************************************
Sub ESB_SetActive()

 Dim intUserAccess As Integer

 strActiveApp = "Sample"
 strActiveDb = "Basic"


 lngStatus = EsbSetActive(lngCtxHndl, strActiveApp, strActiveDb, intUserAccess)

'**************
'Error Checking
'**************

 If lngStatus = 0 Then
  MsgBox (strActiveApp) & "/" & (strActiveDb) & " is now active"

 Else
  MsgBox "EsbSetActive() failed: " & (lngStatus)
 End If

End Sub




'*******
' Logout
'*******
Sub ESB_Logout()

 lngStatus = EsbLogout(lngCtxHndl)

'*********************************************
'Display whether the logout succeeded or failed
'*********************************************
 If lngStatus = 0 Then
  MsgBox "Admin, with login ID (context handle) " & (lngCtxHndl) _
      & ", is logged out"
 Else
  MsgBox "EsbLogout() failed: " & (lngStatus)
 End If

End Sub

'****************************
```

```
' Terminate the VB API
'*****************************
Sub ESB_Term()


EsbTerm (lngInstHndl)

'*********************************
'Display whether the API terminated
'*********************************
 If lngStatus = 0 Then
  MsgBox "The API is terminated"
 Else
  MsgBox "EsbTerm() failed: " & (lngStatus)
 End If

End Sub

'*********************************************
'Gets a string of data from the active database.
'*********************************************
Sub ESB_GetString()

 Const intDStringLen = 256

 Dim strDataString As String * intDStringLen
 Dim intNumGSCalls As Integer

 intNumGSCalls = 1

 lngStatus = EsbGetString(lngCtxHndl, strDataString, intDStringLen)

'************************************************************
'Call EsbGetString() until an empty string (no data) is returned
'************************************************************
 Do While Mid$(strDataString, 1, 1) <> Chr$(0)

  If lngStatus = 0 Then
   MsgBox "EsbGetString() call #" & (intNumGSCalls) & " just read the string" _
   & Chr$(10) & (strDataString)    ' The server's translation of the query string
   lstMessages (strDataString)     ' Display each returned string on a line
   intNumGSCalls = intNumGSCalls + 1  ' Increment now often EsbGetString() is called
  Else
   MsgBox "EsbGetString() failed: " & (lngStatus)
  End If

  lngStatus = EsbGetString(lngCtxHndl, strDataString, intDStringLen)
 Loop

End Sub

'*********************************************************************
'EsbSendString() sends a string of data to the active database.
'This function should be called after EsbBeginReport(),EsbBeginUpdate(),
'or EsbBeginCalc()
'*********************************************************************
Sub ESB_SendString()
```

```
 Dim strQueryString     As String
 Dim arrQueryStrings(1 To 8) As String
 Dim intCounter       As Integer

 arrQueryStrings(1) = "<PAGE (Market, Measures) "
 arrQueryStrings(2) = "<COLUMN (Year, Scenario) "
 arrQueryStrings(3) = "<ROW (Product) "
 arrQueryStrings(4) = "<ICHILD Market "
 arrQueryStrings(5) = "Qtr1 Qtr2 "
 arrQueryStrings(6) = "Actual Budget Variance "
 arrQueryStrings(7) = "<ICHILD Product "
 arrQueryStrings(8) = "!"

'*****************************************************
'Send a series of query strings to the active database
'*****************************************************

 For intCounter = 1 To 8
  strQueryString = arrQueryStrings(intCounter)
  lngStatus = EsbSendString(lngCtxHndl, strQueryString)

'**************
'Error Checking
'**************
  If lngStatus = 0 Then
   MsgBox "EsbSendString() sent query string # " & (intCounter) _
      & " to the active database"
   lstMessages (strQueryString)
  Else
   MsgBox "EsbSendString() failed: " & (lngStatus)
  Exit Sub
  End If
 Next

End Sub

'*****************************************************************
'Sends a report specification to the active database from a file
'*****************************************************************
Sub ESB_QryFile()

