SYBASE iAnywhere

Mobile Link
クライアント管理

バージョン 12.0.1

2012 年 1 月

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

バージョン 12.0.1
2012 年 1 月

Copyright © 2012 iAnywhere Solutions, Inc. Portions copyright © 2012 Sybase, Inc. All rights reserved.

iAnywhere との間に書面による合意がないかぎり、このマニュアルは現状のまま提供されるものであり、その使用または記載内容の誤りに対して一切の責任を負いません。

次の条件に従うかぎり、このマニュアルの一部または全体を使用、印刷、複製、配布することができます。1) マニュアルの一部または全体にかかわらず、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) iAnywhere 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。

iAnywhere®、Sybase®、http://www.sybase.com/detail?id=1011207 に示す商標は Sybase, Inc. またはその関連会社の商標です。® は米国での登録商標を示します。

このマニュアルに記載されているその他の会社名と製品名は各社の商標である場合があります。

目次

はじめに

このマニュアルでは、Mobile Link クライアントを設定、構成、同期する方法について説明します。Mobile Link クライアントには、SQL Anywhere または Ultra Light のいずれかのデータベースを使用できます。また、Dbmlsync API についても説明します。Dbmlsync API を使用すると、同期を C++ または .NET のクライアントアプリケーションにシームレスに統合できます。

Mobile Link クライアントの概要

この項では、Mobile Link 同期に使用できるクライアントについて紹介するとともに、あらゆる種類の Mobile Link クライアントに共通する情報を示します。

Mobile Link クライアント

Mobile Link サーバーは、現在 2 つのクライアントの使用に対応しています。サポートされている MobiLink クライアントは以下のとおりです。

- SQL Anywhere
- Ultra Light

以下のセクションには、サポートされている MobiLink クライアントに関する情報およびすべての MobiLink クライアントに共通の情報が含まれています。

SQL Anywhere クライアント

SQL Anywhere データベースを Mobile Link クライアントとして使用するには、データベースに同期オブジェクトを追加します。追加する必要があるオブジェクトは、パブリケーション、Mobile Link ユーザー、パブリケーションをユーザーに結び付けるサブスクリプションです。 次の項を参照してください。

- 「リモートデータベースの作成」69 ページ
- 「パブリケーション」73 ページ
- 「Mobile Link ユーザーの作成」82 ページ
- 「同期サブスクリプションの作成」84 ページ

同期は、Dbmlsync API、SQL SYNCHRONIZE 文、または dbmlsync コマンドラインユーティリティを使用して開始できます。ほとんどの同期では、データベーストランザクションログから読み込まれたデータを使用しますが、スクリプト化されたアップロードの同期とダウンロード専用のパブリケーションの同期ではトランザクションログファイルは必要ありません。

「同期の開始」86 ページを参照してください。

SQL Anywhere クライアントの詳細については、「SQL Anywhere クライアント」69 ページを参照してください。

dbmlsync コマンドラインオプションの詳細については、「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」101 ページを参照してください。

同期のカスタマイズの詳細については、「dbmlsync の同期のカスタマイズ」97 ページを参照してください。

Ultra Light クライアント

Ultra Light アプリケーションは、アプリケーションに適切な同期機能の呼び出しが含まれていると自動的に Mobile Link が有効になります。

Ultra Light のアプリケーションとライブラリは、アプリケーション側での同期アクションを処理します。Ultra Light アプリケーションは、同期をほとんど考慮しないで記述できます。Ultra Light ランタイムは、前回の同期以後に加えられた変更を追跡します。

TCP/IP、HTTP、HTTPS、または Microsoft ActiveSync を使用している場合には、同期関数を 1 回呼び出すと、アプリケーションから同期が開始されます。

参照

- 「Ultra Light クライアント」2 ページ
- 「Windows Mobile での Ultra Light と ActiveSync」『Ultra Light データベース管理とリファレンス』
- 「Ultra Light 同期パラメーターとネットワークプロトコルオプション」『Ultra Light データベース管理とリファレンス』
- Ultra Light データベース管理とリファレンス

クライアントのネットワークプロトコル

Mobile Link サーバーでは、-x コマンドラインオプションを使用して、同期クライアントが Mobile Link サーバーに接続するための 1 つ以上のネットワークプロトコルを指定します。クライアントが使用する同期プロトコルと一致するネットワークプロトコルを選択してください。

mlsrv12 コマンドラインオプションの構文は次のとおりです。

mlsrv12 -c "*connection-string***" -x** *protocol***(** *options* **)**

次の例では、TCP/IP プロトコルが選択されますが、プロトコルオプションが指定されていません。

```
mlsrv12 -c "DSN=SQL Anywhere 12 Demo" -x tcpip
```

プロトコルは、次の形式のオプションを使用して設定できます。

(*keyword***=***value***;...)**

次に例を示します。

```
mlsrv12 -c "DSN=SQL Anywhere 12 Demo" -x tcpip(
  host=localhost;port=2439)
```

参照

Mobile Link ネットワークプロトコルオプションの詳細については、次の表を参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

項目	参照先
Mobile Link サーバーのネットワークオプションの設定方法	「-x mlsrv12 オプション」『Mobile Link サーバー管理』
Mobile Link クライアントアプリケーションで使用可能なすべてのネットワークプロトコルオプション	「Mobile Link クライアントネットワークプロトコルオプション」
SQL Anywhere クライアントのオプションの設定方法	「CommunicationAddress (adr) 拡張オプション」 「CommunicationType (ctp) 拡張オプション」
Ultra Light クライアントのオプションの設定方法	「Stream Parameters 同期パラメーター」『Ultra Light データベース管理とリファレンス』 「Stream Type 同期パラメーター」『Ultra Light データベース管理とリファレンス』 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light データベース管理とリファレンス』

Mobile Link のシステムテーブル

Mobile Link サーバーのシステムテーブル

統合データベースとして使用するデータベースを設定すると、Mobile Link サーバーに必要な Mobile Link システムテーブルが作成されます。

「Mobile Link サーバーのシステムテーブル」『Mobile Link サーバー管理』を参照してください。

Ultra Light のシステムテーブル

Ultra Light データベースのスキーマは独自フォーマットで格納されます。

Ultra Light のシステムテーブルの詳細については、「Ultra Light のシステムテーブル」『Ultra Light データベース管理とリファレンス』を参照してください。

SQL Anywhere システムテーブル

SQL Anywhere システムテーブルは直接アクセスできませんが、システムビューを使用してアクセスできます。 「システムビュー」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

次の SQL Anywhere システムビューは、Mobile Link のユーザーにとって特に関心のあるものです。

- 「SYSSYNC システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSPUBLICATION システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSSUBSCRIPTION システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSSYNCSCRIPT システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSSYNCPROFILE システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSARTICLE システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSARTICLECOL システムビュー」『SQL Anywhere サーバー SQL リファレンス』

SQL Anywhere は、システムビューに対してクエリを実行して必要な情報を提供する統合ビューも備えています。 「統合ビュー」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

Mobile Link ユーザー

「Mobile Link ユーザー」とは、Mobile Link サーバーに接続するときに認証に使用される名前で、「同期ユーザー」とも呼ばれます。

ユーザーを同期システムに含めるための条件は次のとおりです。

●リモートデータベース上に Mobile Link ユーザー名を作成してください。

●Mobile Link ユーザー名を Mobile Link サーバーに登録してください。

Mobile Link ユーザーの名前とパスワードは、データベースユーザーの名前とパスワードとは異なります。Mobile Link ユーザー名は、リモートデータベースから Mobile Link サーバーへの接続を認証するために使用されます。

カスタム認証スクリプトを使用しないかぎり、Mobile Link ユーザー名では常に大文字と小文字が区別されます。したがって、リモートデータベースで指定されたユーザー名は、大文字と小文字の区別を含め、統合データベースに登録されたユーザー名と正確に一致する必要があります。Mobile Link ユーザー名を定義するときは、次の点に注意してください。

- **Mobile Link 同期システムでは、大文字と小文字の区別を除き、2 つのユーザー名を同じにすることはできません。** たとえば、ユーザー名に 「aA」 または 「Aa」 を指定することはできますが、両方は指定できません。

- **ユーザー名を使用し始めたら、常に同じ大文字／小文字を使用する必要があります。** たとえば、ユーザー 「Aa」 を追加して同期した場合、「Aa」 を使用して同期を継続する必要があります。「aA」 を使用すると、処理に失敗します。

カスタム認証スクリプトを使用する場合、ユーザー名の大文字と小文字の区別はスクリプトで判断されます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

ユーザー名を使用して、Mobile Link サーバーの動作を制御することもできます。そのためには、同期スクリプト内で **username** パラメーターを使用します。 「スクリプトでのリモート ID と Mobile Link ユーザー名」11 ページを参照してください。

Mobile Link ユーザー名は、統合データベースの Mobile Link システムテーブル ml_user の name カラムに格納されます。

Mobile Link ユーザー名は、同期システム内でユニークである必要はありません。セキュリティ上問題がない場合は、各リモートデータベースに同じ Mobile Link ユーザー名を割り当てることもできます。

Ultra Light ユーザー認証

Ultra Light と Mobile Link のユーザー認証スキームは異なりますが、Ultra Light ユーザー ID の値を Mobile Link ユーザー名と共有して簡素化できます。このように簡素化できるのは、Ultra Light アプリケーションを単一ユーザーが使用している場合のみです。

「Ultra Light ユーザー」『Ultra Light データベース管理とリファレンス』を参照してください。

Mobile Link ユーザーの作成と登録

Mobile Link ユーザーをリモートデータベースに作成し、統合データベースに登録します。

リモートデータベースの Mobile Link ユーザーの作成

リモートデータベース側にユーザーを追加する場合、次のオプションがあります。

- SQL Anywhere リモートデータベースの場合は、Sybase Central または CREATE SYNCHRONIZATION USER 文を使用します。

  「Mobile Link ユーザーの作成」82 ページを参照してください。

- Ultra Light リモートデータベースの場合は、User Name と Password 同期パラメーターを設定します。

  「User Name 同期パラメーター」『Ultra Light データベース管理とリファレンス』と「Password 同期パラメーター」『Ultra Light データベース管理とリファレンス』を参照してください。

統合データベースへの Mobile Link ユーザー名の追加

リモートデータベースでユーザー名が作成された後、次の方法のいずれかを使用して、統合データベースにユーザー名を登録できます。

- mluser ユーティリティを使用する。

  「Mobile Link ユーザー認証ユーティリティ (mluser)」『Mobile Link サーバー管理』を参照してください。

- Sybase Central を使用する。

- authenticate_user イベント用または authenticate_user_hashed イベント用のスクリプトを実装する。これらのスクリプトのどちらかを起動すると、Mobile Link サーバーによって、認証が正常に行われるユーザーが自動的に追加されます。

  「authenticate_user 接続イベント」『Mobile Link サーバー管理』または「authenticate_user_hashed 接続イベント」『Mobile Link サーバー管理』を参照してください。

- mlsrv12 で -zu+ コマンドラインオプションを指定する。この場合、最初に同期するときに、統合データベースに追加されていない既存の Mobile Link ユーザーが追加されます。このオプションは、開発時には便利ですが、配備されたアプリケーションへの使用はおすすめできません。

  「-zu mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。

ユーザーの最初のパスワードを設定する

各ユーザーのパスワードは、ユーザー名とともに ml_user テーブルに格納されます。最初のパスワードは、Sybase Central または mluser コマンドラインユーティリティを使用して指定できます。

Sybase Central は、個々のユーザーとパスワードを追加する場合に便利です。mluser ユーティリティは、バッチで追加する場合に便利です。

パスワードがないユーザーを作成した場合、そのユーザーは Mobile Link で認証されず、接続や同期のためにパスワードが必要ありません。

♦ ユーザーの最初の Mobile Link パスワードの設定 (Sybase Central の場合)

1. **[表示]** » **[フォルダー]** を選択します。
2. Mobile Link 12 プラグインを使用して Mobile Link プロジェクトを開き、**[統合データベース]** を展開します。
3. 統合データベースの名前を展開します。
4. **[ユーザー]** をクリックします。
5. **[ファイル]** » **[新規]** » **[ユーザー]** をクリックします。
6. **ユーザー作成ウィザード**の指示に従います。

♦ 最初の Mobile Link パスワードの設定 (コマンドラインの場合)

1. 各行に 1 人分ずつ、ユーザー名とパスワードを空白スペースで区切って入力したファイルを作成します。
2. コマンドプロンプトを開き、mluser コマンドラインユーティリティを実行します。次に例を示します。

```
mluser -c "DSN=my_dsn" -f password-file
```


Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

このコマンドラインでは、-c オプションで、統合データベースへの ODBC 接続を指定します。-f オプションで、ユーザー名とパスワードを含むファイルを指定します。

参照

- 「Mobile Link ユーザー認証ユーティリティ (mluser)」『Mobile Link サーバー管理』

新しいユーザーからの同期

通常、Mobile Link サーバーに接続するには、各 Mobile Link クライアントが有効な Mobile Link ユーザー名とパスワードを指定します。

