SQL Anywhere - Ultra Light
文書バージョン: 17 – 2016-05-11

Ultra Light Embedded SQL API リファレンス

目次

1 Ultra Light Embedded SQL API リファレンス

Embedded SQL アプリケーションは複数の Ultra Light 関数をサポートします。

この章で説明する関数のプロトタイプは、EXEC SQL INCLUDE SQLCA コマンドを使用してインクルードします。

ヘッダファイル

- `mlfiletransfer.h`
- `ulprotos.h`

> **注記**
>
> 主な **SQL Anywhere** マニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows のスタートメニューを使用してアクセスするか (Microsoft Windows)、`C:¥Program Files¥SQL Anywhere 17¥Documentation` にナビゲートします。
>
> また、DocCommentXchange の Web で、主な SQL Anywhere API リファレンスマニュアルにアクセスすることもできます。http://dcx.sap.com

このセクションの内容:

メッセージが ActiveSync に対する Mobile Link プロバイダからの同期メッセージであるかどうかを確認し、そのメッセージを処理するコードを呼び出すことができます。

ULLibraryVersion(void) メソッド [42 ページ]
Ultra Light ランタイムライブラリのバージョン番号を返します。

ULRegisterForEvent(SQLCA *, char const *, char const *, char const *, ul_bool) メソッド [43 ページ]
イベントの通知を受け取るためのキューを登録または登録解除します。

ULResetLastDownloadTime(SQLCA *, char const *) メソッド [44 ページ]
アプリケーションが以前にダウンロードされたデータを再同期するように、パブリケーションの最終ダウンロード時間をリセットします。

ULRevokeConnectFrom(SQLCA *, char const *) メソッド [45 ページ]
Ultra Light データベースからユーザ ID のアクセス権を取り消します。

ULRollbackPartialDownload(SQLCA *) メソッド [45 ページ]
失敗した同期からの変更をロールバックします。

ULRSALibraryVersion(void) メソッド [46 ページ]
RSA 暗号化ライブラリのバージョン番号を返します。

ULSendNotification(SQLCA *, char const *, char const *, char const *) メソッド [46 ページ]
指定された名前と一致するすべてのキューに通知を送信します。

ULSetDatabaseID(SQLCA *, ul_u_long) メソッド [47 ページ]
データベースの ID 番号を設定します。

ULSetDatabaseOptionString(SQLCA *, ul_database_option_id, char const *) メソッド [48 ページ]
文字列値からデータベースオプションを設定します。

ULSetDatabaseOptionULong(SQLCA *, ul_database_option_id, ul_u_long) メソッド [48 ページ]
数値のデータベースオプションを設定します。

ULSetErrorCallback(SQLCA *, ul_error_callback_fn_a, ul_void *, char *, size_t) メソッド [49 ページ]
エラーの発生時に呼び出されるようコールバックを設定します。

ULSetSynchronizationCallback(SQLCA *, ul_sync_observer_fn, ul_void *) メソッド [49 ページ]
同期の実行中に呼び出されるようコールバックを設定します。

ULSetSyncInfo(SQLCA *, char const *, ul_sync_info *) メソッド [50 ページ]
指定された ul_sync_info 構造体に基づいて、指定された名前を使用して同期プロファイルを作成します。

ULSignalSyncIsComplete() メソッド [51 ページ]
同期メッセージの処理が完了したことを示します。

ULStartSynchronizationDelete(SQLCA *) メソッド [51 ページ]
この接続の START SYNCHRONIZATION DELETE を設定します。

ULStaticFini() メソッド [51 ページ]
Embedded SQL アプリケーションで Ultra Light ランタイムのファイナライズを実行します。

ULStaticInit() メソッド [52 ページ]
Embedded SQL アプリケーションで Ultra Light ランタイムの初期化を実行します。

ULStopSynchronizationDelete(SQLCA *) メソッド [52 ページ]
この接続の STOP SYNCHRONIZATION DELETE を設定します。

ULSynchronize(SQLCA *, ul_sync_info *) メソッド [53 ページ]

Ultra Light アプリケーションで同期を開始します。

ULSynchronizeFromProfile(SQLCA *, char const *, char const *, ul_sync_observer_fn, ul_void *) メソッド [53 ページ]
指定されたプロファイルとマージパラメータを使用して、データベースを同期します。

ULTriggerEvent(SQLCA *, char const *, char const *) メソッド [54 ページ]
ユーザ定義のイベントをトリガして、登録されたすべてのキューに通知を送信します。

ULTruncateTable(SQLCA *, ul_table_num) メソッド [55 ページ]
テーブルをトランケートし、STOP SYNCHRONIZATION DELETE 文を一時的にアクティブにします。

ULValidateDatabaseLowLevelA(SQLCA *, char const *, ul_u_short, ul_validate_callback_fn, void *) メソッド [56 ページ]
指定されたデータベースの低レベル構造を検証します。

ULValidateDatabaseConnectedA(SQLCA *, char const *, ul_u_short, ul_validate_callback_fn, void *) メソッド [57 ページ]
現在の接続でのデータベースを検証します。

ul_database_option_id 列挙体 [58 ページ]
ユーザが設定できる可能なデータベースオプションを設定します。

ul_database_property_id 列挙体 [59 ページ]
ユーザが取得できる可能なデータベースプロパティを設定します。

ml_file_transfer_info 構造体 [61 ページ]
ファイルのアップロードまたはダウンロードのパラメータが格納された構造体。

ml_file_transfer_status 構造体 [63 ページ]
ファイルのアップロードまたはダウンロードの進行中の、ステータスまたは進行状況情報が格納された構造体。

mlft_stream_error 構造体 [64 ページ]
ファイルのアップロードまたはダウンロードの進行中の、ステータスまたは進行状況情報が格納された構造体。

MLFT_STATUS_FLAG_IS_BLOCKING 変数 [65 ページ]
ml_file_transfer_status.flags フィールドでビット配列を定義して、Mobile Link サーバからの応答を待機中、ファイル転送はブロックされていることを示します。

1.1 db_fini メソッド

Ultra Light ランタイムライブラリで使用したリソースを解放します。

構文

```
unsigned short db_fini( SQLCA * sqlca );
```

戻り値

- 処理中にエラーが発生した場合は 0。エラー情報は SQLCA に設定されます。
- エラーが発生しなかった場合は 0 以外。

備考

db_fini が呼び出された後に、他の Ultra Light ライブラリ呼び出しをしたり、Embedded SQL コマンドを実行したりしないでください。

使用する SQLCA ごとに 1 回ずつ db_fini を呼び出します。

1.2 db_init メソッド

Ultra Light ランタイムライブラリを初期化します。

構文

```
unsigned short db_init( SQLCA * sqlca );
```

戻り値

- 処理中にエラーが発生した場合は 0 (たとえば永続ストアの初期化時)。エラー情報は SQLCA に設定されます。
- エラーが発生しなかった場合は 0 以外。Embedded SQL コマンドと関数の使用を開始できます。

備考

この関数は、他の Ultra Light ライブラリを呼び出す前、および Embedded SQL コマンドを実行する前に呼び出す必要があります。

通常は、この関数を 1 回だけ呼び出して、ヘッダファイル `sqlca.h` に定義されているグローバル変数 sqlca のアドレスを渡してください。アプリケーションに複数の実行パスがある場合、複数の db_init を呼び出すことができます。ただし、それぞれに別個の sqlca ポインタが必要です。この別個の SQLCA ポインタには、ユーザが定義したものを使用するか、db_fini で解放されたグローバル SQLCA を使用します。

マルチスレッドアプリケーションでは、各スレッドは、別個の SQLCA を獲得するために db_init を呼び出します。同じ SQLCA を使用する接続とトランザクションは、1 つのスレッドで続けて実行してください。

SQLCA を初期化すると、その前の ULEnable 関数呼び出しによる設定がすべてリセットされます。SQLCA を再初期化する場合は、アプリケーションで必要な ULEnable 関数をすべて発行する必要があります。