 Dim lngDbCtxHndl   As Long
 Dim lngRFCtxHndl   As Long
 Dim strAppName    As String
 Dim strDbName    As String
 Dim strReportFile  As String
 Dim intWhetherOutput As Integer
 Dim intWhetherLock  As Integer

 lngDbCtxHndl = lngCtxHndl
 lngRFCtxHndl = lngCtxHndl
 strAppName = "Sample"
 strDbName = "Basic"
 strReportFile = "MyRpt01"
```

```
 intWhetherOutput = ESB_TRUE   ' If TRUE, data is output from server
 intWhetherLock = ESB_FALSE    ' If TRUE, blocks are locked for update
                ' If both are FALSE, report spec checked for syntax


 lngStatus = EsbReportFile(lngDbCtxHndl, lngRFCtxHndl, strAppName, strDbName, _
             strReportFile, intWhetherOutput, intWhetherLock)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "The report file" & Chr$(10) & (strReportFile) & Chr$(10) _
     & "was sent to " & (strAppName) & (strDbName) & Chr$(10) _
     & "EsbGetString() will read the data"

 '*************************************************************************
 'Calls EsbGetString to read the returned data until an empty string is returned
 '*************************************************************************
 Call ESB_GetString

 Else
  MsgBox "EsbReportFile() failed: " & (lngStatus)
 End If
End Sub

'*********************************************************************
'Sends a report specification to the active database as a single string
'*********************************************************************
Sub ESB_QryStr()

 Dim intWhetherOutput As Integer
 Dim intWhetherLock  As Integer
 Dim strQueryString  As String

 strQueryString = "<DESC Year !"  ' One query string
 intWhetherOutput = ESB_TRUE    ' If TRUE, data is output from server
 intWhetherLock = ESB_FALSE    ' If TRUE, blocks are locked for update
                ' If both are FALSE, report spec checked for syntax


 lngStatus = EsbReport(lngCtxHndl, intWhetherOutput, intWhetherLock, strQueryString)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "The report specification" & Chr$(10) & (strQueryString) & Chr$(10) _
     & "was sent to the active database" & Chr$(10) _
     & "EsbGetString() will read the data"

  '*****************************************************
  ' Server outputs data if intWhetherOutput = ESB_TRUE;
  ' ESB_GetString calls EsbGetString() to read the returned
  ' data until an empty string is returned
  '*****************************************************
  Call ESB_GetString
```

```
 Else
  MsgBox "EsbReport() failed: " & (lngStatus)
 End If

End Sub

'*********************************************************************
'Sends an update specification to the active database from a file
'*********************************************************************
Sub ESB_UpdFile()

 Dim lngDbCtxHndl   As Long
 Dim lngUFCtxHndl   As Long
 Dim strAppName    As String
 Dim strDbName    As String
 Dim strUpdateFile  As String
 Dim intWhetherStore  As Integer
 Dim intWhetherUnlock As Integer

 lngDbCtxHndl = lngCtxHndl
 lngUFCtxHndl = lngCtxHndl
 strAppName = "Sample"
 strDbName = "Basic"
 strUpdateFile = "CDupdtDb"


 intWhetherStore = ESB_TRUE   ' Database is updated & data is stored (on server)
 intWhetherUnlock = ESB_TRUE  ' Locked blocks are unlocked after data is updated

'****************************************
'Lock database blocks before you update them
'****************************************
 Call ESB_LockDatabase


'****************************************
'Send update file to the specified database
'****************************************
 lngStatus = EsbUpdateFile(lngDbCtxHndl, lngUFCtxHndl, strAppName, strDbName, _
             strUpdateFile, intWhetherStore, intWhetherUnlock)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "The update file" & Chr$(10) & (strUpdateFile) & Chr$(10) _
     & "was sent to " & (strAppName) & (strDbName)
 Else
  MsgBox "EsbUpdateFile() failed: " & (lngStatus)
 End If