Mobile Link サーバーの起動時に -zu+ オプションを指定すると、サーバーは未登録のユーザーからの同期要求を受け付けて応答できます。ml_user テーブルにリストされていないユーザーからの要求を受信すると、要求は処理され、ユーザーは ml_user テーブルに追加されます。

Mobile Link クライアントが、現在の ml_user テーブルにないユーザー名で同期を実行した場合に -zu+ を使用すると、Mobile Link はデフォルトで次のアクションを取ります。

- **パスワードを持たない新しいユーザー**　ユーザーがパスワードを入力しない場合、ml_user テーブルに NULL パスワードでユーザー名が追加されます。このユーザーはパスワードなしで同期できます。
- **パスワードを持つ新しいユーザー**　ユーザーがパスワードを入力すると、ユーザー名とパスワードの両方が ml_user テーブルに追加され、Mobile Link システム内で新しいユーザー名が認識されるようになります。これ以降、このユーザーは同期するために同じパスワードの指定が必要になります。
- **新しいパスワードを持つ新しいユーザー**　新しいユーザーは、[新しいパスワード] フィールドまたは [パスワード] フィールドに情報を入力できます。いずれの場合も、新しいパスワード設定が古いパスワード設定に上書きされ、新しいパスワードを使用して、新しいユーザーが Mobile Link システムに追加されます。これ以降、このユーザーは同期するために同じパスワードの指定が必要になります。

未知のユーザーによる同期の防止

デフォルトでは、Mobile Link サーバーは ml_user テーブルに登録されているユーザーのみ認識します。このデフォルト設定には 2 つの利点があります。第 1 に、Mobile Link サーバーへの不正なアクセスによるリスクが減少します。第 2 に、すでに登録されているユーザーが間違ったユーザー名やスペルミスのあるユーザー名で誤って接続するのを防ぐことができます。Mobile Link システムに予期しない動作が発生する危険性があるため、誤って接続するような事態は避けてください。

参照

- 「-zu mlsrv12 オプション」『Mobile Link サーバー管理』

エンドユーザーに対するパスワード入力の要求

Mobile Link サーバーでユーザー認証を無効にするように選択しないかぎり、Mobile Link クライアントから同期を行うすべてのエンドユーザーは、毎回 Mobile Link ユーザー名とパスワードを入力しなければなりません。

♦ エンドユーザーへの Mobile Link パスワードの入力要求

- ユーザー名とパスワードを入力するメカニズムは、Ultra Light クライアントと SQL Anywhere クライアントで異なります。
  - **Ultra Light**　Ultra Light クライアントでは同期時に、同期構造体の password フィールドに有効な値を指定します。組み込みの Mobile Link 同期の場合、有効なパスワードとは、ml_user Mobile Link システムテーブルにある値と一致するものです。

    アプリケーションでは、エンドユーザーに対して Mobile Link ユーザー名とパスワードの入力を要求してから、同期を行ってください。

    「Ultra Light 同期パラメーターとネットワークプロトコルオプション」『Ultra Light データベース管理とリファレンス』を参照してください。
  - **SQL Anywhere**　ユーザーは、-mp オプションを使用して dbmlsync コマンドラインで有効なパスワードを指定するか、MobilinkPwd 拡張オプションを使用して同期サブスクリプションがあるデータベースにパスワードを格納することができます。このようにしないと、dbmlsync 接続ウィンドウでパスワードの指定を要求するプロンプトが表示されます。拡張オプションは、コマンドラインでパスワードを指定するよりも安全です。これは、コマンドラインは、同じコンピューターで実行中の他のプロセスから参照できるからです。

    認証が失敗すると、ユーザー名とパスワードを再入力するように要求されます。

    次の項を参照してください。

    ○ 「-c dbmlsync オプション」 111 ページ
    ○ 「-mp dbmlsync オプション」 120 ページ
    ○ 「MobiLinkPwd (mp) 拡張オプション」 156 ページ

パスワードの変更

Mobile Link では、エンドユーザーが自身のパスワードを変更するメカニズムが用意されています。インターフェイスは、Ultra Light クライアントと SQL Anywhere クライアントで異なります。

♦ エンドユーザーへの Mobile Link パスワードの入力要求

- ユーザー名とパスワードを入力するメカニズムは、Ultra Light クライアントと SQL Anywhere クライアントで異なります。
  - **SQL Anywhere**　dbmlsync コマンドラインまたは dbmlsync 接続ウィンドウ (コマンドラインパラメーターを指定しない場合) で、有効な既存のパスワードと新しいパスワードを入力します。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

「-mp dbmlsync オプション」120 ページと「-mn dbmlsync オプション」120 ページを参照してください。

- **Ultra Light**　アプリケーションでは、同期構造体の password フィールドに既存のパスワードを、new_password フィールドに新しいパスワードを同期時に指定します。

  「Password 同期パラメーター」『Ultra Light データベース管理とリファレンス』と「New Password 同期パラメーター」『Ultra Light データベース管理とリファレンス』を参照してください。

最初のパスワードは、統合データベースサーバーで、または最初の同期で設定できます。「ユーザーの最初のパスワードを設定する」6 ページと「新しいユーザーからの同期」7 ページを参照してください。

パスワードをいったん割り当てると、クライアント側でパスワードを空の文字列にリセットすることはできません。

リモート ID

「リモート ID」により、Mobile Link 同期システムのリモートデータベースがユニークに識別されます。

SQL Anywhere データベースまたは Ultra Light データベースを作成したときは、リモート ID は NULL です。データベースを Mobile Link と同期する場合、Mobile Link クライアントは null のリモート ID がないかどうかをチェックします。null のリモート ID が見つかると、GUID をリモート ID として割り当てます。リモート ID を一度設定すると、手動で変更されないかぎり (リモート ID の手動変更はおすすめしません)、データベースでは同じリモート ID を保持します。

SQL Anywhere リモートデータベースでは、Mobile Link サーバーが、リモート ID とサブスクリプションによって同期の進行状況を追跡します。Ultra Light データベースでは、Mobile Link サーバーがリモート ID とパブリケーションによって同期の進行状況を追跡します。この情報は、ml_subscription システムテーブルに保存されます。リモート ID は、同期ごとに Mobile Link サーバーログにも記録されます。

リモート ID はユニークであることが必要

各リモートデータベースは、リモート ID でユニークに識別される必要があります。このためには、組み込み GUID のリモート ID を使用します。スクリプトやビジネスロジックで読みやすい ID を代わりに使用してリモートデータベースを表す場合は、Mobile Link ユーザー名やユニークな認証パラメーターを使用することを考慮してください。独自のリモート ID を割り当てる必要がある場合は、各リモートデータベースがユニークなリモート ID に割り当てられていることを確認してください。

同じリモート ID が複数の同時同期処理で使用されていると、同期スクリプトやビジネスロジックによっては、破損やデータ損失が生じる可能性があります。同じリモート ID を使用する同時同期処理は、次のいずれかの理由で生じることがあります。

- リモートデータベースが重複するリモート ID に割り当てられている。この場合、重複するリモート ID はユニークなリモート ID に設定する必要があります。

- ネットワークエラーによってクライアントが切断され、すぐに再同期される。元の同期と新しい同期が同時に処理される可能性があります。

いずれの場合も、Mobile Link サーバーは破損やデータ損失の回避を自動的に試みます。これを行うために、Mobile Link サーバーは通常、エラーの同時同期処理のうち 1 つを除くすべてをキャンセルします。

警告
リモート ID を再使用する場合は注意してください。リモート ID を再使用する場合 (たとえば、リモートデータベースを置換またはリストアする場合、置換リモートデータベースの最初の同期前にそのリモート ID の統合データベースで ml_reset_sync_state ストアドプロシージャーを呼び出します。 詳細については、「ml_reset_sync_state システムプロシージャー」『Mobile Link サーバー管理』を参照してください。

参照

- 「Mobile Link ユーザー」 4 ページ
- 「認証パラメーター」『Mobile Link サーバー管理』

Mobile Link リモート ID 名

リモート ID は GUID として作成されますが、意味のある名前に変更することもできます。SQL Anywhere と Ultra Light データベースの場合、リモート ID は、ml_remote_id と呼ばれるプロパティとしてデータベースに保存されます。

SQL Anywhere クライアントについては、「リモート ID の設定」 71 ページを参照してください。

Ultra Light クライアントについては、「Ultra Light ml_remote_id オプション」『Ultra Light データベース管理とリファレンス』を参照してください。

リモート ID を手動で設定し、その後リモートデータベースを再作成した場合は、再作成したリモートデータベースに古いリモートデータベースとは異なる名前を付けるか、ml_reset_sync_state ストアドプロシージャーを使用して、統合データベース内でリモートデータベースのステータス情報をリセットします。 「ml_reset_sync_state システムプロシージャー」『Mobile Link サーバー管理』を参照してください。

スターターデータベースを複数のロケーションに配備する場合は、リモート ID に NULL が設定されているデータベースを配備するのが最も安全です。事前に移植するようにデータベースを同期した場合は、配備前にリモート ID を NULL に設定し直すことができます。この方法では、リモートデータベースが初めて同期したときにユニークなリモート ID が割り当てられるため、リモート ID の重複を確実に避けることができます。また、リモート ID はリモートデータベースセットアップ手順として設定できますが、ユニークでなければなりません。

例

リモートデータベースあたり 1 ユーザーに Mobile Link 設定を定義する場合の管理作業を簡素化するには、各リモートデータベースで Mobile Link の 3 つの識別子すべてに同じ番号を使用することが必要な場合があります。たとえば、SQL Anywhere リモートデータベースでは、次のように設定できます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
-- Set the MobiLink user name:
  CREATE SYNCHRONIZATION USER "1" ... ;

-- Set the partition number for DEFAULT GLOBAL AUTOINCREMENT:
  SET OPTION PUBLIC.GLOBAL_DATABASE_ID = '1';