1.3 MLFileDownload(ml_file_transfer_info *) メソッド

Mobile Link インタフェースを使用して、Mobile Link サーバからファイルをダウンロードします。

構文

```
public bool MLFileDownload (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

備考

転送対象ファイルのソースロケーションを設定する必要があります。このロケーションは、Mobile Link サーバ上の Mobile Link ユーザフォルダ (または Mobile Link サーバ上のデフォルトフォルダ) を指定する必要があります。また、ファイルのターゲットロケーションとファイル名を設定することもできます。

たとえば、新しいデータベースまたは置き換えるデータベースを Mobile Link サーバからダウンロードするようにアプリケーションをプログラミングできます。検索される最初のロケーションは各ユーザのサブフォルダなので、ユーザごとにファイルをカスタマイズできます。サーバのルートフォルダは、指定ファイルがユーザのフォルダになかった場合に検索されるロケーションなので、デフォルトの転送ファイルは、サーバのルートフォルダに配置することもできます。

例

次の例に、MLFileDownload メソッドの使用方法を示します。

```
ml_file_transfer_info info;
MLInitFileTransferInfo( &info );
MLFTEnableZlibCompression( &info );
info.filename = "myfile";
info.username = "user1";
info.password = "pwd";
info.version = "ver1";
info.stream = "HTTP";
info.stream_parms = "host=myhost.com;compression=zlib";
if( ! MLFileDownload( &info ) ) {
       // file download failed
}
MLFiniFileTransferInfo( &info );
```

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.4 MLFileUpload(ml_file_transfer_info *) メソッド

Mobile Link インタフェースを使用して、Mobile Link サーバからファイルをアップロードします。

構文

```
public bool MLFileUpload (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

備考

転送対象ファイルのソースロケーションを設定する必要があります。このロケーションは、Mobile Link サーバ上の Mobile Link ユーザフォルダ (または Mobile Link サーバ上のデフォルトフォルダ) を指定する必要があります。また、ファイルのターゲットロケーションとファイル名を設定することもできます。

たとえば、新しいデータベースまたは置き換えるデータベースを Mobile Link サーバからアップロードするようにアプリケーションをプログラミングできます。検索される最初のロケーションは各ユーザのサブフォルダなので、ユーザごとにファイルをカスタマイズできます。サーバのルートフォルダは、指定ファイルがユーザのフォルダになかった場合に検索されるロケーションなので、デフォルトの転送ファイルは、サーバのルートフォルダに配置することもできます。

例

次の例に、MLFileUpload メソッドの使用方法を示します。

```
ml_file_transfer_info info;
MLInitFileTransferInfo( &info );
MLFTEnableZlibCompression( &info );
info.filename = "myfile";
info.username = "user1";
info.password = "pwd";
info.version = "ver1";
info.stream = "HTTP";
info.stream_parms = "host=myhost.com;compression=zlib";
if( ! MLFileUpload( &info ) ) {
          // file upload failed
}
```

```
MLFiniFileTransferInfo( &info );
```

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.5 MLFiniFileTransferInfo(ml_file_transfer_info *) メソッド

ml_file_transfer_info 構造体が初期化されている場合、この構造体で割り当てられているリソースをファイナライズします。

構文

```
public void MLFiniFileTransferInfo (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

備考

このメソッドは、ファイルのアップロードまたはダウンロードが完了した後で呼び出してください。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.6 MLFTEnableRsaE2ee(ml_file_transfer_info *) メソッド

RSA エンドツーエンド暗号化機能を指定できるようにします。

構文

```
public void MLFTEnableRsaE2ee (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.7 MLFTEnableRsaEncryption(ml_file_transfer_info *) メソッド

RSA 暗号化機能を指定できるようにします。

構文

```
public void MLFTEnableRsaEncryption (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.8 MLFTEnableRsaFipsE2ee(ml_file_transfer_info *) メソッド

RSAFIPS エンドツーエンド暗号化機能を指定できるようにします。

構文

```
public void MLFTEnableRsaFipsE2ee (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.9 MLFTEnableRsaFipsEncryption(ml_file_transfer_info *) メソッド

RSAFIPS 暗号化機能を指定できるようにします。

構文

```
public void MLFTEnableRsaFipsEncryption (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.10 MLFTEnableZlibCompression(ml_file_transfer_info *) メソッド

ZLIB 暗号化機能を指定できるようにします。

構文

```
public void MLFTEnableZlibCompression (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.11 MLInitFileTransferInfo(ml_file_transfer_info *) メソッド

ml_file_transfer_info 構造体を初期化します。

構文

```
public bool MLInitFileTransferInfo (ml_file_transfer_info * info)
```

パラメータ

info ファイル転送情報を含んだ構造。

備考

このメソッドは、ファイルのアップロードまたはダウンロードを開始する前に呼び出してください。

関連情報

ml_file_transfer_info 構造体 [61 ページ]

1.12 ULCancelGetNotification(SQLCA *, char const *) メソッド

指定された名前に一致する、すべてのキューに登録されている保留中の取得通知コールをキャンセルします。

構文

```
public ul_u_long ULCancelGetNotification (
    SQLCA * sqlca,
    char const * queue_name
)
```

パラメータ

sqlca SQLCA へのポインタ。
queue_name キューの名前。

戻り値

影響を受けるキューの数 (ブロックされた読み込み数とは異なります)。

1.13 ULChangeEncryptionKey(SQLCA *, char const *) メソッド

Ultra Light データベースの暗号化キーを変更します。

構文

```
public ul_bool ULChangeEncryptionKey (
    SQLCA * sqlca,
    char const * new_key
)
```

パラメータ

sqlca SQLCA へのポインタ。
new_key 新しい暗号化キー。

備考

このメソッドを呼び出すアプリケーションでは、データベースが同期されていること、または信頼できるバックアップコピーが作成されていることを、先に確認しておく必要があります。このメソッドは、実行を継続する必要のある操作であるため、信頼できるバックアップがあることが重要です。データベース暗号化キーを変更すると、まずデータベースのすべてのローは古いキーを使用して復号され、次に新しいキーを使用して再度暗号化されて、書き込まれます。この操作は元に戻せません。暗号化変更処理が完了しなかった場合、データベースは無効な状態のままになり、再度アクセスすることはできなくなります。

1.14 ULCheckpoint(SQLCA *) メソッド

チェックポイント操作を実行し、保留中になっているコミット済みトランザクションをデータベースにフラッシュします。

構文

```
public ul_ret_void ULCheckpoint (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

このメソッドを呼び出しても、現在のトランザクションすべてがコミットされるわけではありません。このメソッドは、パフォーマンスを向上させるために後回しされた自動トランザクションチェックポイントとともに使用されます。

このメソッドを使用すると、保留中のコミット済みトランザクションがすべてデータベースの記憶領域に書き込まれることが保証されます。

1.15 ULCountUploadRows(SQLCA *, char const *, ul_u_long) メソッド

同期するためにアップロードする必要があるローの数を数えます。

構文

```
public ul_u_long ULCountUploadRows (
    SQLCA * sqlca,
    char const * pub_list,
    ul_u_long threshold
)
```

パラメータ

sqlca SQL へのポインタ。

pub_list チェック対象となるパブリケーションのカンマ区切りのリストを含む文字列。空の文字列 (UL_SYNC_ALL マクロ) は、"非同期" とマーク付けされたものを除くすべてのテーブルを表します。アスタリスクのみの文字列 (UL_SYNC_ALL_PUBS マクロ) は、いずれかのパブリケーションで参照されているすべてのテーブルを表します。一部のテーブルは、どのパブリケーションの一部でもないため、pub_list string が "*" の場合は含まれません。

threshold カウントするローの最大数を判断します。呼び出しの所要時間を制限します。threshold が 0 の場合、制限はありません (つまり、同期する必要のあるすべてのローがメソッドによってカウントされます)。また、threshold が 1 の場合、同期の必要なローがあるかどうかを簡単に判別するために使用できます。

戻り値

指定されたパブリケーションのセットまたはデータベース全体のいずれかで、同期を必要とするローの数。

備考

このメソッドを使用すると、ユーザは同期、または自動バックグラウンド同期が行われるタイミングの決定を求められます。

次の呼び出しでは、データベース全体をチェックして、同期させるローの総数を確認します。