 '*******************************
 'Calls error checking sub routine
 '*******************************
 Call ESB_ListErrorStackMsgs

End Sub
```

```
'************************************************************
'Starts sending a report specification to the active database
'************************************************************
Sub ESB_BeginReport()

 Dim intWhetherOutput As Integer
 Dim intWhetherLock  As Integer
 Dim strQueryString  As String

 intWhetherOutput = ESB_TRUE  ' If TRUE, data is output from server
 intWhetherLock = ESB_FALSE  ' If TRUE, blocks are locked for update
              ' If both are FALSE, report spec checked for syntax


 lngStatus = EsbBeginReport(lngCtxHndl, intWhetherOutput, intWhetherLock)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "EsbBeginReport() succeeded"
 Else
  MsgBox "EsbBeginReport() failed: " & (lngStatus)
 End If

End Sub

'************************************************************************
'EsbEndReport marks the end of the report specification sent to the
'active database.
'************************************************************************

Sub ESB_EndReport()

 lngStatus = EsbEndReport(lngCtxHndl)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "EsbEndReport() succeeded"

 Else
  MsgBox "EsbEndReport() failed: " & (lngStatus)

  '*******************************
  'Calls error checking sub routine
  '*******************************
  Call ESB_ListErrorStackMsgs

  Exit Sub

 End If

End Sub
```

```
'*************************************************************
'Executes a calc script against the active database from a file
'*************************************************************

Sub ESB_CalcFile()

 Dim lngDbCtxHndl    As Long
 Dim lngCSCtxHndl    As Long
 Dim strAppName     As String
 Dim strDbName     As String
 Dim strCalcScriptFile As String
 Dim intWhetherCalc   As Integer  ' If TRUE, the calc script is executed

 lngDbCtxHndl = lngCtxHndl
 lngCSCtxHndl = lngCtxHndl
 strAppName = "Sample"
 strDbName = "Basic"
 strCalcScriptFile = "Calc5Dim"
 intWhetherCalc = ESB_TRUE


 lngStatus = EsbCalcFile(lngDbCtxHndl, lngCSCtxHndl, strAppName, strDbName, _
           strCalcScriptFile, intWhetherCalc)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox (strAppName) & (strDbName) & " is being calculated" & Chr$(10) _
    & "using the calc script in " & (strCalcScriptFile)

  '*******************************************************
  'Call Esb_GetProcessState to get the current state of calc
  '*******************************************************
  Call ESB_GetProcessState

 Else
  MsgBox "EsbCalcFile() failed: " & (lngStatus)

   '*******************************
   'Calls error checking sub routine
   '*******************************
   Call ESB_ListErrorStackMsgs
 End If
End Sub

'*********************************
'Clear data from the active database
'*********************************
Sub ESB_ClrData()

 lngStatus = EsbClearDatabase(lngCtxHndl)

'********************
'Begin error checking
'********************
 If lngStatus = 0 Then
```

```
  MsgBox "WAIT!! Data is being cleared from " & (strActiveApp) & (strActiveDb)


  '************************************************************
  'Call Esb_GetProcessState to get the current state of process
  '************************************************************
  Call ESB_GetProcessState

 Else
  MsgBox "EsbClearDatabase() failed: " & (lngStatus)

  '*******************************
  'Calls error checking sub routine
  '*******************************
  Call ESB_ListErrorStackMsgs
 End If
End Sub

'************************************************
'Import data from different sources
'************************************************
Sub ESB_LdData()

 Dim structRulesFile    As ESB_OBJDEF_T
 Dim structDataFile     As ESB_OBJDEF_T
 Dim structSQLSource    As ESB_MBRUSER_T