-- Set the MobiLink remote ID:
  SET OPTION PUBLIC.ml_remote_id = '1';
```

スクリプトでのリモート ID と Mobile Link ユーザー名

Mobile Link ユーザー名はユーザーを識別し、認証に使用されます。リモート ID は Mobile Link リモートデータベースをユニークに識別します。

多くの同期スクリプトでは、リモートデータベースの識別にオプションでリモート ID (名前付きパラメーター s.remote_id) または Mobile Link ユーザー名 (s.remote_id) を使用できます。リモート ID を使用するといくつかの利点があります (特に Ultra Light の場合)。

リモートデータベースと Mobile Link ユーザーが 1 対 1 の関係になるように配備する場合は、リモート ID を無視できます。この場合、Mobile Link イベントスクリプトでは、username パラメーターを参照できます。このパラメーターは、認証に使用する Mobile Link ユーザー名です。

Mobile Link ユーザーが別のデータベースでデータを同期し、各リモートデータベースのデータが同じ場合は、同期スクリプトは、Mobile Link ユーザー名を参照できます。Mobile Link ユーザーが別のデータベースで別のデータセットを同期する場合は、同期スクリプトは、リモート ID を参照する必要があります。

Mobile Link サーバーはリモート ID によって同期の進行状況を追跡するため、Ultra Light データベースでは、前回のアップロード状態が不明な場合でも、異なるユーザーによって同じデータベースを同期できます。この場合、別のユーザーごとのローの一部は失われてダウンロードされることがない可能性があるため、タイムスタンプベースのダウンロードスクリプトでは Mobile Link ユーザー名は参照できなくなります。これを防ぐには、同じリモートデータベースを使用して、統合データベースのテーブルを各ユーザーのローにマッピングする必要があります。現在の同期に対するリモート ID に基づいたテーブルのマッピングと統合テーブルのジョインによって、すべてのユーザーのすべてのデータを確実にダウンロードできます。

別のスクリプトバージョンを使用して、異なるデータを別のリモートデータベースに同期することもできます。

参照

- 「スクリプトバージョン」『Mobile Link サーバー管理』

ユーザー認証メカニズム

ユーザーの認証は、データを保護するためのセキュリティシステムの一部です。

Mobile Link では、ユーザー認証メカニズムを選択することができます。インストール環境全体にわたって 1 つのメカニズムを使用する必要はありません。Mobile Link には、インストール環境内の各スクリプトバージョンが異なる認証メカニズムを使用できるという柔軟性があります。

- **Mobile Link ユーザー認証なし**　パスワード保護が必要でないデータの場合は、インストール環境でユーザー認証を使用しないように選択できます。この場合、Mobile Link ユーザー名は ml_user テーブルに含まれていなければなりませんが、hashed_password カラムは NULL です。
- **組み込みの Mobile Link ユーザー認証**　Mobile Link では、ml_user Mobile Link システムテーブルに格納されているユーザー名とパスワードを使用して、認証が実行されます。

  組み込みのメカニズムについては、次の項で説明します。
- **カスタム認証**　Mobile Link スクリプトの authenticate_user を使用して、組み込みの Mobile Link ユーザー認証システムを、独自の別のシステムに置き換えることができます。たとえば、使用する統合データベースの管理システムによって、Mobile Link システムの代わりにデータベースのユーザー認証を使用できます。

  「カスタムユーザー認証」 15 ページを参照してください。

Mobile Link と関連製品の他のセキュリティ関連機能については、次の項を参照してください。

- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- Ultra Light クライアント：「Ultra Light データベースのセキュリティ」『Ultra Light データベース管理とリファレンス』
- SQL Anywhere クライアント：「データのセキュリティ」『SQL Anywhere サーバー データベース管理』

ユーザー認証アーキテクチャー

Mobile Link のユーザー認証システムは、ユーザー名とパスワードに依存します。組み込みのメカニズムを使用して、Mobile Link サーバーに対してユーザー名とパスワードを認証させることも、独自のカスタムユーザー認証メカニズムを実装することもできます。

組み込みの認証システムでは、ユーザー名とパスワードの両方が、統合データベース内の ml_user Mobile Link システムテーブルに格納されます。パスワードは、ハッシュされた状態で格納されます。これは、Mobile Link サーバー以外のアプリケーションからは、ml_user テーブルを読み込んでオリジナルのフォームのパスワードを再構成できないようにするためです。統合データベースにユーザー名とパスワードを追加するには、Sybase Central または mluser ユーティリティを使用するか、Mobile Link サーバーの起動時に -zu+ オプションを指定します。

「Mobile Link ユーザーの作成と登録」 5 ページを参照してください。

Mobile Link クライアントは、Mobile Link サーバーに接続するときに、次の値を提供します。

- **ユーザー名**　Mobile Link ユーザー名。必須です。同期を実行するには、ユーザー名を ml_user システムテーブルに格納する必要があります。または、Mobile Link サーバーを -zu+

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプションを使用して起動し、ml_user テーブルに新しいユーザーを追加する必要があります。

- **パスワード**　Mobile Link パスワード。この値は、ユーザーが不明な場合、または ml_user Mobile Link システムテーブル内の対応するパスワードが NULL の場合にのみオプションになります。
- **新しいパスワード**　新しい Mobile Link パスワード。この値はオプションであり、Mobile Link ユーザーはこの値を設定することでパスワードを変更できます。

カスタム認証

オプションで、ユーザー認証メカニズムを独自のものに置き換えることができます。

「カスタムユーザー認証」15 ページを参照してください。

認証処理

次に、認証中に発生するイベントの順序を説明します。

1. リモートアプリケーションは、リモート ID、Mobile Link ユーザー名を使用し、オプションでパスワードと新しいパスワードを使用して、同期要求を開始します。Mobile Link サーバーは新しいトランザクションを開始し、begin_connection_autocommit と begin_connection イベントをトリガーします。
2. Mobile Link は、リモート ID が現在同期を実行中ではなく、authentication_status が 4000 に設定されていることを確認します。
3. authenticate_user スクリプトを定義している場合は、次のイベントが発生します。

   a. authenticate_user スクリプトを SQL で作成した場合、このスクリプトは authentication_status が 4000 に事前に設定され、Mobile Link ユーザー名が指定されて、オプションでパスワードと新しいパスワードが使用されます。

      authenticate_user スクリプトを Java または .NET で作成した SQL 文が返された場合、この SQL 文は 事前設定の authentication_status 4000、作成した Mobile Link ユーザー名とオプションでパスワードと新しいパスワードを使用して呼び出されます。

   b. authenticate_user スクリプトで例外が発生したり、スクリプトを実行したときにエラーが発生した場合、同期処理は停止します。

   authenticate_user スクリプトまたは返された SQL 文は、2 ～ 4 つの引数を取るストアドプロシージャーの呼び出しでなければいけません。事前に設定した authentication_status 値が最初のパラメーターとして渡され、このストアドプロシージャーによって更新されます。最初のパラメーターで返された値は、authenticate_user スクリプトからの authentication_status です。
4. authenticate_user_hashed スクリプトが存在すれば、次のイベントが発生します。

a. パスワードが指定されている場合、ハッシュされた値が計算されます。新しいパスワードが指定されている場合、ハッシュされた値が計算されます。

b. authenticate_user_hashed スクリプトが、authentication_status の現在値 (authenticate_user が存在しない場合は事前に設定された authentication_status、または authenticate_user スクリプトから返された authentication_status) とハッシュされたパスワードで呼び出されます。動作は、手順 3 と同じです。最初のパラメーターで返された値は、authenticate_user_hashed スクリプトの authentication_status として使用されます。

5. Mobile Link サーバーは、authenticate_user スクリプトと authenticate_user_hashed スクリプトが存在する場合は、より大きい値を使用し、どちらのスクリプトも存在しない場合は、事前に設定された authentication_status を使用します。

6. Mobile Link サーバーは、指定された Mobile Link ユーザー名を ml_user テーブルで問い合わせます。

a. カスタムスクリプト authenticate_user または authenticate_user_hashed のいずれかが呼び出されますが、指定された Mobile Link ユーザー名が ml_user テーブルになく、authentication_status が有効な場合 (1000 または 2000) は、Mobile Link ユーザー名が Mobile Link システムテーブルの ml_user に追加されます。authentication_status が有効でない場合、ml_user は更新されず、エラーが発生します。

b. カスタムスクリプトが呼び出されず、指定された Mobile Link ユーザー名も ml_user テーブルにない場合は、Mobile Link サーバーを -zu+ オプションで起動していれば、指定した Mobile Link ユーザー名が ml_user に追加されます。それ以外の場合は、エラーが発生し、authentication_status が無効に設定されます。

c. カスタムスクリプトが呼び出され、指定された Mobile Link ユーザー名が ml_user テーブルに存在する場合は、何も実行されません。

d. カスタムスクリプトが「呼び出されず」、指定された Mobile Link ユーザー名が ml_user テーブルに存在する場合、ml_user テーブルにある値に対してパスワードがチェックされます。パスワードが Mobile Link ユーザー ml_user テーブルにある値と一致する場合、authentication_status は有効に設定されます。一致しない場合、authentication_status は無効に設定されます。

7. authentication_status が有効で、authenticate_user と authenticate_user_hashed のいずれのスクリプトも呼び出されなかった場合に、この Mobile Link ユーザーの ml_user テーブルで新しいパスワードを指定すると、パスワードが指定したものに変更されます。

8. authenticate_parameters スクリプトを定義しており、authentication_status が有効である場合 (1000 または 2000)、次のイベントが発生します。

a. パラメーターが authenticate_parameters スクリプトに渡されます。

b. authenticate_parameters スクリプトが現在の authentication_status より大きい authentication_status 値を返す場合、新しい authentication_status は古い値を上書きします。

9. authentication_status が有効でない場合、同期はアボートされます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

10. modify_user スクリプトを定義している場合はそのスクリプトが呼び出され、指定していた Mobile Link ユーザー名が、このスクリプトによって返された新しい Mobile Link ユーザー名に置き換えられます。

11. Mobile Link サーバーは、authentication_status に関係なく、Mobile Link ユーザー認証後に必ずトランザクションをコミットします。authentication_status が有効な場合 (1000 または 2000)、同期は継続されます。authentication_status が有効でない場合、同期はアボートされます。

カスタムユーザー認証

組み込みの Mobile Link メカニズム以外のユーザー認証メカニズムを使用するように選択することもできます。カスタムユーザー認証メカニズムを使用する理由には、たとえば、次のものがあります。

●既存のデータベースユーザー認証スキームまたは外部認証メカニズムとの統合を含めるため。

●組み込みの Mobile Link メカニズムにはない、パスワードの最小長や有効期限などのカスタム機能を提供するため。

次の 3 つのカスタム認証ツールがあります。

●mlsrv12 -zu+ オプション

●authenticate_user スクリプトまたは authenticate_user_hashed スクリプト

●authenticate_parameters スクリプト

mlsrv12 -zu+ オプションを使用すると、ユーザーの自動追加処理を制御できます。たとえば、-zu + オプションを指定すると、認識されなかった Mobile Link ユーザー名が最初の同期時に ml_user テーブルに追加されます。-zu+ オプションを必要とするのは、組み込みの Mobile Link 認証だけです。

authenticate_user スクリプト、authenticate_user_hashed スクリプト、authenticate_parameters スクリプトは、デフォルトの Mobile Link ユーザー認証メカニズムを無効にします。正常に認証が行われるユーザーは、自動的に ml_user テーブルに追加されます。

authenticate_user スクリプトを使用して、ユーザー ID とパスワードのカスタム認証を作成できます。このスクリプトがあると、組み込みのパスワード比較の代わりにそのメカニズムが実行されます。このスクリプトでは、認証の成功または失敗を示すエラーコードを返さなければなりません。

Mobile Link は、authenticate_user イベント用に事前に定義されたスクリプトをいくつかインストールします。これらのスクリプトは、LDAP、POP3、IMAP サーバーを使用した認証を簡単に行えるようにします。 「外部サーバーに対する認証」16 ページを参照してください。

ユーザー ID とパスワード以外の値によるカスタム認証を作成するには、authenticate_parameters を使用します。

参照

- 「-zu mlsrv12 オプション」『Mobile Link サーバー管理』
- 「authenticate_user 接続イベント」『Mobile Link サーバー管理』
- 「authenticate_user_hashed 接続イベント」『Mobile Link サーバー管理』
- 「authenticate_parameters 接続イベント」『Mobile Link サーバー管理』

Java と .NET のユーザー認証

Java クラスと .NET クラスではアプリケーションサーバーなどのコンピューティング環境で使用されるユーザー名とパスワードの他のソースにアクセスできるため、ユーザー認証は Java と .NET の同期論理で本来使用されているものです。

%SQLANYSAMP12%¥MobiLink¥JavaAuthentication ディレクトリに簡単な例があります。*%SQLANYSAMP12%¥MobiLink¥JavaAuthentication¥CustEmpScripts.java* のサンプルコードは簡単なユーザー認証システムを実装します。最初の同期時に、Mobile Link ユーザー名が login_added テーブルに追加されます。それ以降の同期時には、login_audit テーブルにローが 1 つ追加されます。このサンプルでは、login_added テーブルにユーザー ID を追加する前のテストは行いません。

ユーザー認証を説明する .NET サンプルについては、「.NET 同期のサンプル」『Mobile Link サーバー管理』を参照してください。

外部サーバーに対する認証

Mobile Link には、authenticate_user イベントを使用して外部サーバーに対する認証を簡単に行えるようにする、事前に定義された Java 同期スクリプトが用意されています。事前に定義されたスクリプトは、次の認証サーバーで使用できます。

- JavaMail 1.2 API を使用している POP3 または IMAP サーバー
- Java Naming と Directory Interface (JNDI) を使用している LDAP サーバー

これらのスクリプトをどのように使用するかは、Mobile Link ユーザー名を外部認証システムのユーザー ID に直接マッピングしているかどうかによって決まります。

> **注意**
> Sybase Central で [認証] タブを使用して外部サーバーへの認証を設定することもできます。「Sybase Central の Mobile Link プラグイン」『Mobile Link クイックスタート』を参照してください。

Mobile Link ユーザー名をユーザー ID に直接マッピングしている場合

Mobile Link ユーザー名を認証システムの有効なユーザー ID に直接マッピングしている単純なケースでは、authenticate_user 接続イベントに対する応答で、定義済みのスクリプトを直接使用できます。認証コードは、ml_property テーブルに格納されているプロパティに基づいてそのコード自体を初期化します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

♦ **事前に定義されたスクリプトの authenticate_user での直接使用**

1. 事前に定義された Java 同期スクリプトを Mobile Link の ml_scripts システムテーブルに追加します。追加するには、ストアドプロシージャーを使用するか、Sybase Central を使用します。

   ●ml_add_java_connection_script ストアドプロシージャーを使用するには、次のコマンドを実行します。

   ```
   call ml_add_java_connection_script(
     'MyVersion',
     'authenticate_user',
     'ianywhere.ml.authentication.ServerType.authenticate' )
   ```

   ここで、*MyVersion* はスクリプトバージョンの名前で、*ServerType* は **LDAP**、**POP3**、または **IMAP** です。

   ●Sybase Central の**接続スクリプト追加ウィザード**を使用するには、スクリプトタイプとして **authenticate_user** を選択し、コードエディターで次のコードを入力します。

   ```
   ianywhere.ml.authentication.ServerType.authenticate
   ```

   ここで、*ServerType* は **LDAP**、**POP3**、または **IMAP** です。

   「ml_add_java_connection_script システムプロシージャー」『Mobile Link サーバー管理』を参照してください。

2. この認証サーバーのプロパティを追加します。

   設定が必要な各プロパティに対し、ml_add_property ストアドプロシージャーを使用します。

   ```
   call ml_add_property(
     'ScriptVersion',
     'MyVersion',
     'property_name',
     'property_value' )
   ```

   ここで、*MyVersion* はスクリプトバージョンの名前で、*property_name* は認証サーバーによって決まります。また、*property_value* は使用しているアプリケーションに適切な値です。この呼び出しを、設定の必要な各プロパティに対して繰り返し行います。

   「外部認証識別符号プロパティ」18 ページと「ml_add_property システムプロシージャー」『Mobile Link サーバー管理』を参照してください。

Mobile Link ユーザー名をユーザー ID に直接マッピングしていない場合

Mobile Link ユーザー名がユーザー ID と同じでない場合は、コードを間接的に呼び出す必要があり、ユーザー ID を ml_user 値から抽出するか、マッピングしなければなりません。これを行うには、Java クラスを作成します。

「Java による同期スクリプトの作成」『Mobile Link サーバー管理』を参照してください。

次に、簡単な例を示します。extractUserID メソッド内のコードは、ml_user 値をユーザー ID にマッピングする方法によって異なるため、この例では省きます。すべての作業は、認証クラスの "authenticate" メソッドで行われます。