```
count = ULCountUploadRows( sqlca, UL_SYNC_ALL, 0 );
```

次の呼び出しでは、最大 1000 のローに対して PUB1 パブリケーションと PUB2 パブリケーションがチェックされます。

```
count = ULCountUploadRows( sqlca, UL_TEXT("PUB1,PUB2"), 1000 );
```

次の呼び出しでは、PUB1 パブリケーションと PUB2 パブリケーションで同期させる必要のあるローがあるかどうかがチェックされます。

```
count = ULCountUploadRows( sqlca, UL_TEXT("PUB1,PUB2"), 1 );
```

1.16 ULCreateDatabase(SQLCA *, char const *, char const *, void *) メソッド

Ultra Light データベースを作成します。

構文

```
public ul_bool ULCreateDatabase (
    SQLCA * sqlca,
    char const * connect_parms,
    char const * create_parms,
    void * reserved
)
```

パラメータ

sqlca 初期化済み SQLCA へのポインタ。

connect_parms セミコロンで区切った接続パラメータ文字列で、キーワード=値のペアで設定されます。接続文字列には、データベースの名前を含める必要があります。ここに含まれるパラメータは、データベースの接続時に指定されるパラメータセットと同じです。

create_parms セミコロンで区切った作成パラメータ文字列です。page_size=2048;checksum_level=2 などのように、キーワード=値のペアで設定されます。

reserved このパラメータは、今後の使用のために予約されています。

戻り値

データベースが正常に作成された場合は ul_true。それ以外の場合は ul_false を返します。通常、ul_false は、無効なファイル名やアクセスの拒否によって発生します。

備考

2 セットのパラメータで指定される情報を使用してデータベースが作成されます。

connect_parms パラメータは、データベースがアクセスされるたびに適用される接続パラメータのリストです。ファイル名、ユーザ ID、パスワード、オプションの暗号化キーは、その例です。

create_parms パラメータはパラメータのリストで、データベースの作成時にのみ意味を持ちます。難読化、ページサイズ、時間形式や日付形式は、その例です。

アプリケーションでこのメソッドを呼び出すことができるのは、SQLCA の初期化後です。

次のコードは、ULCreateDatabase メソッドを使用して、ファイル `myfile.udb` に Ultra Light データベースを作成する方法を示します。

```
if( ULCreateDatabase( &sqlca,
   "DBF=path/myfile.udb;key=anicelongpassphrase",
   ULGetCollation_1250LATIN2(),
   "checksum_level=2",
   NULL )
{
   // success
};
```

1.17 ULCreateNotificationQueue(SQLCA *, char const *, char const *) メソッド

この接続のイベント通知キューを作成します。

構文

```
public ul_bool ULCreateNotificationQueue (
    SQLCA * sqlca,
    char const * name,
    char const * parameters
)
```

パラメータ

sqlca SQLCA へのポインタ。
name 新しいキューの名前。
parameters 現在使われていません。NULL に設定されます。

戻り値

成功した場合は true、失敗した場合は false。

備考

キュー名は、接続ごとにスコープされるため、別々の接続で同じ名前を持つキューを作成できます。イベント通知が送信されると、データベース内で一致する名前を持つすべてのキューが、個別のインスタンスの通知を受け取ります。名前では、大文字と小文字が区別されません。ULRegisterForEvent メソッドを呼び出したときに、キューが指定されていない場合は、接続ごとにデフォルトのキューが作成されます。その名前がすでに存在する場合や有効でない場合は、エラーが発生して呼び出しが失敗します。

1.18 ULDeclareEvent(SQLCA *, char const *) メソッド

登録およびトリガされるイベントを宣言します。

構文

```
public ul_bool ULDeclareEvent (
    SQLCA * sqlca,
    char const * event_name
)
```

パラメータ

sqlca SQLCA へのポインタ。
event_name 新しいユーザ定義イベントの名前。

戻り値

イベントが正常に宣言された場合は true。正常に宣言されず、名前がすでに使用されているか無効な場合は false。

備考

Ultra Light では、データベースまたは環境での操作によってトリガされるシステムイベントの一部が事前に定義されています。この関数は、ユーザ定義イベントを宣言します。ユーザ定義イベントは、ULTriggerEvent メソッドでトリガされます。イベント名は、ユニークにする必要があります。名前では、大文字と小文字が区別されません。

関連情報

ULTriggerEvent(SQLCA *, char const *, char const *) メソッド [54 ページ]

1.19 ULDeleteAllRows(SQLCA *, ul_table_num) メソッド

テーブルからすべてのローを削除します。

構文

```
public ul_ret_void ULDeleteAllRows (
    SQLCA * sqlca,
    ul_table_num number
)
```

パラメータ

sqlca SQLCA へのポインタ。
number トランケートするテーブルの ID。

備考

アプリケーションによっては、テーブル内のローをすべて削除してから、新しいデータセットをテーブルにダウンロードする方が便利なことがあります。接続で stop sync プロパティが設定されている場合は、削除されたローが同期されません。

注記

別の接続からのコミットされていない挿入は削除されません。また、別の接続が DeleteAllRows メソッドを呼び出した後にロールバックを行った場合は、その接続からのコミットされていない削除は削除されません。

インデックスを使用しないでこのテーブルを開いた場合、テーブルは読み込み専用とみなされ、データを削除できません。

1.20 ULDestroyNotificationQueue(SQLCA *, char const *) メソッド

指定されたイベント通知キューを破棄します。

構文

```
public ul_bool ULDestroyNotificationQueue (
    SQLCA * sqlca,
    char const * name
)
```

パラメータ

sqlca SQLCA へのポインタ。
name 破棄するキューの名前。

戻り値

成功した場合は true、失敗した場合は false。

備考

キュー内に未読の通知が残っている場合は、警告が通知されます。未読の通知は破棄されます。接続のデフォルトのイベントキューが作成されている場合、接続が閉じると破棄されます。

1.21 ULEnableAesDBEncryption(SQLCA *) メソッド

AES データベース暗号化を有効にします。

構文

```
public ul_ret_void ULEnableAesDBEncryption (SQLCA * sqlca)
```

パラメータ

sqlca 初期化済み SQLCA へのポインタ。

備考

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。この関数を呼び出してから ULInitDatabaseManager メソッドを呼び出すようにしてください。

> **注記**
>
> このメソッドを呼び出すと、暗号化ルーチンがアプリケーションにインクルードされ、アプリケーションのコードサイズがその分だけ増加します。

1.22 ULEnableAesFipsDBEncryption(SQLCA *) メソッド

FIPS 140-2 認定 AES データベース暗号化を有効にします。

構文

```
public ul_ret_void ULEnableAesFipsDBEncryption (SQLCA * sqlca)
```

パラメータ

sqlca 初期化済み SQLCA へのポインタ。

備考

> **注記**
>
> このメソッドを呼び出すと、適切なルーチンがアプリケーションにインクルードされ、アプリケーションのコードサイズがその分だけ増加します。

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

関連情報

ULEnableAesDBEncryption(SQLCA *) メソッド [24 ページ]

1.23 ULEnableHttpSynchronization(SQLCA *) メソッド

HTTP 同期を有効にします。

構文

```
public ul_ret_void ULEnableHttpSynchronization (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

1.24 ULEnableRsaE2ee(SQLCA *) メソッド

RSA エンドツーエンド暗号化を有効にします。

構文

```
public ul_ret_void ULEnableRsaE2ee (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

1.25 ULEnableRsaFipsE2ee(SQLCA *) メソッド

FIPS 140-2 認定 RSA エンドツーエンド暗号化を有効にします。

構文