 Dim strErrorsOnLoadFile  As String
 Dim intWhetherAbortOnError As Integer

 structDataFile.hCtx = lngCtxHndl
 structDataFile.Type = ESB_OBJTYPE_TEXT
 structDataFile.AppName = "Sample"
 structDataFile.DbName = "Basic"
 structDataFile.FileName = "CalcDat"

 strErrorsOnLoadFile = "ErrsOnLd.txt"
 intWhetherAbortOnError = ESB_TRUE

'****************************************
'Import data from CalcDat.txt to Sample/Basic
'****************************************
 lngStatus = EsbImport(lngCtxHndl, structRulesFile, structDataFile, structSQLSource,
_
          strErrorsOnLoadFile, intWhetherAbortOnError)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "WAIT!! Data from " & (structDataFile.FileName) & Chr$(10) _
    & "is being imported to " & (structDataFile.AppName) & (structDataFile.DbName)

  '************************************************************
  'Call Esb_GetProcessState to get the current state of import
  '************************************************************
  Call ESB_GetProcessState
```

```
 Else
  MsgBox "EsbImport() failed: " & (lngStatus)

  '*******************************
  'Calls error checking sub routine
  '*******************************
  Call ESB_ListErrorStackMsgs
 End If
End Sub

'*********************************************************************
' ESB_LockDatabase() calls EsbReportFile() to lock blocks for update
'*********************************************************************
Sub ESB_LockDatabase()


 Dim lngDbCtxHndl   As Long
 Dim lngRFCtxHndl   As Long
 Dim strAppName    As String
 Dim strDbName    As String
 Dim strReportFile  As String
 Dim intWhetherOutput As Integer  ' If TRUE, data is output from server
 Dim intWhetherLock  As Integer  ' If TRUE, blocks are locked for update

 lngDbCtxHndl = lngCtxHndl
 lngRFCtxHndl = lngCtxHndl
 strAppName = "Sample"
 strDbName = "Basic"
 strReportFile = "CDlockDb"


 intWhetherOutput = ESB_FALSE   ' FALSE: no data is output from server
 intWhetherLock = ESB_TRUE    ' TRUE: blocks are locked for update


 lngStatus = EsbReportFile(lngDbCtxHndl, lngRFCtxHndl, strAppName, strDbName, _
             strReportFile, intWhetherOutput, intWhetherLock)

'**************
'Error Checking
'**************
 If lngStatus = 0 Then
  MsgBox "The report file" & Chr$(10) & (strReportFile) & Chr$(10) _
     & "was sent to " & (strAppName) & (strDbName) & Chr$(10) _
     & "Blocks are locked for update" & Chr$(10) _
     & "EsbUpdateFile() will update the CalcData database"
 Else
  MsgBox "EsbReportFile() failed: " & (lngStatus)

  '*******************************
  'Calls error checking sub routine
  '*******************************
  Call ESB_ListErrorStackMsgs
 End If
End Sub
```

```
'****************************************************************
'Get the current state of an asynchronous process until it finishes
'****************************************************************
Sub ESB_GetProcessState()

 Dim structProcessState As ESB_PROCSTATE_T

 lngStatus = EsbGetProcessState(lngCtxHndl, structProcessState)
 Do Until structProcessState.State = ESB_STATE_DONE
  lngStatus = EsbGetProcessState(lngCtxHndl, structProcessState)
 Loop

 MsgBox "Asynchronous Process Completed"

End Sub

'************************************************************
'This is an error checking subroutine that uses EsbGetMessage
'************************************************************
Sub ESB_ListErrorStackMsgs()

 Const intMsgLen = 256
 Dim strMsg As String * intMsgLen


 lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, _
 strMsg, intMsgLen)

 Dim intStackNmbr As Integer

 intStackNmbr = 1

'********************************************************************
'Do while the error stack has messages and drops messages in a ListBox
'********************************************************************
 Do While Mid$(strMsg, 1, 1) <> Chr$(0)
  lstMessages "MESSAGE ON ERROR STACK:"
  lstMessages "Stack #" & (intStackNmbr)
  lstMessages "Level #" & (intMsgLev)
  lstMessages "Message #" & (lngMsgNmbr)
  lstMessages (strMsg)
  intStackNmbr = intStackNmbr + 1
  lngStatus = EsbGetMessage(lngInstHndl, intMsgLev, lngMsgNmbr, strMsg, intMsgLen)
 Loop