```
package com.mycompany.mycode;

import ianywhere.ml.authentication.*;
import ianywhere.ml.script.*;

public class MLEvents
{
   private DBConnectionContext _context;
   private POP3 _pop3;

   public MLEvents( DBConnectionContext context )
   {
      _context = context;
      _pop3 = new POP3( context );
   }

   public void authenticateUser(
     InOutInteger status,
     String userID,
     String password,
     String newPassword )
   {
      String realUserID = extractUserID( userID );
      _pop3.authenticate( status, realUserID, password, newPassword );
   }

   private String extractUserID( String userID )
   {
      // code here to map ml_user to a "real" POP3 user
   }
}
```

この例では、初期化プロパティを検出できるようにするために、POP3 オブジェクトを DBConnectContext オブジェクトで初期化する必要があります。この方法でオブジェクトを初期化しない場合は、コードにプロパティを設定する必要があります。次に例を示します。

```
POP3 pop3 = new POP3();
pop3.setServerName( "smtp.sybase.com" );
pop3.setServerPort( 25 );
```

これはどのような認証クラスにも適用されますが、プロパティはクラスによって異なります。

外部認証識別符号プロパティ

Mobile Link では、特に LDAP の場合において、可能なかぎり適切なデフォルトを用意しています。設定できるプロパティはさまざまですが、次に基本的なプロパティを説明します。

POP3 認証識別符号

mail.pop3.host	サーバーのホスト名
mail.pop3.port	ポート番号 (デフォルトの 110 を使用する場合は省略可能)

http://java.sun.com/products/javamail/javadocs/com/sun/mail/pop3/package-summary.html を参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

IMAP 認証識別符号

mail.imap.host	サーバーのホスト名
mail.imap.port	ポート番号 (デフォルトの 143 を使用する場合は省略可能)

http://java.sun.com/products/javamail/javadocs/com/sun/mail/imap/package-summary.html を参照してください。

LDAP 認証識別符号

java.naming.provider.url	ldap://ops-yourLocation/dn=sybase,dn=com などの LDAP サーバーの URL

詳細については、JNDI のマニュアルを参照してください。

Mobile Link クライアントユーティリティ

Mobile Link には、次の 2 つのクライアントユーティリティがあります。

- 「Microsoft ActiveSync プロバイダーインストールユーティリティ (mlasinst)」
- 「Mobile Link ファイル転送ユーティリティ (mlfiletransfer)」

次の各項も参照してください。

- Ultra Light ユーティリティ：「Ultra Light ユーティリティ」『Ultra Light データベース管理とリファレンス』 「Ultra Light ユーティリティ」『Ultra Light データベース管理とリファレンス』
- Mobile Link サーバーユーティリティ：「Mobile Link ユーティリティ」『Mobile Link サーバー管理』 「Mobile Link ユーティリティ」『Mobile Link サーバー管理』
- SQL Anywhere のその他のユーティリティ：「データベース管理ユーティリティ」『SQL Anywhere サーバー データベース管理』「データベース管理ユーティリティ」『SQL Anywhere サーバー データベース管理』

Microsoft ActiveSync プロバイダーインストールユーティリティ (mlasinst)

Microsoft ActiveSync (Windows Vista またはそれ以降での名称は Windows Mobile デバイス センター) 用 Mobile Link プロバイダーをインストールするか、Windows Mobile デバイス上で Ultra Light アプリケーションを登録してインストールします。

構文

mlasinst [*options*] [[*src*] *dst name class* [*args*]]

オプション	説明
-d	最初にアプリケーションを無効にします。
-k *path*	デスクトッププロバイダー *mlasdesk.dll* のロケーションを指定します。 デフォルトで、このファイルは *%SQLANY12%¥bin32* にあります。 エンドユーザー (通常は SQL Anywhere 全体がインストールされていないユーザー) は、Mobile Link Microsoft ActiveSync プロバイダーのインストール時に、-k の指定が必要になる場合があります。
-l *filename*	アクティビティログファイルの名前を指定します。
-n	アプリケーションを登録しますが、デバイスにはコピーしません。 このオプションを指定すると、Mobile Link Microsoft ActiveSync プロバイダーのインストールに加えてアプリケーションが登録されますが、アプリケーションはデバイスにコピーはされません。アプリケーションに複数のファイルがある場合 (静的ライブラリではなく Ultra Light ランタイムライブラリ DLL を使用するようにコンパイルされている場合など)、または他の方法でアプリケーションをデバイスにコピーする場合は、このオプションを指定します。
-t *n*	デスクトッププロバイダーがクライアントからの応答を待つ秒数を指定します。この時間が経過すると、タイムアウトします。デフォルトは 30 です。
-u	Microsoft ActiveSync 用 Mobile Link プロバイダーをアンインストールします。 このオプションを指定すると、Mobile Link Microsoft ActiveSync プロバイダー用に登録されたアプリケーションの登録がすべて解除され、Mobile Link Microsoft ActiveSync プロバイダーがアンインストールされます。この操作によってデスクトップコンピューターやデバイスからファイルが削除されることはありません。デバイスがデスクトップに接続されていない場合は、エラーが返されます。
-v *path*	デバイスプロバイダー *mlasdev.dll* のロケーションを指定します。デフォルトでは、このファイルは *%SQLANY12%¥CE* 内のプラットフォーム固有のディレクトリ内で検索されます。 エンドユーザー (通常は SQL Anywhere 全体がインストールされていないユーザー) は、Mobile Link Microsoft ActiveSync プロバイダーのインストール時に、-v の指定が必要になる場合があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

その他のパラメーター	説明
src	アプリケーションをデバイスにコピーするためのソースファイル名とパスを指定します。このパラメーターを指定するのは、アプリケーションを登録してデバイスにコピーする場合のみです。-n オプションを使用する場合は、このパラメーターを指定しないでください。
dst	デバイス上でアプリケーションに使用するコピー先ファイル名とパスを指定します。
name	Microsoft ActiveSync がアプリケーションを参照するときに使用する名前を指定します。
class	アプリケーションの登録済み Windows クラス名を指定します。
args	Microsoft ActiveSync によってアプリケーションが起動されるときにアプリケーションに渡すコマンドライン引数を指定します。

備考

このユーティリティは、Microsoft ActiveSync 用 Mobile Link プロバイダーをインストールします。プロバイダーには、デスクトップ上で実行されるコンポーネント (*mlasdesk.dll*) と、Windows Mobile デバイスに展開されるコンポーネント (*mlasdev.dll*) の両方が含まれています。mlasinst ユーティリティは、デスクトッププロバイダーの現在のロケーションを示すレジストリエントリを作成し、デバイスプロバイダーをデバイスにコピーします。

また、*mlasinst* ユーティリティに追加の引数を指定すると、Ultra Light アプリケーションを Windows Mobile デバイスに登録してインストールできます。さらに、Microsoft ActiveSync ソフトウェアを使用して Ultra Light アプリケーションの登録とインストールを行う方法もあります。

ライセンス要件に応じて、このアプリケーションをデスクトップコンポーネントやデバイスコンポーネントとともにエンドユーザーに提供できる場合があります。この場合、エンドユーザーは、Microsoft ActiveSync で使用できるようにアプリケーションのコピーを作成できます。

Microsoft ActiveSync プロバイダーをインストールするには、リモートデバイスに接続してください。

Microsoft ActiveSync プロバイダーインストールユーティリティの使用方法の詳細は、次の項を参照してください。

- SQL Anywhere： 「Microsoft ActiveSync 用 Mobile Link プロバイダーのインストール」
- Ultra Light： 「Windows Mobile での Ultra Light と ActiveSync」『Ultra Light データベース管理とリファレンス』

例

次のコマンドは、デフォルト引数を使用して、Microsoft ActiveSync 用の Mobile Link プロバイダーをインストールします。アプリケーションの登録は行いません。正常にインストールするには、デバイスをデスクトップマシンに接続してください。

```
mlasinst
```

次のコマンドは、Microsoft ActiveSync 用の Mobile Link プロバイダーをアンインストールします。正常にアンインストールするには、デバイスをデスクトップマシンに接続してください。

```
mlasinst -u
```

次のコマンドは、Microsoft ActiveSync 用の Mobile Link プロバイダーがまだインストールされていない場合はインストールし、アプリケーション *myapp.exe* を登録します。また、*c:¥My Files ¥myapp.exe* ファイルをデバイス上の *¥Program Files¥myapp.exe* にコピーします。-p -x 引数は、Microsoft ActiveSync によって起動されるときの *myapp.exe* に対するコマンドラインオプションです。このコマンドは、1 行に入力してください。

```
mlasinst "C:¥My Files¥myapp.exe" "¥Program Files¥myapp.exe"
  "My Application" MYAPP -p -x
```

参照

- 「Microsoft ActiveSync との同期」 91 ページ
- 「Ultra Light 同期パラメーターとネットワークプロトコルオプション」『Ultra Light データベース管理とリファレンス』

Mobile Link ファイル転送ユーティリティ (mlfiletransfer)

Mobile Link を介してファイルをアップロードまたはダウンロードします。

構文

mlfiletransfer [*options*] *file*

オプション	説明
-ap *param1*, ...	Mobile Link 認証パラメーター。 「認証パラメーター」『Mobile Link サーバー管理』を参照してください。
-g	転送の進行状況を表示します。
-i	前回の実行の部分的な転送を無視します。
-k	リモートを識別するリモートキー。省略可能です。
-lf *filename*	転送するファイルのローカル名。デフォルトでは、サーバーで認識される名前 (*file* など) が使用されます。
-lp *path*	転送するファイルのローカルパス。デフォルトでは、ローカルパスは Windows Mobile ではルートディレクトリ、その他のプラットフォームでは現在のディレクトリになります。
-p *password*	Mobile Link ユーザー名のパスワード。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
-q	クワイエットモード。メッセージを表示しません。
-s	ファイルを Mobile Link にアップロードします。デフォルトはダウンロードです。
-u *username*	Mobile Link ユーザー名。このオプションは必須です。
-v *version*	スクリプトバージョン。このオプションは必須です。
-x *protocol* **(** *options* **)**	*protocol* には、**tcpip**、**tls**、**http**、または **https** のいずれかを指定します。このオプションは必須です。 使用できる protocol-options は、プロトコルによって異なります。 各プロトコルのオプションのリストについては、「Mobile Link クライアントネットワークプロトコルオプション」 24 ページを参照してください。
file	転送するファイルのサーバーでのファイル名。Mobile Link は、ダウンロード時に、-ftr ディレクトリの *username* サブフォルダーでファイルを検索します。このサブフォルダーにファイルが見つからない場合は、-ftr ディレクトリを検索します。ファイルがどちらのディレクトリでも見つからない場合は、エラーが生成されます。 「-ftr mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。 ファイル名には、パスを含めたり、省略記号 (ピリオド 3 つ)、カンマ、スラッシュ (/)、円記号 (¥) を使用したりしないでください。 Mobile Link は、アップロード時に、-ftru mlsrv12 オプションで指定されたディレクトリでファイルを検索します。「-ftru mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。

備考

このユーティリティは、初めてリモートデータベースを作成する場合、リモートデバイスでソフトウェアをアップグレードする必要がある場合などに、ファイルをダウンロードするときに役立ちます。

このユーティリティを使用してファイルをダウンロードするには、-ftr オプションを使用して Mobile Link サーバーを起動する必要があります。-ftr オプションは、転送するファイルのルートディレクトリを作成し、登録されている Mobile Link ユーザーごとにサブフォルダーを作成します。

このユーティリティを使用してファイルをアップロードするには、-ftru オプションを使用して Mobile Link サーバーを起動する必要があります。-ftru オプションは、アップロードするファイルのロケーションを作成します。

mlfiletransfer の代わりに mlagent を使用してもかまいません。 「クライアントデバイスでの Mobile Link エージェントの設定と実行」『Mobile Link サーバー管理』を参照してください。

Ultra Light ユーザーは、Ultra Light ランタイムで MLFileDownload メソッドと MLFileUpload メソッドも使用できます。 「Mobile Link ファイル転送」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「-ftr mlsrv12 オプション」『Mobile Link サーバー管理』
- 「authenticate_file_transfer 接続イベント」『Mobile Link サーバー管理』

例

次のコマンドは、Mobile Link サーバーを CustDB サンプルデータベースに接続します。-ftr %SystemRoot%¥system32 オプションは、要求されたファイルがないかどうか、*Windows¥system32* ディレクトリをモニターし始めるように Mobile Link サーバーに指示します。この例では、Mobile Link サーバーはまず *C:¥Windows¥system32¥mobilink-username* ディレクトリでファイルを探します。ファイルがない場合、*C:¥Windows¥system32* ディレクトリで探します。通常は、ファイルがないかどうか、*Windows¥system32* フォルダーを Mobile Link サーバーにモニターさせる必要はありません。この例では、同じ場所にあるメモ帳ユーティリティを転送できるように、*Windows¥system32* ディレクトリを使用しています。