```
public ul_ret_void ULEnableRsaFipsE2ee (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

1.26 ULEnableRsaFipsSyncEncryption(SQLCA *) メソッド

SSL ストリームまたは TLS ストリームの RSA FIPS 暗号化を有効にします。

構文

```
public ul_ret_void ULEnableRsaFipsSyncEncryption (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

これは、ストリームパラメータを TLS または HTTPS に設定するときに必要です。

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

関連情報

ULEnableRsaSyncEncryption(SQLCA *) メソッド [28 ページ]

1.27 ULEnableRsaSyncEncryption(SQLCA *) メソッド

SSL ストリームまたは TLS ストリームの RSA 暗号化を有効にします。

構文

```
public ul_ret_void ULEnableRsaSyncEncryption (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

これは、ストリームパラメータを TLS または HTTPS に設定するときに必要です。

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

関連情報

ULEnableRsaFipsSyncEncryption(SQLCA *) メソッド [27 ページ]

1.28 ULEnableTcpipSynchronization(SQLCA *) メソッド

TCP/IP 同期を有効にします。

構文

```
public ul_ret_void ULEnableTcpipSynchronization (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

1.29 ULEnableZlibSyncCompression(SQLCA *) メソッド

同期ストリームの ZLIB 圧縮を有効にします。

構文

```
public ul_ret_void ULEnableZlibSyncCompression (SQLCA * sqlca)
```

パラメータ

sqlca 初期化済み SQLCA へのポインタ。

備考

このメソッドは、C++ API アプリケーションと Embedded SQL アプリケーションで使用できます。このメソッドを呼び出してから Synchronize メソッドを呼び出すようにしてください。同期タイプを有効にする呼び出しを実行する前に同期しようとすると、SQLE_METHOD_CANNOT_BE_CALLED エラーが発生します。

1.30 ULErrorInfoInitFromSqlca(ul_error_info *, SQLCA const *) メソッド

SQLCA to the ul_error_info オブジェクトからエラー情報をコピーします。

構文

```
public void ULErrorInfoInitFromSqlca (
    ul_error_info * errinf,
    SQLCA const * sqlca
)
```

パラメータ

sqlca SQLCA へのポインタ。
errinf ul_error_info object です。

1.31 ULErrorInfoParameterAt(ul_error_info const *, ul_u_short, char *, size_t) メソッド

序数を指定してエラーパラメータを取得します。

構文

```
public size_t ULErrorInfoParameterAt (
    ul_error_info const * errinf,
    ul_u_short parmNo,
    char * buffer,
    size_t bufferSize
)
```

パラメータ

errinf ul_error_info object です。
parmNo パラメータの、1 から始まる序数。
buffer パラメータ文字列を受け取るバッファ。
bufferSize バッファのサイズ。

戻り値

パラメータの格納に必要なサイズ (バイト単位)。または、序数が無効の場合は 0。戻り値が bufferSize 値より大きい場合、パラメータはトランケートされています。

1.32 ULErrorInfoParameterCount(ul_error_info const *) メソッド

エラーパラメータの数を取得します。

構文

```
public ul_u_short ULErrorInfoParameterCount (ul_error_info const * errinf)
```

パラメータ

errinf ul_error_info object です。

戻り値

エラーパラメータ数。

1.33 ULErrorInfoString(ul_error_info const *, char *, size_t) メソッド

エラーの説明を取得します。

構文

```
public size_t ULErrorInfoString (
    ul_error_info const * errinf,
    char * buffer,
    size_t bufferSize
)
```

パラメータ

errinf ul_error_info object です。
buffer エラーの説明を受信するバッファ。
bufferSize バッファのサイズ (バイト数)。

戻り値

文字列の格納に必要なサイズ (バイト単位)。戻り値が len 値より大きい場合、文字列はトランケートされています。

1.34 ULErrorInfoURL(ul_error_info const *, char *, size_t, char const *) メソッド

このエラーの資料ページの URL を取得します。

構文

```
public size_t ULErrorInfoURL (
    ul_error_info const * errinf,
    char * buffer,
    size_t bufferSize,
    char const * reserved
)
```

パラメータ

errinf ul_error_info object です。
buffer URL を受け取るバッファ。
bufferSize バッファのサイズ (バイト数)。
reserved 今後の使用のために予約済み。

戻り値

URL の格納に必要なサイズ (バイト単位)。戻り値が len 値より大きい場合、URL はトランケートされています。

1.35 ULGetDatabaseID(SQLCA *) メソッド

グローバルオートインクリメントカラムに使用される現在のデータベース ID を取得します。

構文

```
public ul_u_long ULGetDatabaseID (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

戻り値

SetDatabaseID メソッドへの最後の呼び出しで設定された値。または、これまでに ID が設定されていない場合は UL_INVALID_DATABASE_ID。

1.36 ULGetDatabaseProperty(SQLCA *, ul_database_property_id, char *, size_t, ul_bool *) メソッド

データベースプロパティの値を取得します。

構文

```
public void ULGetDatabaseProperty (
    SQLCA * sqlca,
    ul_database_property_id id,
    char * dst,
    size_t buffer_size,
    ul_bool * null_indicator
)
```

パラメータ

sqlca SQLCA へのポインタ。

id データベースプロパティの識別子。

dst プロパティの値を格納する文字配列。

buffer_size 文字配列 dst のサイズ。

null_indicator データベースパラメータが NULL であることを示すインジケータ。

1.37 ULGetErrorParameter(SQLCA const *, ul_u_long, char *, size_t) メソッド

序数のパラメータ番号を使用してエラーパラメータを取得します。

構文

```
public size_t ULGetErrorParameter (
    SQLCA const * sqlca,
    ul_u_long parm_num,
    char * buffer,
    size_t size
)
```

パラメータ

sqlca SQLCA へのポインタ。

parm_num 序数のパラメータ番号。
buffer エラーパラメータを格納するバッファへのポインタ。
size バッファのサイズ (バイト数)。

戻り値

このメソッドは、指定されたバッファにコピーされる文字数を返します。

関連情報

ULGetErrorParameterCount(SQLCA const *) メソッド [35 ページ]

1.38 ULGetErrorParameterCount(SQLCA const *) メソッド

エラーパラメータの数を取得します。

構文

```
public ul_u_long ULGetErrorParameterCount (SQLCA const * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

戻り値

エラーパラメータ数。結果が 0 の場合を除き、1 からこの戻り値までの値を使用して、ULGetErrorParameter メソッドを呼び出し、対応するエラーパラメータ値を取得できます。

関連情報

ULGetErrorParameter(SQLCA const *, ul_u_long, char *, size_t) メソッド [34 ページ]

1.39 ULGetIdentity(SQLCA *) メソッド

@identity の値を取得します。

構文

```
public ul_u_big ULGetIdentity (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

戻り値

オートインクリメントカラムまたはグローバルオートインクリメントカラムに最後に挿入された値。

1.40 ULGetLastDownloadTime(SQLCA *, char const *, DECL_DATETIME *) メソッド

指定したパブリケーションが最後にダウンロードされた時刻を取得します。

構文

```
public ul_bool ULGetLastDownloadTime (
    SQLCA * sqlca,
    char const * pub_name,
    DECL_DATETIME * value
)
```

パラメータ

sqlca SQLCA へのポインタ。

pub_name 最終ダウンロード時間を取得するパブリケーション名を含む文字列。

value 投入する DECL_DATETIME 構造体へのポインタ。たとえば、値 January 1, 1990 は、パブリケーションがまだ同期されていないことを示します。

戻り値

pub-name 値によって指定されたパブリケーションの最終ダウンロード時間までに value が正常に投入された場合は true。それ以外の場合は、false を返します。