End Sub

Sub lstMessages(strItem As String)
  frmRprts.lstMessages.AddItem (strItem)
End Sub

Sub lstMessagesClear()
  frmRprts.lstMessages.Clear
End Sub
```

B

Shared Servicesの移行とユーザー管理のAPIの例

```
   /*
Declaration of Include files
*/

#if defined _WIN32 || defined _WINDOWS
#include
#endif

#include
#include
#include
#pragma pack (1)
#include
#include
#pragma pack ()

/
*****************************************************************************
**/
/*--------------------------- Example Usage Starts Here
-------------------------------*/
/
*****************************************************************************
**/

/*
ESS_FUNC_M EssSetSSSecurityMode(ESS_HCTX_T  hCtx,
               ESS_USHORT_T  Option,
               ESS_STR_T   Password);
*/
ESS_FUNC_M ESS_SS_SetSSSecurityMode(ESS_HCTX_T  hCtx)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     newpassword = ESS_NULL;
 ESS_USHORT_T    option;

 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/

 option = 1; /* Using user name as password */
```

```
 sts = EssSetSSSecurityMode(hCtx, option, newpassword);

 if(sts)
  printf("Failed to migrate Analytic Services Server to Shared Services mode.
\n");

 return (sts);
}

/*
ESS_FUNC_M EssGetEssbaseSecurityMode (ESS_HCTX_T hCtx,
                 ESS_PSECURITY_MODE_T mode);
*/
ESS_FUNC_M ESS_SS_GetEssbaseSecurityMode(ESS_HCTX_T  hCtx)
{
 ESS_STS_T      sts = ESS_STS_NOERR;
 ESS_SECURITY_MODE_T mode;

 sts = EssGetEssbaseSecurityMode(hCtx, &mode);

 if(sts)
 {
  printf("Failed to get Essbase Security mode.\n");
 }
 else
 {
  printf("Essbase Security Mode      : %d\n", mode);
 }
 return(sts);
}

/*
ESS_FUNC_M EssListSSMigrFailedUsers(ESS_HCTX_T,
                ESS_PUSHORT_T,
                ESS_PPUSERNAME_T);
*/
ESS_FUNC_M ESS_SS_ListSSMigrFailedUsers(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_PUSERNAME_T  pNativeUserList = NULL;
 ESS_USHORT_T   Count = 0,
           index;

 sts = EssListSSMigrFailedUsers(hCtx, &Count, &pNativeUserList);

 if (!sts)
 {
  if (Count && pNativeUserList)
  {
   printf ("\n------- User List -------\n\n");

   for (index = 0; index < Count; index++)
   {
    if (pNativeUserList[index])
     printf ("%s\n", pNativeUserList[index]);
   }
```

```
    EssFree(hInst, pNativeUserList);
   }
   else
    printf("\nUser list is empty\n\n");
  }
  else
   printf("Failed to get Shared Services migration failed Users list.\n");

  return (sts);
}

/*
ESS_FUNC_M EssListSSMigrFailedGroups(ESS_HCTX_T,
                ESS_PUSHORT_T,
                ESS_PPUSERNAME_T);
*/
ESS_FUNC_M ESS_SS_ListSSMigrFailedGroups(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
  ESS_STS_T     sts = ESS_STS_NOERR;
  ESS_PUSERNAME_T  pNativeUserList = NULL;
  ESS_USHORT_T   Count = 0,
           index;

  sts = EssListSSMigrFailedGroups(hCtx, &Count, &pNativeUserList);

  if (!sts)
  {
   if (Count && pNativeUserList)
   {
    printf ("\n------- Group List -------\n\n");

    for (index = 0; index < Count; index++)
    {
     if (pNativeUserList[index])
      printf ("%s\n", pNativeUserList[index]);
    }