```
mlsrv12 -c "DSN=SQL Anywhere 12 CustDB" -zu+ -ftr %SystemRoot%¥system32
```

次のコマンドは、mlfiletransfer ユーティリティを実行します。このコマンドによって、Mobile Link サーバーはローカルディレクトリに *notepad.exe* をダウンロードします。

```
MLFileTransfer -u 1 -v "custdb 12.0" -x tcpip notepad.exe
```

Mobile Link クライアントネットワークプロトコルオプション

この項では、Mobile Link クライアントを Mobile Link サーバーに接続するときに使用できるネットワークプロトコルオプションについて説明します。次のように、複数の Mobile Link クライアントユーティリティで Mobile Link クライアントネットワークプロトコルオプションが使用されます。

クライアントネットワークプロトコルオプションを使用するユーティリティ	参照先
dbmlsync	「CommunicationAddress (adr) 拡張オプション」

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

クライアントネットワークプロトコルオプションを使用するユーティリティ	参照先
Ultra Light	「Stream Parameters 同期パラメーター」『Ultra Light データベース管理とリファレンス』 または「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light データベース管理とリファレンス』の -x オプション
Ultra Light J	● 「Ultra Light J 同期ストリームのネットワークプロトコルのオプション」『Ultra Light - Java プログラミング』 ● StreamHTTPParms インターフェイス [Ultra Light J]『Ultra Light - Java プログラミング』 ● StreamHTTPSParms インターフェイス [Ultra Light J]『Ultra Light - Java プログラミング』
Relay Server	「Relay Server 設定ファイル」『Relay Server』
Mobile Link モニター	「Mobile Link モニターの起動」『Mobile Link サーバー管理』
Mobile Link ファイル転送	「Mobile Link ファイル転送ユーティリティ (mlfiletransfer)」
Mobile Link Listener	「Windows デバイス用の Mobile Link Listener ユーティリティ (dblsn)」『Mobile Link サーバー起動同期』の -x オプション
QAnywhere Agent	「-x qaagent オプション」『QAnywhere』

クライアントが使用する同期プロトコルと一致するネットワークプロトコルを選択してください。 Mobile Link サーバーの接続オプションを設定する方法については、「-x mlsrv12 オプション」『Mobile Link サーバー管理』を参照してください。

プロトコルオプション

● **TCP/IP プロトコルオプション** **tcpip** オプションを指定する場合は、オプションで次のプロトコルオプションを指定できます。

TCP/IP プロトコルオプション	詳細の参照先
client_port=*nnnnn*[-*mmmmm*]	「client_port」
compression={**zlib**|**none**}	「compression」
e2ee_type={**rsa**|**ecc**}	「e2ee_type」

TCP/IP プロトコルオプション	詳細の参照先
e2ee_public_key=*file*	「e2ee_public_key」
host=*hostname*	「host」
network_adapter_name=*name*	「network_adapter_name」
network_leave_open={**off**|**on**}	「network_leave_open」
network_name=*name*	「network_name」
port=*portnumber*	「port」
timeout=*seconds*	「timeout」
zlib_download_window_size=*window-bits*	「zlib_download_window_size」
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」

- **セキュリティ付きの TCP/IP プロトコル**　**tls** オプション (TLS セキュリティを使用する TCP/IP) を指定すると、オプションで次のプロトコルオプションを指定できます。

TLS プロトコルオプション	詳細の参照先
certificate_company=*company_name*	「certificate_company」
certificate_name=*name*	「certificate_name」
certificate_unit=*company_unit*	「certificate_unit」
client_port=*nnnnn*[-*mmmmm*]	「client_port」
compression={**zlib**|**none**}	「compression」
e2ee_type={**rsa**|**ecc**}	「e2ee_type」
e2ee_public_key=*file*	「e2ee_public_key」
fips={**y**|**n**}	「fips」
host=*hostname*	「host」
identity=*filename*	「identity」
identity_name=*name*	「identity_name」
identity_password= *password*	「identity_password」

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

TLS プロトコルオプション	詳細の参照先
network_adapter_name=*name*	「network_adapter_name」
network_leave_open={**off**|**on**}	「network_leave_open」
network_name=*name*	「network_name」
port=*portnumber*	「port」
timeout=*seconds*	「timeout」
tls_type={**rsa**|**ecc**}	「tls_type」
trusted_certificate=*filename*	「trusted_certificate」
zlib_download_window_size=*window-bits*	「zlib_download_window_size」
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」

- **HTTP プロトコル**　**http** オプションを指定する場合は、オプションで次のプロトコルオプションを指定できます。

HTTP プロトコルオプション	詳細の参照先
buffer_size=*number*	「buffer_size」
client_port=*nnnnn*[-*mmmmm*]	「client_port」
compression={**zlib**|**none**}	「compression」
custom_header=*header*	「custom_header」
e2ee_type={**rsa**|**ecc**}	「e2ee_type」
e2ee_public_key=*file*	「e2ee_public_key」
http_buffer_responses={**on**|**off**}	「http_buffer_responses」
http_password=*password*	「http_password」
http_proxy_password=*password*	「http_proxy_password」
http_proxy_userid=*userid*	「http_proxy_userid」
http_userid=*userid*	「http_userid」
host=*hostname*	「host」

HTTP プロトコルオプション	詳細の参照先
network_adapter_name=*name*	「network_adapter_name」
network_leave_open={**off**|**on**}	「network_leave_open」
network_name=*name*	「network_name」
persistent={**off**|**on**}	「persistent」
port=*portnumber*	「port」
proxy_host= *proxy-hostname-or-ip*	「proxy_host」
proxy_port=*proxy-portnumber*	「proxy_port」
set_cookie=*cookie-name* = *cookie-value*	「set_cookie」
timeout=*seconds*	「timeout」
url_suffix=*suffix*	「url_suffix」
version=*HTTP-version-number*	「version」
zlib_download_window_size=*window-bits*	「zlib_download_window_size」
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」

- **HTTPS プロトコル**　**https** オプション (RSA 暗号化を使用する HTTP) を指定する場合は、オプションで次のプロトコルオプションを指定できます。

HTTPS プロトコルオプション	詳細の参照先
buffer_size=*number*	「buffer_size」
certificate_company=*company_name*	「certificate_company」
certificate_name=*name*	「certificate_name」
certificate_unit=*company_unit*	「certificate_unit」
client_port=*nnnnn*[**-***mmmmm*]	「client_port」
compression={**zlib**|**none**}	「compression」
custom_header=*header*	「custom_header」
e2ee_type={**rsa**|**ecc**}	「e2ee_type」

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

HTTPS プロトコルオプション	詳細の参照先
e2ee_public_key=*file*	「e2ee_public_key」
fips={**y**|**n**}	「fips」
host=*hostname*	「host」
http_buffer_responses{**off** |**on** }	「http_buffer_responses」
http_password=*password*	「http_password」
http_proxy_password=*password*	「http_proxy_password」
http_proxy_userid=*userid*	「http_proxy_userid」
http_userid=*userid*	「http_userid」
identity=*filename*	「identity」
identity_name=*name*	「identity_name」
identity_password=*password*	「identity_password」
network_adapter_name=*name*	「network_adapter_name」
network_leave_open={**off**|**on**}	「network_leave_open」
network_name=*name*	「network_name」
persistent={**off**|**on**}	「persistent」
port=*portnumber*	「port」
proxy_host=*proxy-hostname-or-ip*	「proxy_host」
proxy_port= *proxy-portnumber*	「proxy_port」
set_cookie=*cookie-name* = *cookie-value*	「set_cookie」
timeout=*seconds*	「timeout」
tls_type={**rsa**|**ecc**}	「tls_type」
trusted_certificate=*filename*	「trusted_certificate」
url_suffix=*suffix*	「url_suffix」
version=*HTTP-version-number*	「version」

HTTPS プロトコルオプション	詳細の参照先
zlib_download_window_size=*window-size*	「zlib_download_window_size」
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

buffer_size

ネットワークに書き込む前にバッファーする最大バイト数を指定します。HTTP と HTTPS では、このバイト数が、HTTP 要求の本文の最大サイズに変換されます。

構文

buffer_size=*bytes*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

- モバイル OS - 16K
- デスクトップ OS - 64K

備考

一般に、HTTP と HTTPS のバッファー サイズが大きくなるほど、HTTP 要求応答のサイクルの数は減りますが、必要なメモリ量は増えます。

単位はバイトです。キロバイトには K、メガバイトには M、ギガバイトには G を指定してください。

最大値は 1 G です。

このオプションは、クライアントからの要求のサイズを制御します。Mobile Link からの応答のサイズは制限しません。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例は、最大バイト数を 32 K に設定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr=buffer_size=32K"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "buffer_size=32K";
```

certificate_company

このオプションを指定した場合、証明書に記されている組織フィールドがこの値と一致する場合にだけ、アプリケーションはサーバー証明書を受け入れます。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

certificate_company=*organization*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

備考

Mobile Link クライアントは認証局が署名した証明書をすべて信頼するため、同じ認証局が他の会社用に発行した証明書も信頼してしまうことがあります。識別方法がないままだと、クライアントは競争相手の Mobile Link サーバーを自分の会社のものだと勘違いし、誤って機密性の高い情報を送信してしまう可能性があります。このオプションによって追加の検証が指定され、証明書の識別情報部分にある組織フィールドが、指定した特定の値と照合されます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「証明書フィールドの確認」『SQL Anywhere サーバー データベース管理』
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「trusted_certificate」58 ページ
- 「certificate_name」32 ページ
- 「certificate_unit」34 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
  -x https(
    identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
    identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "DSN=mydb;UID=DBA;PWD=sql"
  -e "ctp=https;
      adr='trusted_certificate=c:¥%SQLANY12%¥bin32¥rsaroot.crt;
        certificate_name=RSA Server;
        certificate_company=test;
        certificate_unit=test'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
info.stream = "https";
   info.stream_parms =
      "trusted_certificate=¥%SQLANY12%¥bin32¥rsaroot.crt;"
      "certificate_name=RSA Server;"
      "certificate_company=test;"
      "certificate_unit=test";
```

certificate_name

このオプションを指定した場合、証明書に記されている通称フィールドがこの値と一致する場合にだけ、アプリケーションはサーバー証明書を受け入れます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

certificate_name=*common-name*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

備考

Mobile Link クライアントは認証局が署名した証明書をすべて信頼するため、同じ認証局が他の会社用に発行した証明書も信頼してしまうことがあります。識別方法がないままだと、クライアントは競争相手の Mobile Link サーバーを自分の会社のものだと勘違いし、誤って機密性の高い情報を送信してしまう可能性があります。このオプションによって追加の検証が指定され、証明書の識別情報部分にある通称フィールドが、指定した特定の値と照合されます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「証明書フィールドの確認」『SQL Anywhere サーバー データベース管理』
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「trusted_certificate」 58 ページ
- 「certificate_company」 31 ページ
- 「certificate_unit」 34 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
```

```
-x https(
  identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
  identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "DSN=mydb;UID=DBA;PWD=sql"
  -e "ctp=https;
     adr='trusted_certificate=c:¥%SQLANY12%¥bin32¥rsaroot.crt;
       certificate_name=RSA Server"
       certificate_company=test;
       certificate_unit=test'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
info.stream = "https";
   info.stream_parms =
     "trusted_certificate=¥%SQLANY12%¥bin32¥rsaroot.crt;"
     "certificate_name=RSA Server;"
     "certificate_company=test;"
     "certificate_unit=test";
```

certificate_unit

このオプションを指定した場合、証明書に記されている組織単位フィールドがこの値と一致する場合にだけ、アプリケーションはサーバー証明書を受け入れます。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

certificate_unit=*organization-unit*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