備考

次の呼び出しでは、UL_PUB_PUB1 パブリケーションがダウンロードされた日付と時刻とともに dt 構造体が投入されます。

```
DECL_DATETIME dt;
ret = ULGetLastDownloadTime( &sqlca, UL_TEXT("UL_PUB_PUB1"), &dt );
```

1.41 ULGetNotification(SQLCA *, char const *, char *, ul_length, ul_u_long) メソッド

イベント通知を読み込みます。

構文

```
public ul_bool ULGetNotification (
    SQLCA * sqlca,
    char const * queue_name,
    char * event_name_buf,
    ul_length event_name_buf_len,
    ul_u_long wait_ms
)
```

パラメータ

sqlca SQLCA へのポインタ。
queue_name 読み取るキュー。または、デフォルト接続キューの場合は NULL。
event_name_buf イベント名を保持しているバッファ。
event_name_buf_len バッファのサイズ (バイト単位)。
wait_ms 返す前に、待機 (ブロック) する時間 (ミリ秒単位)。

戻り値

成功した場合は true、失敗した場合は false。

備考

この呼び出しは、通知が受信されるまで、または指定された待機時間が経過するまでブロックします。wait_ms parameter に UL_READ_WAIT_INFINITE を渡し、無期限に待機します。待機をキャンセルするには、指定したキューに別の通知を送信するか、ULCancelGetNotification メソッドを使用します。通知を読み込んだら、ULGetNotificationParameter メソッドを使用して、追加のパラメータを名前で取得します。

関連情報

ULCancelGetNotification(SQLCA *, char const *) メソッド [17 ページ]
ULGetNotificationParameter(SQLCA *, char const *, char const *, char *, ul_length) メソッド [38 ページ]

1.42 ULGetNotificationParameter(SQLCA *, char const *, char const *, char *, ul_length) メソッド

ULGetNotification メソッドによって読み込まれたイベント通知のパラメータを取得します。

構文

```
public ul_bool ULGetNotificationParameter (
    SQLCA * sqlca,
    char const * queue_name,
    char const * parameter_name,
    char * value_buf,
    ul_length value_buf_len
)
```

パラメータ

sqlca SQLCA へのポインタ。
queue_name 読み取るキュー。デフォルト接続キューの場合は NULL。
parameter_name 読み込むパラメータの名前 (または "*")。
value_buf パラメータ値を保持しているバッファ。
value_buf_len バッファのサイズ (バイト単位)。

戻り値

成功した場合は true、失敗した場合は false。

備考

指定されたキューで最近読み込まれた通知のパラメータのみが使用可能です。パラメータは名前によって取得されます。パラメータ名を "*" と指定すると、パラメータ文字列全体が取得されます。

1.43 ULGetSyncResult(SQLCA *, ul_sync_result *) メソッド

前回の同期結果を取得します。

構文

```
public ul_bool ULGetSyncResult (
    SQLCA * sqlca,
    ul_sync_result * sync_result
)
```

パラメータ

sqlca SQLCA へのポインタ。
sync_result 同期の結果を保持する ul_sync_result 構造体へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

1.44 ULGlobalAutoincUsage(SQLCA *) メソッド

グローバルオートインクリメントのデフォルト値を持つすべてのカラムで、デフォルト値が使用されている比率 (%) を取得します。

構文

```
public ul_u_short ULGlobalAutoincUsage (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

戻り値

グローバルオートインクリメントの値のカウンタによる使用済み比率 (%)。

備考

このデフォルト値を使用するカラムがデータベース内に複数含まれている場合は、すべてのカラムに対してこの値が計算され、最大値が返されます。たとえば、戻り値 99 は、少なくとも 1 つのカラムではデフォルト値が残されているが、きわめて少ないことを示します。

関連情報

ULSetDatabaseID(SQLCA *, ul_u_long) メソッド [47 ページ]

1.45 ULGrantConnectTo(SQLCA *, char const *, char const *) メソッド

指定されたパスワードを持つ新しいまたは既存のユーザ ID に、Ultra Light データベースへのアクセスを許可します。

構文

```
public ul_ret_void ULGrantConnectTo (
    SQLCA * sqlca,
    char const * uid,
    char const * pwd
)
```

パラメータ

sqlca SQLCA へのポインタ。
uid ユーザ ID を保持する文字配列。

pwd ユーザ ID のパスワードを保持する文字配列。

備考

このメソッドは、既存のユーザ ID を指定したときに、既存のユーザのパスワードを更新します。

関連情報

ULRevokeConnectFrom(SQLCA *, char const *) メソッド [45 ページ]

1.46 ULInitSyncInfo(ul_sync_info *) メソッド

同期情報の構造体を初期化します。

構文

```
public ul_ret_void ULInitSyncInfo (ul_sync_info * info)
```

パラメータ

info 同期構造体。

1.47 ULIsSynchronizeMessage(ul_u_long) メソッド

メッセージが ActiveSync に対する Mobile Link プロバイダからの同期メッセージであるかどうかを確認し、そのメッセージを処理するコードを呼び出すことができます。

構文

```
public ul_bool ULIsSynchronizeMessage (ul_u_long number)
```

備考

同期メッセージの処理が完了したときに、ULSignalSyncIsComplete メソッドを呼び出す必要があります。

このメソッドの呼び出しを、使用しているアプリケーションの WindowProc 関数にインクルードしてください。ActiveSync を使用する Windows Mobile に適用されます。

以下のコードは、ULIsSynchronizeMessage メソッドを使用した同期メッセージの処理方法の箇所を抜粋したものです。

```
LRESULT CALLBACK WindowProc( HWND hwnd,
        UINT uMsg,
        WPARAM wParam,
        LPARAM lParam )
{
 if( ULIsSynchronizeMessage( uMsg ) ) {
   // execute synchronization code
   if( wParam == 1 ) DestroyWindow( hWnd );
   return 0;
 }
 switch( uMsg ) {
 // code to handle other windows messages
 default:
   return DefWindowProc( hwnd, uMsg, wParam, lParam );
 }
 return 0;
}
```

関連情報

ULSignalSyncIsComplete() メソッド [51 ページ]

1.48 ULLibraryVersion(void) メソッド

Ultra Light ランタイムライブラリのバージョン番号を返します。

構文

```
public char const * ULLibraryVersion (void)
```

戻り値

Ultra Light ランタイムライブラリのバージョン番号。

1.49 ULRegisterForEvent(SQLCA *, char const *, char const *, char const *, ul_bool) メソッド

イベントの通知を受け取るためのキューを登録または登録解除します。

構文

```
public ul_bool ULRegisterForEvent (
    SQLCA * sqlca,
    char const * event_name,
    char const * object_name,
    char const * queue_name,
    ul_bool register_not_unreg
)
```

パラメータ

sqlca SQLCA へのポインタ。
event_name 登録するシステム定義またはユーザ定義のイベント。
object_name イベントを適用するオブジェクト (テーブル名など)。
queue_name 接続キュー名。NULL は、デフォルトの接続キューを表します。
register_not_unreg 登録する場合は true、登録解除する場合は false。

戻り値

正常に登録できた場合は true、キューまたはイベントが存在しない場合は false。

備考

キュー名が指定されていない場合は、デフォルトの接続キューが暗黙で指定され、必要に応じて作成されます。特定のシステムイベントでは、そのイベントが適用されるオブジェクト名を指定できます。たとえば、TableModified イベントではテーブル名を指定できます。ULSendNotification メソッドとは異なり、登録された特定のキューのみイベントの通知を受信します。別の接続に、同じ名前の他のキューがある場合、それらは、同様に明示的に登録されていないかぎり、通知を受信しません。

事前に定義されたシステムイベントは次のとおりです。

TableModified

テーブルのローが挿入、更新、または削除されたときにトリガされます。要求の影響を受けるローの数にかかわらず、要求ごとに 1 つの通知が送信されます。object_name パラメータは、モニタするテーブルを指定します。値 "*" は、データベース内のすべてのテーブルを意味します。このイベントには、table_name というパラメータがあり、このパラメータの値は変更されたテーブルの名前です。

Commit

コミットが完了した後にトリガされます。このイベントにはパラメータはありません。

SyncComplete

同期が完了した後にトリガされます。このイベントにはパラメータはありません。

1.50 ULResetLastDownloadTime(SQLCA *, char const *) メソッド

アプリケーションが以前にダウンロードされたデータを再同期するように、パブリケーションの最終ダウンロード時間をリセットします。

構文