    EssFree(hInst, pNativeUserList);
   }
   else
    printf("\nGroup list is empty\n\n");
  }
  else
   printf("Failed to get Shared Services migration failed Groups list.\n");

  return (sts);
}

/*
ESS_FUNC_M EssSetUserToSS (ESS_HCTX_T  hCtx,
            ESS_STR_T   UserName,
            ESS_USHORT_T  Option,
            ESS_STR_T   Password);
*/
ESS_FUNC_M ESS_SS_SetUserToSS(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
  ESS_STS_T     sts = ESS_STS_NOERR;
```

```
 ESS_USHORT_T   option;
 ESS_STR_T     userName = ESS_NULL;
 ESS_STR_T     newpassword = ESS_NULL;

 sts = EssAlloc(hInst, sizeof(ESS_USERNAME_T), &userName);
 if(sts)
  return (sts);
 memset(userName, 0, sizeof(ESS_USERNAME_T));
 strcpy( userName, "essexer");

 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/

 option = 1; /* Using user name as password */

 sts = EssSetUserToSS(hCtx, userName, option, newpassword);

 if(sts)
  printf("Failed to migrate User %s to Shared Services mode.\n", userName);

 if (userName)
  EssFree(hInst, userName);

 return (sts);
}

/*
ESS_FUNC_M EssSetGroupToSS (ESS_HCTX_T  hCtx,
            ESS_STR_T   GroupName);
*/
ESS_FUNC_M ESS_SS_SetGroupToSS(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     groupName = ESS_NULL;

 sts = EssAlloc(hInst, sizeof(ESS_USERNAME_T), &groupName);
 if(sts)
  return (sts);
 memset(groupName, 0, sizeof(ESS_USERNAME_T));
 strcpy( groupName, "essgrp");

 sts = EssSetGroupToSS(hCtx, groupName);

 if(sts)
  printf("Failed to migrate Group %s to Shared Services mode.\n", groupName);

 if (groupName)
  EssFree(hInst, groupName);

 return (sts);
}

/*
```

```
ESS_FUNC_M EssSetUsersToSS(ESS_HCTX_T  hCtx,
            ESS_USHORT_T Option,
            ESS_STR_T   Password);
*/
ESS_FUNC_M ESS_SS_SetUsersToSS(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_USHORT_T   option;
 ESS_STR_T     newpassword = ESS_NULL;

 /* New Shared Services Native User Password Option:
  *
  * 0 to use user provided password
  * 1 to use the user name as password
  * 2 to automatically generate a password
  **/

 option = 0; /* Using user provided password */

 sts = EssAlloc(hInst, sizeof(ESS_PASSWORD_T), &newpassword);
 if(sts)
  return (sts);
 memset(newpassword, 0, sizeof(ESS_PASSWORD_T));
 strcpy( newpassword, "password");

 sts = EssSetUsersToSS(hCtx, option, newpassword);

 if(sts)
  printf("Failed to migrate Users to Shared Services mode.\n");

 if (newpassword)
  EssFree(hInst, newpassword);

 return (sts);
}

/*
ESS_FUNC_M EssSetGroupsToSS(ESS_HCTX_T  hCtx);
*/
ESS_FUNC_M ESS_SS_SetGroupsToSS(ESS_HCTX_T hCtx)
{
 ESS_STS_T     sts = ESS_STS_NOERR;

 sts = EssSetGroupsToSS(hCtx);

 if(sts)
  printf("Failed to migrate Groups to Shared Services mode.\n");

 return (sts);
}

/*
ESS_FUNC_M EssSetEasLocation (ESS_HCTX_T hCtx,
             ESS_STR_T EasLocation);
*/
ESS_FUNC_M ESS_SS_SetEasLocation(ESS_HCTX_T hCtx, ESS_HINST_T hInst)
{
```

```
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_STR_T     easLoc = ESS_NULL;