備考

Mobile Link クライアントは認証局が署名した証明書をすべて信頼するため、同じ認証局が他の会社用に発行した証明書も信頼してしまうことがあります。識別方法がないままだと、クライアントは競争相手の Mobile Link サーバーを自分の会社のものだと勘違いし、誤って機密性の高い情報を送信してしまう可能性があります。このオプションによって追加の検証が指定され、証明書の識別情報部分にある組織単位フィールドが、指定した特定の値と照合されます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「証明書フィールドの確認」『SQL Anywhere サーバー データベース管理』
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「trusted_certificate」 58 ページ
- 「certificate_company」 31 ページ
- 「certificate_name」 32 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
  -x https(
    identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
    identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "DSN=mydb;UID=DBA;PWD=sql"
  -e "ctp=https;
      adr='trusted_certificate=c:¥%SQLANY12%¥bin32¥rsaroot.crt;
        certificate_name=RSA Server"
        certificate_company=test;
        certificate_unit=test'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
info.stream = "https";
   info.stream_parms =
      "trusted_certificate=¥%SQLANY12%¥bin32¥rsaroot.crt;"
      "certificate_name=RSA Server;"
      "certificate_company=test;"
      "certificate_unit=test";
```

client_port

通信に使用するクライアントポートの範囲を指定します。

構文

client_port=*nnnnn*[**-***mmmmm*]

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

なし。

備考

可能なポート番号の範囲を示す開始値と終了値を指定します。クライアントを特定のポート番号に制限するには、*nnnnn* と *mmmmm* に同じ番号を指定します。値を 1 つだけ指定すると、範囲の上限値は初期値より 100 大きくなり、ポート数の合計は 101 になります。

このオプションは、ファイアウォール内のクライアントがファイアウォール外の Mobile Link サーバーと通信する場合に役立ちます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例は、許容できるクライアントポートとして 10000 台のポート範囲を設定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr=client_port=10000-19999"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "client_port=10000-19999";
```

compression

使用するデータ圧縮の種類を設定します。

構文

compression= { zlib | none }

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

デフォルト

Ultra Light では、デフォルトでは compression は使用されません。

dbmlsync では、デフォルトで zlib compression がオンになっています。

> **警告**
> SQL Anywhere クライアントでは、圧縮をオフにすると、データはまったく難読化されなくなります。セキュリティが問題になる場合は、ストリームを暗号化する必要があります。
>
> 「トランスポートレイヤーセキュリティ」『SQL Anywhere サーバー データベース管理』を参照してください。

備考

zlib compression を使用する場合は、zlib_download_window_size オプションと zlib_upload_window_size オプションを使用して、アップロードとダウンロードの圧縮を設定できます。この 2 つのオプションを使用して、アップロードまたはダウンロードの圧縮をオフにすることもできます。

アプリケーションを静的な Ultra Light ランタイムに直接リンクする場合、ULEnableZlibSyncCompression(sqlca)を呼び出して、zlib 機能をアプリケーションに直接リンクします。それ以外の場合、*mlczlib12.dll* を展開する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「zlib_download_window_size」 62 ページ
- 「zlib_upload_window_size」 63 ページ

例

次のオプションでは、アップロードの圧縮のみを設定できます。アップロードのウィンドウサイズは 9 に設定します。

```
"compression=zlib;zlib_download_window_size=0;zlib_upload_window_size=9"
```

custom_header

カスタム HTTP ヘッダーを指定します。

構文

custom_header=*header*

HTTP ヘッダーの形式は、*header-name*: *header-value* です。

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

カスタム HTTP ヘッダーを指定すると、クライアントは HTTP 要求を送信するごとにそのヘッダーを含めます。複数のカスタムヘッダーを指定するには、セミコロン (;) を区切り文字として使用しながら、custom_header を複数回使用してください。たとえば、**custom_header=***header1***:***value1***; customer_header=***header2***:***value2* のようになります。

カスタムヘッダーは、カスタムヘッダーが必要なサードパーティツールとの対話を同期クライアントが行う場合に便利です。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

一部の HTTP プロキシは、すべての要求に対して特殊なヘッダーを含めること要求します。次の例では、Embedded SQL または C++ の Ultra Light アプリケーション内の値 ProxyUser に MyProxyHdr というカスタム HTTP ヘッダーを設定します。

```
info.stream = "http";
info.stream_parms =
  "host=www.myhost.com;proxy_host=www.myproxy.com;
  custom_header=MyProxyHdr:ProxyUser";
```

e2ee_type

エンドツーエンド暗号化のキー交換に使用する非対称アルゴリズムを指定します。

構文

e2ee_type= { rsa | ecc }

使用可能なプロトコル

TCPIP、TLS、HTTP、HTTPS

デフォルト

RSA

備考

rsa と **ecc** のどちらかにし、サーバーで指定した値と一致させてください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

エンドツーエンド暗号化を有効にするためには、e2ee_public_key オプションが必須です。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「e2ee_public_key」 39 ページ
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「キーペアジェネレーターユーティリティ (createkey)」『SQL Anywhere サーバー データベース管理』

例

次の例は、Ultra Light クライアント用のエンドツーエンド暗号化を示します。

```
info.stream = "https";
info.stream_parms =
"tls_type=rsa;trusted_certificate=rsaroot.crt;e2ee_type=rsa;e2ee_public_key=rsapublic.pem"
```

e2ee_public_key

エンドツーエンド暗号化に使用する、サーバーのパブリックキーまたは X.509 証明書を含むファイルを指定します。

構文

e2ee_public_key=*file*

使用可能なプロトコル

TCPIP、TLS、HTTP、HTTPS

デフォルト

なし。

備考

ファイルは PEM または DER のいずれかでコード化できます。

キータイプは、e2ee_type パラメーターで指定したタイプと一致させてください。

エンドツーエンド暗号化を有効にするためには、このオプションが必須です。

エンドツーエンド暗号化は TLS/HTTPS プロトコルオプション fips と組み合わせて使用することもできます。このオプションは、ECC の使用時にはサポートされません。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「fips」 40 ページ
- 「e2ee_type」 38 ページ
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「キーペアジェネレーターユーティリティ (createkey)」『SQL Anywhere サーバー データベース管理』

例

次の例は、Ultra Light クライアント用のエンドツーエンド暗号化を示します。

```
info.stream = "https";
info.stream_parms =
"tls_type=rsa;trusted_certificate¥=rsaroot.crt;e2ee_type=rsa;e2ee_public_key=rsapublic.pem"
```

fips

TLS 暗号化とエンドツーエンド暗号化に FIPS 認定の暗号化の実装を使用します。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

fips={ y | n }

使用可能なプロトコル

HTTPS、TLS

デフォルト

N

備考

fips オプションでは RSA 暗号化のみがサポートされます。

fips オプションが有効なクライアントは、fips オプションが無効なサーバーに接続できます。逆についても同様です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

このオプションは、エンドツーエンド暗号化と組み合わせて使用できます。fips が **y** に設定されている場合、Mobile Link クライアントは、FIPS 140-2 基準を満たした RSA 実装および AES 実装を使用します。このオプションは、ECC の使用時にはサポートされません。 「e2ee_type」 38 ページと「e2ee_public_key」 39 ページを参照してください。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「tls_type」 57 ページ
- 「e2ee_type」 38 ページ

例

TCP/IP プロトコルの FIPS 認定の RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
  -x tls(
    tls_type=rsa;
    fips=y;
    identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
    identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e
   "CommunicationType=tls;
    CommunicationAddress=
       'tls_type=rsa;
        fips=y;
        trusted_certificate=¥rsaroot.crt;
        certificate_name=RSA Server'"
```

C または C++ の Embedded SQL で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
    info.stream = "tls";
    info.stream_parms =
      "tls_type=rsa;
       fips=y;
       trusted_certificate=¥rsaroot.crt;
       certificate_name=RSA Server";
```

host

Mobile Link サーバーを実行中のコンピューター、または、Web サーバーを介して同期する場合は Web サーバーを実行中のコンピューターのホスト名または IP アドレスを指定します。

構文

host=*hostname-or-ip*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

- Windows Mobile - デフォルト値は、デバイスに Microsoft ActiveSync パートナーシップが設定されているデスクトップコンピューターの IP アドレスです。
- 他のすべてのデバイス - デフォルトは **localhost** です。

備考

Windows Mobile では、localhost を使用しないでください。これはリモートデバイス自体を示します。デフォルト値を使用すると、Windows Mobile デバイスに Microsoft ActiveSync パートナーシップが設定されているデスクトップコンピューター上の Mobile Link サーバーに、Windows Mobile デバイスを接続できます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例では、クライアントはコンピューター myhost のポート 1234 に接続します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='host=myhost;port=1234'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "host=myhost;port=1234";
```

http_buffer_responses

On に設定されている場合、このオプションでは、バイトを受信するとすぐに処理するのではなく、HTTP パケットを Mobile Link からバッファーにストリーミングしてから処理します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

http_buffer_responses={ off | on }

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

Off

備考

余分なメモリオーバーヘッドが必要となるため、この機能は、HTTP 同期の安定性に関する問題を回避する場合にのみ使用してください。特に、Windows Mobile デバイス用の Microsoft ActiveSync プロキシサーバーでは、サーバー側で接続が閉じてから 15 秒以内に読み込まれないデータは破棄されます。Mobile Link クライアントはダウンロードを受信するとすぐに処理するため、割り当てられた 15 秒以内に HTTP パケットの読み込みの完了に失敗する可能性があります。これにより、非永続 HTTP を使用している同期が失敗し、ストリームエラーコードが生成されます。**http_buffer_responses=on** を指定することで、各 HTTP パケットがバッファーに完全に読み込まれてから、クライアントで処理されるため、15 秒のタイムアウトを回避することができます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

http_password

RFC 2617 の基本認証またはダイジェスト認証を使用してサードパーティの HTTP サーバーとゲートウェイに対する認証を行います。

構文

http_password=*password*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

この機能は、RFC 2617 に記述されている基本認証とダイジェスト認証をサポートします。

基本認証ではパスワードはクリアテキストで HTTP ヘッダーに含められますが、HTTPS を使用すると、ヘッダーを暗号化してパスワードを保護できます。ダイジェスト認証では、ヘッダーはクリアテキストでは送信されず、ハッシュされます。

このオプションは、http_userid と併用する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「http_userid」 46 ページ
- 「http_proxy_password」 44 ページ
- 「http_proxy_userid」 45 ページ

例

次に示す Embedded SQL または C++ の Ultra Light アプリケーションの例は、Web サーバーに対する認証のユーザー ID とパスワードを提供します。

```
synch_info.stream = "https";
synch_info.stream_parms = "http_userid=user;http_password=pwd";
```

http_proxy_password

RFC 2617 の基本認証またはダイジェスト認証を使用してサードパーティの HTTP プロキシに対する認証を行います。

構文

http_proxy_password=*password*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

この機能は、RFC 2617 に記述されている基本認証とダイジェスト認証をサポートします。

基本認証では、パスワードはクリアテキストで HTTP ヘッダーに含められ、HTTPS を使用できます。ただし、プロキシに対する最初の接続は HTTP を通して行われるため、このパスワードはクリアテキストです。ダイジェスト認証では、ヘッダーはクリアテキストでは送信されず、ハッシュされます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

このオプションは、http_proxy_userid と併用する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「http_password」 43 ページ
- 「http_userid」 46 ページ
- 「http_proxy_userid」 45 ページ

例

次に示す Embedded SQL または C++ の Ultra Light アプリケーションの例は、Web プロキシに対する認証のユーザー ID とパスワードを提供します。

```
synch_info.stream = "https";
synch_info.stream_parms = "http_proxy_userid=user;http_proxy_password=pwd";
```

http_proxy_userid

RFC 2617 の基本認証またはダイジェスト認証を使用してサードパーティの HTTP プロキシに対する認証を行います。

構文

http_proxy_userid=*userid*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

この機能は、RFC 2617 に記述されている基本認証とダイジェスト認証をサポートします。

基本認証では、パスワードはクリアテキストで HTTP ヘッダーに含められ、HTTPS を使用できます。ただし、プロキシに対する最初の接続は HTTP を通して行われるため、このパスワードはクリアテキストです。ダイジェスト認証では、ヘッダーはクリアテキストでは送信されず、ハッシュされます。

このオプションは、http_proxy_password と併用する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「http_password」 43 ページ
- 「http_userid」 46 ページ
- 「http_proxy_password」 44 ページ

例

次に示す Embedded SQL または C++ の Ultra Light アプリケーションの例は、Web プロキシに対する認証のユーザー ID とパスワードを提供します。

```
synch_info.stream = "https";
synch_info.stream_parms = "http_proxy_userid=user;http_proxy_password=pwd";
```

http_userid

RFC 2617 の基本認証またはダイジェスト認証を使用してサードパーティの HTTP サーバーとゲートウェイに対する認証を行います。

構文

http_userid=*userid*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

この機能は、RFC 2617 に記述されている基本認証とダイジェスト認証をサポートします。

基本認証ではパスワードはクリアテキストで HTTP ヘッダーに含められますが、HTTPS を使用すると、ヘッダーを暗号化してパスワードを保護できます。ダイジェスト認証では、ヘッダーはクリアテキストでは送信されず、ハッシュされます。

このオプションは、http_password と併用する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「http_password」 43 ページ
- 「http_proxy_password」 44 ページ
- 「http_proxy_userid」 45 ページ

例

次に示す Embedded SQL または C++ の Ultra Light アプリケーションの例は、Web サーバーに対する認証のユーザー ID とパスワードを提供します。