```
public ul_ret_void ULResetLastDownloadTime (
    SQLCA * sqlca,
    char const * pub_list
)
```

パラメータ

sqlca SQLCA へのポインタ。

pub_list リセットするパブリケーションのカンマ区切りのリストを含む文字列。空の文字列は、"非同期" とマーク付けされたものを除くすべてのテーブルを割り当てます。アスタリスクのみの文字列 ("*") は、すべてのパブリケーションを割り当てます。一部のテーブルは、どのパブリケーションの一部でもないため、pub_list string が "*" の場合は含まれません。

備考

次のメソッド呼び出しは、すべてのテーブルの最終ダウンロード時間をリセットします。

```
ULResetLastDownloadTime( &sqlca, UL_TEXT("*") );
```

1.51 ULRevokeConnectFrom(SQLCA *, char const *) メソッド

Ultra Light データベースからユーザ ID のアクセス権を取り消します。

構文

```
public ul_ret_void ULRevokeConnectFrom (
    SQLCA * sqlca,
    char const * uid
)
```

パラメータ

sqlca SQLCA へのポインタ。
uid データベースアクセスから除外するユーザ ID を保持する文字配列。

1.52 ULRollbackPartialDownload(SQLCA *) メソッド

失敗した同期からの変更をロールバックします。

構文

```
public ul_ret_void ULRollbackPartialDownload (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

備考

同期のダウンロード時に通信エラーが発生した場合、Ultra Light ではダウンロードした変更を適用できるため、アプリケーションでは同期が中断した時点から同期を再開することができます。ダウンロードした変更が不要である (ダウンロードが中断した時点での再開を望まない) 場合は、ULRollbackPartialDownload メソッドを使用することで、失敗したダウンロードトランザクションをロールバックします。

1.53 ULRSALibraryVersion(void) メソッド

RSA 暗号化ライブラリのバージョン番号を返します。

構文

```
public char const * ULRSALibraryVersion (void)
```

戻り値

RSA 暗号化ライブラリのバージョン番号。

1.54 ULSendNotification(SQLCA *, char const *, char const *, char const *) メソッド

指定された名前と一致するすべてのキューに通知を送信します。

構文

```
public ul_u_long ULSendNotification (
    SQLCA * sqlca,
    char const * queue_name,
    char const * event_name,
    char const * parameters
)
```

パラメータ

sqlca SQLCA へのポインタ。
queue_name 接続キュー名。NULL は、デフォルトの接続キューを表します。
event_name 登録するシステム定義またはユーザ定義のイベント。
parameters 現在使われていません。NULL に設定されます。

戻り値

送信済みの通知の数 (一致するキューの数)。

備考

これに含まれるのは、現在の接続におけるキューです。この呼び出しはブロックしません。特別なキュー名の "*" を使用すると、すべてのキューに送信します。指定されたイベント名は、システム定義またはユーザ定義のイベントと対応する必要はありません。読み込まれたイベント名は、通知を識別するためにそのまま渡され、送信者と受信者に対してしか意味を持たないからです。

パラメータの値には、"名前=値" のペアをセミコロンで区切ったオプションリストを指定します。通知が読み込まれた後、パラメータの値が ULGetNotificationParameter メソッドによって読み込まれます。

関連情報

ULGetNotificationParameter(SQLCA *, char const *, char const *, char *, ul_length) メソッド [38 ページ]

1.55 ULSetDatabaseID(SQLCA *, ul_u_long) メソッド

データベースの ID 番号を設定します。

構文

```
public ul_ret_void ULSetDatabaseID (
    SQLCA * sqlca,
    ul_u_long value
)
```

パラメータ

sqlca SQLCA へのポインタ。
value レプリケーションまたは同期を設定する時に、特定のデータベースをユニークに識別する正の整数。

関連情報

ULGlobalAutoincUsage(SQLCA *) メソッド [39 ページ]

1.56 ULSetDatabaseOptionString(SQLCA *, ul_database_option_id, char const *) メソッド

文字列値からデータベースオプションを設定します。

構文

```
public void ULSetDatabaseOptionString (
    SQLCA * sqlca,
    ul_database_option_id id,
    char const * value
)
```

パラメータ

sqlca SQLCA へのポインタ。
id 設定するデータベースオプションの識別子。
value データベースオプションの値。

1.57 ULSetDatabaseOptionULong(SQLCA *, ul_database_option_id, ul_u_long) メソッド

数値のデータベースオプションを設定します。

構文

```
public void ULSetDatabaseOptionULong (
    SQLCA * sqlca,
    ul_database_option_id id,
    ul_u_long value
)
```

パラメータ

sqlca SQLCA へのポインタ。
id 設定するデータベースオプションの識別子。
value データベースオプションの値。

1.58 ULSetErrorCallback(SQLCA *, ul_error_callback_fn_a, ul_void *, char *, size_t) メソッド

エラーの発生時に呼び出されるようコールバックを設定します。

構文

```
public ul_ret_void ULSetErrorCallback (
    SQLCA * sqlca,
    ul_error_callback_fn_a callback,
    ul_void * user_data,
    char * buffer,
    size_t len
)
```

パラメータ

sqlca SQLCA へのポインタ。

callback コールバック関数。

user_data コールバックに渡されるユーザコンテキスト情報。

buffer コールバックが呼び出されたときにエラーパラメータを保持する、ユーザが提供するバッファ。

len バッファのサイズ (バイト数)。

1.59 ULSetSynchronizationCallback(SQLCA *, ul_sync_observer_fn, ul_void *) メソッド

同期の実行中に呼び出されるようコールバックを設定します。

構文

```
public ul_ret_void ULSetSynchronizationCallback (
    SQLCA * sqlca,
    ul_sync_observer_fn callback,
    ul_void * user_data
)
```

パラメータ

sqlca SQLCA へのポインタ。

callback コールバック。
user_data コールバックに渡されるユーザコンテキスト情報。

1.60 ULSetSyncInfo(SQLCA *, char const *, ul_sync_info *) メソッド

指定された ul_sync_info 構造体に基づいて、指定された名前を使用して同期プロファイルを作成します。

構文

```
public ul_bool ULSetSyncInfo (
    SQLCA * sqlca,
    char const * profile_name,
    ul_sync_info * sync_info
)
```

パラメータ

sqlca SQLCA へのポインタ。
profile_name 同期プロファイルの名前。
sync_info 同期パラメータを保持する ul_sync_info 構造体へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

備考

同じ名前の同期プロファイルがすでにある場合は、この同期プロファイルで置き換えられます。構造体に NULL ポインタを指定することによって、指定されたプロファイルが削除されます。

1.61 ULSignalSyncIsComplete() メソッド

同期メッセージの処理が完了したことを示します。

構文

```
public ul_ret_void ULSignalSyncIsComplete ()
```

備考

Microsoft MActiveSync プロバイダで登録したアプリケーションは、同期メッセージの処理が完了したときに、WNDPROC でこのメソッドを呼び出す必要があります。

1.62 ULStartSynchronizationDelete(SQLCA *) メソッド

この接続の START SYNCHRONIZATION DELETE を設定します。

構文

```
public ul_ret_void ULStartSynchronizationDelete (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

1.63 ULStaticFini() メソッド

Embedded SQL アプリケーションで Ultra Light ランタイムのファイナライズを実行します。

構文