 /* Eas Location */
 sts = EssAlloc(hInst, sizeof(ESS_PATHLEN), &easLoc);
 if(sts)
  return (sts);
 memset(easLoc, 0, sizeof(ESS_PATHLEN));
 strcpy( easLoc, "localhost:10080");

 sts = EssSetEasLocation(hCtx, easLoc);

 if (sts)
  printf("Failed to set EAS Location.\n");

 if (easLoc)
  EssFree(hInst, easLoc);

 return (sts);
}

/*
ESS_FUNC_M EssReRegisterApplication (ESS_HCTX_T hCtx,
                ESS_STR_T AppName,
                ESS_BOOL_T AllApps);
*/
ESS_FUNC_M ESS_SS_ReRegisterApplication(ESS_HCTX_T  hCtx, ESS_HINST_T hInst)
{
 ESS_STS_T     sts = ESS_STS_NOERR;
 ESS_BOOL_T    allApps;
 ESS_STR_T     appName = ESS_NULL;

 sts = EssAlloc(hInst, sizeof(ESS_APPNAME_T), &appName);
 if(sts)
  return (sts);
 memset(appName, 0, sizeof(ESS_APPNAME_T));
 strcpy( appName, "Sample");


 /* Do you want All applications re-registered?
  * Enter ESS_TRUE for Yes
  *    ESS_FALSE for No
  **/
 allApps = ESS_FALSE; /* Re-registering only 1 application */

 sts = EssReRegisterApplication(hCtx, appName, allApps);

 if (sts)
  printf("Failed to Re-register Application %s.\n", appName);

 if (appName)
  EssFree(hInst, appName);

 return (sts);
}

/
```

```
*****************************************************************************
**/
/*-------------------------- Example Usage Starts Here
--------------------------------*/
/
*****************************************************************************
**/

/
*****************************************************************************
**/
/*---------------------------------- Testing API
---------------------------------------*/
/
*****************************************************************************
**/

/*
Declaration of handles and connection information variables
*/


/*****************************************************/
/*************** MAIN FUNCTION ***********************/
/*****************************************************/

main()
{
 ESS_HINST_T   hInst;
 ESS_HCTX_T    hCtx;
 ESS_SVRNAME_T  srvrName  =  "localhost";
 ESS_USERNAME_T  userName  =  "essexer";
 ESS_PASSWORD_T  pswd    =  "password";
 ESS_STS_T    sts     =  ESS_STS_NOERR;
 ESS_USHORT_T   Items;
 ESS_PAPPDB_T   pAppsDbs  =  ESS_NULL;

 ESS_INIT_T InitStruct = {ESS_API_VERSION,
            ESS_NULL,
            0L,
            255,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            ESS_NULL,
            0L
           };

 sts = EssInit(&InitStruct, &hlnst);
 if (sts)
 {
  printf("EssInit failure: %ld\n", sts);
  exit ((int) sts);
 }
```

```
 sts = EssLogin(hInst, srvrName, userName, pswd, &Items, &pAppsDbs, &hCtx);
 if (sts)
 {
  printf("EssLogin failure: %ld\n", sts);
  exit ((int) sts);
 }

 sts = ESS_SS_SetSSSecurityMode(hCtx);
 if (sts)
  printf("ESS_SS_SetSSSecurityMode failed: %ld\n", sts);

 sts = ESS_SS_GetEssbaseSecurityMode(hCtx);
 if (sts)
  printf("ESS_SS_GetEssbaseSecurityMode failed: %ld\n", sts);

 sts = ESS_SS_ListSSMigrFailedUsers(hCtx, hInst);
 if (sts)
  printf("ESS_SS_ListSSMigrFailedUsers failed: %ld\n", sts);

 sts = ESS_SS_ListSSMigrFailedGroups(hCtx, hInst);
 if (sts)
  printf("ESS_SS_ListSSMigrFailedGroups failed: %ld\n", sts);