```
synch_info.stream = "https";
synch_info.stream_parms = "http_userid=user;http_password=pwd";
```

identity

このオプションを使用すると、サードパーティのサーバーとプロキシに対して Mobile Link クライアントを認証するために、クライアント側の証明書を使用できるようになります。

構文

identity=*filename*

使用可能なプロトコル

TLS、HTTPS

デフォルト

なし。

備考

filename は、クライアントの ID を含むファイルを識別します。ID は、クライアントの証明書、対応するプライベートキー、オプションとして中間の認証局で構成されます。

プライベートキーが暗号化されている場合は、identity_password オプションを使用してパスワードを指定します。

Mobile Link クライアントは、クライアント側の証明書を使用して、Mobile Link に対して直接認証することはできません。クライアント側の証明書は、それを受け入れるように設定されたサードパーティのサーバーとプロキシに対して認証するためにのみ使用でき、クライアントと Mobile Link サーバー間に位置しています。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「identity_password」 48 ページ

identity_name

この機能は、Common Access Card (CAC) のクライアント ID (証明書とプライベートキー) を使用した認証をサポートします。この機能は、Windows Mobile だけでサポートされます。

このパラメーターを使用して、パブリック証明書の通称を指定します。

注意

別途ライセンスが必要な必須コンポーネント　この機能は CAC 認証アドオンの一部であり、別途ライセンスが必要です。「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

identity_name=*name*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

備考

このパラメーターは、FIPS 認定の RSA 暗号化のみと組み合わせて使用できます。

パブリック証明書がデバイスの証明書ストアにインストールされている必要があります。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

identity_password

ID ファイルで検出されたプライベートキーを暗号化するために使用されるパスワードです。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

identity_password=*password*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

備考

このオプションは、ID ファイルにあるプライベートキーが暗号化されている場合にのみ必要です。 「identity」47 ページを参照してください。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

network_adapter_name

MobiLink クライアントが MobiLink への接続に使用するネットワークアダプターの名前を明示的に指定できるようにします。

構文

network_adapter_name=*name*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

なし。

備考

このオプションは、Windows Mobile と Windows デスクトップに対してのみ有効です。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

network_leave_open

network_name を指定すると、オプションとして、同期が終了した後にネットワーク接続を開いたままにするように指定できます。

構文

network_leave_open={ off | on }

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

デフォルトは **off** です。

備考

このオプションを使用するには、network_name を指定する必要があります。

このオプションを on に設定すると、同期が終了した後もネットワーク接続は開いたままになります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「network_name」 50 ページ

例

次の例では、クライアントは、ネットワーク名 MyNetwork を使用し、同期の完了後も接続を開いたままにしておくように指定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='network_name=MyNetwork;network_leave_open=on'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "network_name=MyNetwork;network_leave_open=on";
```

network_name

ネットワークに接続しようとして失敗した場合に開始するネットワーク名を指定します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

network_name=*name*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

なし。

備考

ネットワーク名を指定して、Mobile Link の自動ダイヤル機能を使用できるようにします。これによって、手動でダイヤルすることなく Windows Mobile デバイスまたは Windows デスクトップコンピューターから接続できます。自動ダイヤルとは、Mobile Link サーバーへの接続の 2 回目の試行のことです。クライアントがダイヤルせずに接続しようとして失敗し、network_name を指定すると、自動ダイヤルがアクティブになります。スケジュールと組み合わせて使用すると、リモートデータベースを自動的に同期できます。スケジュールと組み合わせなくても、手動でダイヤルして接続せずに dbmlsync を実行できます。

Windows Mobile では、*name* は、[設定] » [接続] » [接続] のドロップダウンリスト内のネットワークプロファイルである必要があります。インターネットネットワークまたは勤務先ネットワークのデフォルト設定を使用するには、名前をそれぞれキーワード **default_internet** または **default_work** に設定します。

Windows デスクトッププラットフォームでは、*name* は [ネットワークとダイアルアップ接続] 内のネットワークプロファイルである必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「同期のスケジュール」 95 ページ
- 「network_leave_open」 50 ページ

例

次の例では、クライアントは、ネットワーク名 MyNetwork を使用し、同期の完了後も接続を開いたままにしておくように指定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='network_name=MyNetwork;network_leave_open=on'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "network_name=MyNetwork;network_leave_open=on";
```

persistent

同期のすべての HTTP 要求に単一の TCP/IP 接続を使用します。

構文

persistent={ off | on }

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

On

備考

On という値を指定すると、クライアントは同期ですべての HTTP 要求に同じ TCP/IP 接続を使用しようとします。

仲介サーバーが非永続的な接続を要求したり、クライアントで仲介サーバーが永続的な接続に対応していないことが検出されたりすると、残りの同期化セッションで接続が自動的に非永続的な接続にダウングレードされます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

port

Mobile Link サーバーのソケットポート番号を指定します。

構文

port=*port-number*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

TCP/IP のデフォルト値は **2439** です。これは、Mobile Link サーバーの IANA 登録ポート番号です。

HTTP のデフォルト値は **80** です。

HTTPS のデフォルト値は **443** です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

ポート番号は 10 進数で、Mobile Link サーバーが受信するように設定されているポートと一致させます。

Web サーバーを介して同期する場合は、HTTP 要求または HTTPS 要求を受け付ける Web サーバーポートを指定してください。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例では、クライアントはコンピューター myhost のポート 1234 に接続します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='host=myhost;port=1234'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "host=myhost;port=1234";
```

proxy_host

プロキシサーバーのホスト名または IP アドレスを指定します。

構文

proxy_host=*proxy-hostname-or-ip*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

HTTP プロキシを通す場合だけ使用します。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例では、クライアントはコンピューター myproxyhost 上で実行されているプロキシサーバーのポート 1234 に接続します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='proxy_host=myproxyhost;proxy_port=1234'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "proxy_host=myproxyhost;proxy_port=1234";
```

proxy_port

プロキシサーバーのポート番号を指定します。

構文

proxy_port=*proxy-port-number*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

HTTP プロキシを通す場合だけ使用します。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例では、クライアントはコンピューター myproxyhost 上で実行されているプロキシサーバーのポート 1234 に接続します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr='proxy_host=myproxyhost;proxy_port=1234'"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "proxy_host=myproxyhost;proxy_port=1234";
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

set_cookie

同期中に使用される HTTP 要求内に設定するカスタム HTTP cookie を指定します。

構文

set_cookie=*cookie-name***=***cookie-value*,...

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

なし。

備考

カスタム HTTP cookie は、同期クライアントがサードパーティツール (セッションを識別するために cookie を使用する認証ツールなど) と対話をする場合に便利です。たとえば、ユーザーエージェントが Web サーバー、プロキシ、またはゲートウェイに接続し、それ自体の認証を行うシステムが存在するとします。正常に動作する場合、エージェントはサーバーから 1 つ以上の cookie を受け取ります。続いてエージェントは同期を開始し、set_cookie オプションを通してそのセッション cookie を渡します。

複数の名前と値の組み合わせがある場合は、カンマで区切ります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

Embedded SQL または C++ の Ultra Light アプリケーションで 2 つのカスタム HTTP cookie を設定する例を示します。

```
info.stream = "http";
info.stream_parms =
  "host=www.myhost.com;
  set_cookie=MySessionID=12345, enabled=yes;";
```

timeout

クライアントがネットワーク操作が失敗したと判断するまで待機する時間 (秒単位) を指定します。

構文

timeout=*seconds*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

240 秒

備考

指定した時間内で接続、読み取り、書き込みの試行が完了しなかった場合、クライアントは同期に失敗します。

同期全体を通して、クライアントは指定された間隔で活性更新を送信して、クライアントが接続されていることを Mobile Link サーバーに通知します。また、Mobile Link は活性更新を送り返して、サーバーが接続されていることをクライアントに通知します。低速ネットワークでタイムアウトが指定時間よりも遅れることを防ぐために、Mobile Link クライアントは、タイムアウト値の半分が経過するたびに Mobile Link サーバーにキープアライブバイトを送信します。この値を 240 秒に設定すると、キープアライブメッセージは 120 秒ごとに送信されます。

設定するタイムアウトの値が低くなりすぎないように注意します。活性チェックを行うと、接続がアクティブであることを確認するために、Mobile Link サーバーとクライアントは各タイムアウト時間内に通信する必要があるため、ネットワークトラフィックが増加します。ネットワークまたはサーバーの負荷が非常に大きく、タイムアウト時間が非常に短い場合は、Mobile Link サーバーと dbmlsync が、接続がアクティブであることを確認できないため、活性接続が中止されることがあります。活性タイムアウトは、通常 30 秒 以上に設定します。

タイムアウトの最大値は 10 分です。600 秒を超える数を指定することはできますが、600 秒と解釈されます。

値 0 は、タイムアウトが 10 分であることを意味します。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例は、タイムアウトを 300 秒に設定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr=timeout=300"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "timeout=300";
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

tls_type

同期に使用する暗号を解く鍵を指定します。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

tls_type=*algorithm*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

RSA

備考

この同期のための通信はすべて、指定された暗号を使用して暗号化されます。暗号は次のいずれかを指定してください。

- **ecc** 楕円曲線暗号化です。
- **rsa** RSA 暗号化です。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「トランスポートレイヤーセキュリティを使用する MobiLink クライアントの設定」『SQL Anywhere サーバー データベース管理』
- 「fips」 40 ページ
- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「certificate_company」 31 ページ
- 「certificate_name」 32 ページ
- 「certificate_unit」 34 ページ
- 「trusted_certificate」 58 ページ

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

例

TCP/IP プロトコルの RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
  -x tls(
    tls_type=rsa;
    identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
    identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e
   "CommunicationType=tls;
    CommunicationAddress=
     'tls_type=rsa;
      trusted_certificate=¥rsaroot.crt;
      certificate_name=RSA Server'"
```

C または C++ の Embedded SQL で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
info.stream = "tls";
info.stream_parms =
 "tls_type=rsa;
  trusted_certificate=¥rsaroot.crt;
  certificate_name=RSA Server";
```

trusted_certificate

安全な同期に使用される信頼できるルート証明書のリストを含むファイルを指定します。

注意
別途ライセンスが必要な必須コンポーネント

ECC 暗号化と FIPS 認定の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。

「別途ライセンスが必要なコンポーネント」『SQL Anywhere 12 紹介』を参照してください。

構文

trusted_certificate=*filename*

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

TLS 同期ストリームを介して同期が発生すると、Mobile Link サーバーは、その証明書、その証明書に署名したエンティティの証明書など、自己署名ルート証明書に至るまでの証明書をクライアントに送信します。

クライアントは、連鎖が有効で連鎖内のルート証明書が信頼できることを確認します。この機能を使用すると、信頼できるルート証明書を指定できます。

証明書は、次の優先度の規則に従って使用されます。

- Ultra Light クライアントでは、証明書が ulinit または ulload によってデータベースに設定された場合は、それらの証明書が使用されます。
- trusted_certificate パラメーターが指定された場合は、指定されたファイルの証明書が使用され、ulinit または ulload を使用してデータベースに格納された信頼できる証明書は置き換えられます。
- trusted_certificate パラメーターと ulinit または ulload のいずれによっても証明書が指定されておらず、Windows、Windows Mobile、または Android を使用している場合は、証明書はオペレーティングシステムの信頼できる証明書ストアから読み込まれます。この証明書ストアは、Web サーバーを安全に管理するために HTTPS を介して接続するときに、Web ブラウザで使用されます。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「接続パラメーターでの Ultra Light ファイルパスの指定」『Ultra Light データベース管理とリファレンス』
- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「-x mlsrv12 オプション」『Mobile Link サーバー管理』
- 「tls_type」 57 ページ
- 「certificate_company」 31 ページ
- 「certificate_name」 32 ページ
- 「certificate_unit」 34 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバーとクライアント両方での設定が必要です。各コマンドは、1 行に入力する必要があります。

サーバーでは、実装は次のようになります。

```
mlsrv12
  -c "DSN=SQL Anywhere 12 Demo;UID=DBA;PWD=sql"
```

```
-x https(
  identity=c:¥%SQLANY12%¥bin32¥rsaserver.id;
  identity_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "DSN=mydb;UID=DBA;PWD=sql"
  -e "ctp=https;
     adr='trusted_certificate=c:¥%SQLANY12%¥bin32¥rsaroot.crt;
       certificate_name=RSA Server"
       certificate_company=test;
       certificate_unit=test'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
info.stream = "https";
   info.stream_parms =
      "trusted_certificate=¥%SQLANY12%¥bin32¥rsaroot.crt;"
      "certificate_name=RSA Server;"
      "certificate_company=test;"
      "certificate_unit=test";
```

url_suffix

同期中に送信される各 HTTP 要求の 1 行目の URL に追加するサフィックスを指定します。

構文

url_suffix=*suffix*

suffix の構文は、Windows で Microsoft IIS を使用しているか、Linux で Apache を使用しているかによって異なります。

リダイレクター	*suffix* の構文
IIS	Windows の場合のデフォルトは /ias_relay_server/server/rs_server.dll です。
Apache	Linux の場合のデフォルトは /srv/iarelayserver です。

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

デフォルトは / です。

備考

プロキシサーバーまたは Web サーバーを介して同期する場合、Mobile Link サーバーを検索するために url_suffix が必要な場合があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Relay Server を使用してこのオプションを設定する方法については、「Relay Server 設定ファイル」『Relay Server』を参照してください。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「Relay Server 設定ファイル」『Relay Server』

version

同期に使用する HTTP のバージョンを指定します。

構文

version=*HTTP-version-number*

使用可能なプロトコル

- HTTP、HTTPS

デフォルト

デフォルト値は **1.1** です。

備考

このオプションは、HTTP インフラが特定の HTTP バージョンを必要とする場合に便利です。値は、**1.0** または **1.1** を指定します。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

例

次の例は、HTTP バージョンを 1.0 に設定します。

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e "adr=version=1.0"
```