```
public void ULStaticFini ()
```

備考

このメソッドは、アプリケーションごとに 1 回だけ呼び出してください。その後、他の Ultra Light メソッドを呼び出すことはできません。

1.64 ULStaticInit() メソッド

Embedded SQL アプリケーションで Ultra Light ランタイムの初期化を実行します。

構文

```
public void ULStaticInit ()
```

備考

このメソッドは、他の Ultra Light メソッドを呼び出す前に、アプリケーションごとに 1 回だけ呼び出してください。

1.65 ULStopSynchronizationDelete(SQLCA *) メソッド

この接続の STOP SYNCHRONIZATION DELETE を設定します。

構文

```
public ul_bool ULStopSynchronizationDelete (SQLCA * sqlca)
```

パラメータ

sqlca SQLCA へのポインタ。

戻り値

成功した場合は true、失敗した場合は false。

1.66 ULSynchronize(SQLCA *, ul_sync_info *) メソッド

Ultra Light アプリケーションで同期を開始します。

構文

```
public ul_ret_void ULSynchronize (
    SQLCA * sqlca,
    ul_sync_info * info
)
```

パラメータ

sqlca SQLCA へのポインタ。
info 同期パラメータを保持する ul_sync_info 構造体へのポインタ。

備考

TCP/IP または HTTP 同期では、ULSynchronize メソッドが同期を開始します。同期中のエラーで handle_error スクリプトによって処理されないものは、SQL エラーとしてレポートされます。アプリケーションプログラムでは、このメソッドの SQLCODE 戻り値をテストする必要があります。

次の例は、データベースの同期を示しています。

```
ul_sync_info info;
ULInitSyncInfo( &info );
info.user_name = UL_TEXT( "user_name" );
info.version = UL_TEXT( "test" );
ULSynchronize( &sqlca, &info );
```

1.67 ULSynchronizeFromProfile(SQLCA *, char const *, char const *, ul_sync_observer_fn, ul_void *) メソッド

指定されたプロファイルとマージパラメータを使用して、データベースを同期します。

構文

```
public ul_ret_void ULSynchronizeFromProfile (
    SQLCA * sqlca,
    char const * profile_name,
    char const * merge_parms,
    ul_sync_observer_fn observer,
```

```
    ul_void * user_data
)
```

パラメータ

sqlca SQLCA へのポインタ。
profile_name 同期するプロファイルの名前。
merge_parms 同期で使用するマージパラメータ。
observer ステータス更新の送信先となる observer コールバック。
user_data コールバックに渡されるユーザコンテキストデータ。

備考

このメソッドは、SYNCHRONIZE 文を実行するのと同じです。

1.68 ULTriggerEvent(SQLCA *, char const *, char const *) メソッド

ユーザ定義のイベントをトリガして、登録されたすべてのキューに通知を送信します。

構文

```
public ul_u_long ULTriggerEvent (
    SQLCA * sqlca,
    char const * event_name,
    char const * parameters
)
```

パラメータ

sqlca SQLCA へのポインタ。
event_name 登録するシステム定義またはユーザ定義のイベント。
parameters 現在使われていません。NULL に設定されます。

戻り値

送信済みのイベント通知の数。

備考

パラメータの値には、"名前=値" のペアをセミコロンで区切ったオプションリストを指定します。通知が読み込まれた後、パラメータの値が ULGetNotificationParameter メソッドによって読み込まれます。

関連情報

ULGetNotificationParameter(SQLCA *, char const *, char const *, char *, ul_length) メソッド [38 ページ]

1.69 ULTruncateTable(SQLCA *, ul_table_num) メソッド

テーブルをトランケートし、STOP SYNCHRONIZATION DELETE 文を一時的にアクティブにします。

構文

```
public ul_ret_void ULTruncateTable (
    SQLCA * sqlca,
    ul_table_num number
)
```

パラメータ

sqlca SQLCA へのポインタ。
number トランケートするテーブルの ID。

1.70 ULValidateDatabaseLowLevelA(SQLCA *, char const *, ul_u_short, ul_validate_callback_fn, void *) メソッド

指定されたデータベースの低レベル構造を検証します。

構文

```
public ul_bool ULValidateDatabase (
    SQLCA * sqlca,
    char const * start_parms,
    ul_u_short flags,
    ul_validate_callback_fn callback_fn,
    void * user_data
)
```

パラメータ

sqlca SQLCA へのポインタ。

connect_parms チェックするデータベースを特定する接続パラメータ。

flags `ULVF_` 検証フラグ: **`ULVF_DATABASE`** を特定します。

callback_fn 検証の情報を受け取る関数。

user_data `callback_fn` に渡されたユーザデータ。

戻り値

検証で問題が見つからなければ True、問題が検出されれば False。

備考

この機能を使用して、通常の ULValidateDatabaseConnectedA() の使用を適切に開始しなくなる、データベースの低レベルの破損を検出します。

1.71 ULValidateDatabaseConnectedA(SQLCA *, char const *, ul_u_short, ul_validate_callback_fn, void *) メソッド

現在の接続でのデータベースを検証します。

構文

```
public ul_bool ULValidateDatabaseTableName (
    SQLCA * sqlca,
    char const * table_name,
    ul_u_short flags,
    ul_validate_callback_fn callback_fn,
    void * user_data
)
```

パラメータ

sqlca アクティブな SQLCA ポインタ。

table_name 検証する特定のテーブル名。すべてのテーブルの場合は NULL。

flags `ULVF_` 検証フラグ。

callback_fn 検証の情報を受け取る関数。

user_data `callback_fn` に渡されたユーザデータ。

戻り値

検証で問題が見つからなければ True、問題が検出されれば False。

備考

検証について情報を受け取るには、コールバック関数を実装し、アドレスをこのルーチンに渡します。

フラグのパラメータは検証対象と、次のビット値の組み合わせ対象を決定します。

- ULVF_TABLE
- ULVF_INDEX
- ULVF_DATABASE

また、種別は次のとおりです。

- ULVF_EXPRESS

ULVF_FULL_VALIDATE は (緊急以外の) すべての検証の省略形です。

検証対象を特定のテーブルに限定するには、名称と `table_name` パラメータを指定します。

1.72 ul_database_option_id 列挙体

ユーザが設定できる可能なデータベースオプションを設定します。

構文

```
enum ul_database_option_id
```

メンバー

メンバー名	説明
ul_option_global_database_id	グローバルデータベース ID は、符号なし long 型整数を使用して設定されます。
ul_option_ml_remote_id	リモート ID は文字列を使用して設定されます。
ul_option_commit_flush_timeout	データベースのコミットフラッシュタイムアウトは、ミリ秒単位で測定される時間スレッショルドを表す整数として設定されます。
ul_option_commit_flush_count	データベースのコミットフラッシュカウントは、コミットカウントスレッショルドを表す整数として設定されます。
ul_option_isolation_level	接続の独立性レベルは文字列として設定されます。 (read_committed/read_uncommitted)
ul_option_cache_allocation	データベースファイルキャッシュを変更するために設定します。 割り当てるキャッシュの量の最小から最大までのサイズの幅を、0 ～ 100 までの整数値で設定します。

備考

これらのデータベースオプションは ULConnection.SetDatabaseOption メソッドで使用されます。イニシャライザを表示: いいえ

1.73 ul_database_property_id 列挙体

ユーザが取得できる可能なデータベースプロパティを設定します。

構文