 sts = ESS_SS_SetUserToSS(hCtx, hInst);
 if (sts)
  printf("ESS_SS_SetUserToSS failed: %ld\n", sts);

 sts = ESS_SS_SetGroupToSS(hCtx, hInst);
 if (sts)
  printf("ESS_SS_SetGroupToSS failed: %ld\n", sts);

 sts = ESS_SS_SetUsersToSS(hCtx, hInst);
 if (sts)
  printf("ESS_SS_SetUsersToSS failed: %ld\n", sts);

 sts = ESS_SS_SetGroupsToSS(hCtx);
 if (sts)
  printf("ESS_SS_SetGroupsToSS failed: %ld\n", sts);

 sts = ESS_SS_SetEasLocation(hCtx, hInst);
 if (sts)
  printf("ESS_SS_SetEasLocation failed: %ld\n", sts);

 sts = ESS_SS_ReRegisterApplication(hCtx, hInst);
 if (sts)
  printf("ESS_SS_ReRegisterApplication failed: %ld\n", sts);

 sts = EssLogout(hCtx);
 sts = EssTerm(hInst);
}
```

C

制限

この付録の内容

名前の制限

サブトピック

- Essbase サーバー(ホスト)名の制限
- アプリケーション名の制限
- データベース名の制限
- フィルタ名の制限
- グループ名の制限
- オブジェクト名の制限
- パスワードの制限
- ユーザー名の制限

Essbase サーバー(ホスト)名の制限

- 非 Unicode アプリケーションの制限: 1024 バイト
- Unicode モード・アプリケーションの制限: 1024 文字

アプリケーション名の制限

- 非 Unicode アプリケーションの制限: 8 バイト
- Unicode モード・アプリケーションの制限: 30 文字

アプリケーション名には、DOS ファイル名で許可されたすべての特殊文字を使用できます。スペース、カンマ、タブ、スラッシュ、円記号(バックスラッシュ)、ピリオドは使用できません。一部の特殊文字(たとえば、@、$、%、&)は、オペレーティング・システムでよく使用されるため、使用をお薦めしません。

データベース名の制限

- 非 Unicode アプリケーションの制限: 8 バイト
- Unicode モード・アプリケーションの制限: 30 文字

データベース名には、DOS ファイル名で許可されたすべての特殊文字を使用できます。スペース、カンマ、タブ、スラッシュ、円記号(バックスラッシュ)、ピリオドは使用できません。一部の特殊文字(たとえば、@、$、%、&)は、オペレーティング・システムでよく使用されるため、使用をお薦めしません。

フィルタ名の制限

- 非 Unicode アプリケーションの制限: 256 バイト
- Unicode モード・アプリケーションの制限: 256 文字

グループ名の制限

- 非 Unicode アプリケーションの制限: 256 バイト
- Unicode モード・アプリケーションの制限: 256 文字

オブジェクト名の制限

- 非 Unicode アプリケーションの制限: 8 バイト
- Unicode モード・アプリケーションの制限: 30 文字

オブジェクト名には、DOS ファイル名で許可されたすべての特殊文字を使用できます。スペース、カンマ、円記号(バックスラッシュ)、ピリオドは使用できません。

パスワードの制限

- 非 Unicode アプリケーションの制限: 100 バイト
- Unicode モード・アプリケーションの制限: 100 文字

ユーザー名の制限

- 非 Unicode アプリケーションの制限: 256 バイト
- Unicode モード・アプリケーションの制限: 256 文字

ユーザー名では、大文字と小文字は区別されません。また、円記号(バックスラッシュ)(\)は使用できません。

ドリルスルー URL の制限

ドリルスルー URL には次の制限が適用されます:

- データベース当たりのドリルスルー URL の数は、255 に制限されています。
- ドリルスルー URL のドリル可能領域の数は、256 に制限されています。

- ドリル可能領域当たりの文字数は、65536 に制限されています。

索引

記号

A - Z

あ行

か行

さ行

た行

な行

は行

ま行

や行

ら行

わ行