```
enum ul_database_property_id
```

メンバー

メンバー名	説明
ul_property_date_format	**日付形式。** (date_format)
ul_property_date_order	**日付順。** (date_order)
ul_property_nearest_century	**基準年。** (nearest_century)
ul_property_precision	**精度。** (precision)
ul_property_scale	**位取り。** (scale)
ul_property_time_format	**時間形式。** (time_format)
ul_property_timestamp_format	**タイムスタンプ形式。** (timestamp_format)
ul_property_timestamp_increment	**タイムスタンプインクリメント。** (timestamp_increment)
ul_property_name	**名前。** (Name)
ul_property_file	**ファイル。** (File)
ul_property_encryption	**暗号化。** (Encryption)

メンバー名	説明
ul_property_global_database_id	グローバルデータベース ID。 (global_database_id)
ul_property_ml_remote_id	リモート ID。 (ml_remote_id)
ul_property_char_set	文字セット。 (CharSet)
ul_property_collation	照合順。 (Collation)
ul_property_page_size	ページサイズ。 (PageSize)
ul_property_case_sensitive	大文字と小文字の区別。 (CaseSensitive)
ul_property_conn_count	接続数。 (ConnCount)
ul_property_max_hash_size	デフォルトの最大インデックスハッシュ。 (MaxHashSize)
ul_property_checksum_level	データベースのチェックサムレベル。 (ChecksumLevel)
ul_property_checkpoint_count	データベースのチェックポイント数。 (CheckpointCount)
ul_property_commit_flush_timeout	データベースのコミットフラッシュタイムアウト。 (commit_flush_timeout)
ul_property_commit_flush_count	データベースのコミットフラッシュカウント。 (commit_flush_count)
ul_property_isolation_level	接続の独立性レベル。 (isolation_level)
ul_property_timestamp_with_time_zone_format	タイムゾーン付きタイムスタンプ形式。 (timestamp_with_time_zone_format)
ul_property_cache_allocation	現在のデータベースサイズキャッシュのサイズを最小から最大までのパーセンテージで表したものです。
ul_property_partial_download	部分的なダウンロードの存在を示しています (再開可能)。 (PartialDownload)

メンバー名	説明
ul_property_upload_unknown	前回のアップロードのステータスが不明であることを示します。 (UploadUnknown)

備考

これらのプロパティは ULConnection.GetDatabaseProperty メソッドで使用されます。

イニシャライザを表示: いいえ

1.74 ml_file_transfer_info 構造体

ファイルのアップロードまたはダウンロードのパラメータが格納された構造体。

構文

```
typedef struct ml_file_transfer_info
```

メンバー

ml_file_transfer_info のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

変数

変更子とタイプ	変数	説明
public const char *	filename	Mobile Link を実行しているサーバから転送されるファイルの名前。 Mobile Link は username サブフォルダを検索してから、デフォルトのルートフォルダを検索します。

変更子とタイプ	変数	説明
public const char *	local_path	ダウンロードファイルの格納先となるローカルパス。 このパラメータが空の場合 (デフォルト)、ダウンロードファイルは現在のフォルダに格納されます。 Windows Mobile では、dest_path が空の場合、ファイルはデバイスのルート () フォルダに格納されます。 デスクトップコンピュータでは、dest_path 値が空の場合、ファイルはユーザの現在のフォルダに格納されます。
public const char *	local_filename	ダウンロードファイルのローカル名。 このパラメータが空の場合、ファイル名の値が使用されます。
public const char *	stream	プロトコルは次のいずれかです。TCPIP、TLS、HTTP、HTTPS このフィールドは必須です。
public const char *	stream_parms	指定されたストリームのプロトコルのオプション。
public const char *	username	Mobile Link ユーザ名 このフィールドは必須です。
public const char *	remote_key	Mobile Link リモートキー。
public const char *	password	Mobile Link ユーザ名のパスワード。
public const char *	version	Mobile Link スクリプトのバージョン。 このフィールドは必須です。
public ml_file_transfer_observer_fn	observer	'observer' フィールドを使用することで、ファイルのダウンロードの進捗状況を確認するコールバックを実現できます。 詳細については、後述のコールバック関数の説明を参照してください。
public void *	user_data	同期 observer で使用できるようにした、アプリケーション固有の情報。

変更子とタイプ	変数	説明
public bool	enable_resume	true に設定すると、MLFileDownload メソッドは、通信エラーまたはユーザのキャンセルによって中断した以前のダウンロードを再開します。 サーバ上のファイルがローカルの部分ファイルより新しい場合、部分ファイルが破棄され、新しいバージョンがあらためてダウンロードされます。デフォルトは true です。
public asa_uint8	num_auth_parms	Mobile Link イベントの認証パラメータに渡される認証パラメータの数。
public const char **	auth_parms	Mobile Link イベントの認証パラメータにパラメータを渡します。
public asa_uint16	transferred_file	ファイルが正常に転送された場合は 1、エラーが発生した場合は 0。 MLFileUpload の呼び出し時にファイルがすでに最新の状態になっていると、エラーが発生します。この関数は、false ではなく true を返します。
public asa_uint16	auth_status	Mobile Link イベントの認証パラメータにパラメータを渡します。
public asa_uint32	auth_value	カスタム Mobile Link のユーザ認証スクリプトの結果をレポートします。 Mobile Link サーバが、この情報をクライアントに提供します。
public asa_uint16	file_auth_code	サーバ上の authenticate_file_transfer スクリプト (オプション) のリターンコードが格納されます。
public mlft_stream_error	error	発生したエラーに関する情報が格納されます。

1.75 ml_file_transfer_status 構造体

ファイルのアップロードまたはダウンロードの進行中の、ステータスまたは進行状況情報が格納された構造体。

構文

```
typedef struct ml_file_transfer_status
```

メンバー

ml_file_transfer_status のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

変数

変更子とタイプ	変数	説明
public asa_uint64	file_size	ダウンロード中のファイルの合計サイズ (バイト単位) を示します。
public asa_uint64	bytes_transferred	現時点でダウンロード済みのファイルのサイズを示します。再開されたダウンロードの場合は、以前の同期も含みます。
public asa_uint64	resumed_at_size	ダウンロードの再開で使用され、現在のダウンロードの開始点を示します。
public ml_file_transfer_info *	info	MLFileDownload メソッドに渡された information オブジェクトへのポインタ。 このポインタを使用して user_data パラメータにアクセスできます。
public asa_uint16	flags	追加情報が格納されます。 MLFileDownload メソッドがネットワーク呼び出しをブロックしており、observer メソッドが前回呼び出されたときからダウンロードステータスが変わっていない場合は、MLFT_STATUS_FLAG_IS_BLOCKING が設定されます。
public asa_uint8	stop	true に設定すると、現在のダウンロードがキャンセルされます。 enable_resume パラメータが設定された場合にかぎり、MLFileDownload メソッドを後で呼び出したときにそのダウンロードを再開できます。

1.76 mlft_stream_error 構造体

ファイルのアップロードまたはダウンロードの進行中の、ステータスまたは進行状況情報が格納された構造体。

構文

```
typedef struct mlft_stream_error
```

メンバー

mlft_stream_error のすべてのメンバー (継承されたメンバーも含みます) を次に示します。

変数

変更子とタイプ	変数	説明
public ss_error_code	stream_error_code	特定のストリームエラー。 取り得る値のリストについては、`%SQLANY17%¥SDK¥Include¥sserror.h` 列挙を参照してください。
public asa_int32	system_error_code	システム固有のエラーコード。
public char	error_string	system_error_code 値のためのローカライズ済説明がシステムで利用可能な場合、または stream_error_code 値の追加情報が含まれます。

1.77 MLFT_STATUS_FLAG_IS_BLOCKING 変数

ml_file_transfer_status.flags フィールドでビット配列を定義して、Mobile Link サーバからの応答を待機中、ファイル転送はブロックされていることを示します。

構文

```
#define MLFT_STATUS_FLAG_IS_BLOCKING
```

備考

これが発生した場合には、同じファイル転送進行状況メッセージが定期的に生成されます。

2 このマニュアルの印刷、再生、および再配布

次の条件に従うかぎり、このマニュアルの全部または一部を使用、印刷、再生、配布することができます。

1. ここに示したものとそれ以外のすべての版権と商標の表示をすべてのコピーに含めること。
2. マニュアルに変更を加えないこと。
3. SAP 以外の人間がマニュアルの著者または情報源であるかのように示す一切の行為をしないこと。

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼動システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証は行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (http://help.sap.com/disclaimer を参照)。

go.sap.com/registration/contact.html