Embedded SQL または C++ で作成された Ultra Light アプリケーションでは、実装は次のようになります。

```
synch_info.stream_parms = "version=1.0";
```

zlib_download_window_size

compression オプションを zlib に設定した場合は、このオプションを使用して、ダウンロードの圧縮ウィンドウサイズを指定します。

構文

zlib_download_window_size=*window-bits*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

モバイル OS - 12

デスクトップ OS - 15

備考

ダウンロード圧縮をオフにするには、*window-bits* を 0 に設定します。それ以外の場合は、ウィンドウサイズは 9 ～ 15 になります。通常、ウィンドウサイズを大きくすると圧縮率を高くすることができますが、必要なメモリが大きくなります。

window-bits は、ウィンドウサイズ (履歴バッファーのサイズ) を底が 2 の対数で指定します。次の式は、各 *window-bits* に対して、クライアントで使用されるメモリ量の算出に使用します。

upload (compress): memory = $2^{(\text{window-bits} + 3)}$

download (decompress): memory = $2^{(\text{window-bits})}$

アプリケーションを静的な Ultra Light ランタイムに直接リンクする場合、ULEnableZlibSyncCompression(sqlca)を呼び出して、zlib 機能をアプリケーションに直接リンクします。それ以外の場合、*mlczlib12.dll* を展開する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「compression」 36 ページ
- 「zlib_upload_window_size」 63 ページ

例

次のオプションでは、アップロードの圧縮のみを設定できます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

"compression=zlib;zlib_download_window_size=0"

zlib_upload_window_size

compression オプションを zlib に設定した場合は、このオプションを使用して、アップロードの圧縮ウィンドウサイズを指定します。

構文

zlib_upload_window_size=*window-bits*

使用可能なプロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

モバイルオペレーティングシステムでは 12、デスクトップオペレーティングシステムでは 15

備考

アップロード圧縮をオフにするには、ウィンドウサイズを 0 に設定します。それ以外の場合は、ウィンドウサイズは 9 ～ 15 になります。通常、ウィンドウサイズを大きくすると圧縮率を高くすることができますが、必要なメモリが大きくなります。

window-bits は、ウィンドウサイズ (履歴バッファーのサイズ) を底が 2 の対数で指定します。次の式は、各 *window-bits* に対して、クライアントで使用されるメモリ量の算出に使用します。

upload (compress): memory = $2^{(window\text{-}size + 3)}$

download (decompress): memory = $2^{(window\text{-}size)}$

アプリケーションを静的な Ultra Light ランタイムに直接リンクする場合、ULEnableZlibSyncCompression(sqlca)を呼び出して、zlib 機能をアプリケーションに直接リンクします。それ以外の場合、*mlczlib12.dll* を展開する必要があります。

dbmlsync を使用してネットワークプロトコルオプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 143 ページを参照してください。

Ultra Light を使用してネットワークプロトコルオプションを設定する方法については、「Ultra Light 同期ストリームのネットワークプロトコルのオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「compression」 36 ページ
- 「zlib_download_window_size」 62 ページ

例

次のオプションでは、ダウンロードの圧縮のみを設定できます。

"compression=zlib;zlib_upload_window_size=0"

リモート MobiLink クライアントでのスキーマの変更

要件の変化に応じて、配備したリモートデータベースのスキーマを変更しなければならない場合があります。最も一般的なスキーマの変更とは、新しいカラムを既存のテーブルに追加する、または新しいテーブルをデータベースに追加することです。

以前は、同期に影響を及ぼすスキーマ変更では、スキーマを変更する直前に正常な同期を実行する必要がありました。この必要はなくなりました。スキーマを変更するには、ScriptVersion 拡張オプションを使用するのではなく、新しい SQL 構文を使用して、同期サブスクリプションでスクリプトバージョンを格納する必要があります。

この機能をサポートする SQL 構文は次のとおりです。

- **CREATE SYNCHRONIZATION SUBSCRIPTION 文** SCRIPT VERSION 句を使用して、同期中に使用するスクリプトバージョンを指定します。
- **ALTER SYNCHRONIZATION SUBSCRIPTION 文** SET SCRIPT VERSION 句を使用して、同期中に使用するスクリプトバージョンを指定します。

参照

- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

スクリプトバージョンとサブスクリプションの関連付け

バージョン 12.0.0 の新機能により、リモートデータベースに対するスキーマ変更の実行処理が大幅に簡略化されました。この機能を使用するには、dbmlsync の ScriptVersion 拡張オプションの使用を停止する必要があります。代わりに、CREATE SYNCHRONIZATION SUBSCRIPTION 文と ALTER SYNCHRONIZATION SUBSCRIPTION 文に追加された新しい句を使用して、スクリプトバージョンを同期サブスクリプションに直接関連付けてください。

新しい構文を使用すると、トランザクションの発生時にサブスクリプションに関連付けられたスクリプトバージョンを使用して、各データベーストランザクションがアップロードされます。これにより、スクリプトバージョンの変更に必要なスキーマ変更を、同期なしで実行できるようになります。

古い ScriptVersion 拡張オプションを使用すると、スクリプトバージョンは同期時にトランザクションに関連付けられます。したがって、スキーマ変更の前に同期が必要になります。

いくつかの同期システムは、スキーマ変更以外の理由により、同期間でサブスクリプションに使用されるスクリプトバージョンの変更に依存しています。このようなシステムは、新しい機能を使用して更新できないことがあります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

以降は、同期サブスクリプションを作成するときに、常に SCRIPT VERSION 句を指定することをおすすめします。既存のサブスクリプションは、以下に示す手順でアップグレードできます。

♦ スクリプトバージョンとサブスクリプションの関連付け

dbmlsync の ScriptVersion 拡張オプションを使用して同期する、**my_sub** という既存のサブスクリプションがある場合は、次の手順でスクリプトバージョンをサブスクリプションに直接関連付けます。

1. **my_sub** の同期に現在使用されているスクリプトバージョンを特定します。これを行う最も簡単な方法は、次のとおりです。
   a. -v+ オプションを既存の dbmlsync コマンドラインに追加し、同期します。
   b. dbmlsync 出力ファイルで、使用中のスクリプトバージョンを識別する行を見つけます。次のような行を探します。

      ```
      Script version: my_script_ver_1
      ```

2. 現在のスクリプトバージョンをサブスクリプションに関連付けます。

   ```
   ALTER SYNCHRONIZATION SUBSCRIPTION <sub_name>
   SET SCRIPT VERSION = <ver>
   ```

   <sub_name> はサブスクリプション名 (この場合は **my_sub**) で、<ver> は手順 1 で特定した現在のスクリプトバージョンです。

   これ以降に発生するトランザクションはすべて、スクリプトバージョンに関連付けられます。

3. 古いオプションを使用して、最後にもう 1 回同期します。これにより、手順 2 を完了する前に発生したトランザクションが、正しいスクリプトバージョンで確実にアップロードされます。

4. このサブスクリプションに対して指定されている ScriptVersion 拡張オプションをすべて削除します。拡張オプションは、dbmlsync コマンドライン、同期プロファイルでは指定でき、リモートデータベースのサブスクリプション、パブリケーション、Mobile Link ユーザーには関連付けることができます。

複数のサブスクリプションが存在するデータベースでは、サブスクリプションごとに上記の手順を繰り返します。

SQL Anywhere リモートデータベースのスキーマのアップグレードの実行

SQL Anywhere リモートデータベースを配備した後に、そのスキーマを変更することができます。

注意
リモートデータベースに他の接続がないことが確実である場合は、ALTER PUBLICATION 文を手動で使用して、新規または変更したテーブルをパブリケーションに追加できます。それ以外の場合は、sp_hook_dbmlsync_schema_upgrade フックを使用して、スキーマをアップグレードしてください。

「sp_hook_dbmlsync_schema_upgrade」237 ページを参照してください。

♦ SQL Anywhere リモートデータベースへのテーブルの追加

1. 関連するテーブルスクリプトを統合データベースに追加します。

   新しいテーブルのないリモートデータベースと、新しいテーブルのあるリモートデータベースには、同じスクリプトバージョンを使用できます。ただし、新しいテーブルが存在することによって既存のテーブルの同期方法が変更される場合は、新しいスクリプトバージョンを作成し、そのスクリプトバージョンで同期されるすべてのテーブルに対して新しいスクリプトを作成する必要があります。

2. 通常の同期を実行します。同期処理が正常に実行されたことを確認してから、次の処理を続行してください。

3. ALTER PUBLICATION 文を使用して、テーブルを追加します。次に例を示します。

   ```
   ALTER PUBLICATION your_pub
     ADD TABLE table_name;
   ```

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。
   「sp_hook_dbmlsync_schema_upgrade」237 ページを参照してください。

   詳細については、「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

4. 同期を実行します。必要な場合は、新しいスクリプトバージョンを使用します。

リモートデータベースのテーブル定義の変更

♦ 配備された SQL Anywhere リモートデータベースのパブリッシュ済みのテーブルの変更

既存テーブルのカラムの数か型を変更する場合は、注意してください。Mobile Link クライアントが新しいスキーマで同期する場合、upload_update や download_cursor などのスクリプトが必要です。これらのスクリプトには、リモートテーブルの全カラム用のパラメーターがあります。古いリモートデータベースの場合は、元のカラムだけを保持するスクリプトが必要です。

1. 統合データベースで、新しいスクリプトバージョンを作成します。

2. 新しいスクリプトバージョンには、古いスクリプトバージョンで同期された変更対象のテーブルを含むパブリケーションのすべてのテーブルに対してスクリプトを作成します。

3. リモートデータベースで、古いスクリプトバージョンを使用して通常の同期を実行します。同期処理が正常に実行されたことを確認してから、次の処理を続行してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

4. リモートデータベースで、ALTER PUBLICATION 文を使用して、テーブルをパブリケーションから一時的に削除します。次に例を示します。

   ```
   ALTER PUBLICATION your_pub
     DROP TABLE table_name;
   ```

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。

5. リモートデータベースで、ALTER TABLE 文を使用してテーブルを変更します。

6. リモートデータベースで、ALTER PUBLICATION 文を使用して、テーブルをパブリケーションに戻します。

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。

7. 新しいスクリプトバージョンを使用して同期します。

参照

- 「スクリプトバージョン」『Mobile Link サーバー管理』
- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「sp_hook_dbmlsync_schema_upgrade」 237 ページ
- 「ALTER TABLE 文」『SQL Anywhere サーバー SQL リファレンス』

Ultra Light リモートデータベースのスキーマのアップグレード

既存のアプリケーションで DDL を実行することで、Ultra Light リモートデータベースのスキーマを変更できます。

- 新しいデータベースで新しいアプリケーションを配備する場合は、Mobile Link サーバーとの同期を行って Ultra Light データベースにデータを再移植する必要があります。
- データベースをアップグレードする DDL を含む新しいアプリケーションを配備する場合、データは保持されます。
- 既存のアプリケーションが、一般的な方法で DDL 文を受信できる場合は、DDL をデータベースに適用することができ、データは保持されます。

全ユーザーが同時にアプリケーションを新しいバージョンにアップグレードすることは、通常、現実的ではありません。したがって、新旧 2 つのバージョンを使用できるようにして、単一の統合データベースとこれらを同期する必要があります。2 種類以上の同期スクリプトを作成し、それらを統合データベースに格納して、Mobile Link サーバーの動作を制御できます。次に、使用するアプリケーションのバージョンごとに、同期の開始時に正しいバージョン名を指定することによって、適切な同期スクリプトのセットを選択できます。

Ultra Light DDL の詳細については、「Ultra Light SQL 文」『Ultra Light データベース管理とリファレンス』を参照してください。

参照

- 「Ultra Light データベーススキーマのアップグレードの配備」『Ultra Light データベース管理とリファレンス』

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Mobile Link 用 SQL Anywhere クライアント

この項では、Mobile Link 同期のために SQL Anywhere クライアントを設定し実行する方法について説明します。

SQL Anywhere クライアント

次の項からは、Mobile Link での SQL Anywhere クライアントの使用に関するトピックについて説明します。

リモートデータベースの作成

SQL Anywhere データベースは、Mobile Link システムでリモートデータベースとして使用できます。必要な作業は、パブリケーション、Mobile Link ユーザー、同期サブスクリプションを作成して、ユーザーを統合データベースに登録することだけです。

注意
トランザクションログのないデータベースは、スクリプト化されたアップロードやダウンロード専用のパブリケーションのリモートデータベースとしてのみ使用できます。

同期モデル作成ウィザードを使用して Mobile Link クライアントアプリケーションを作成した場合、これらのオブジェクトはモデルの配備時に自動的に作成されます。この場合も、概念を理解している必要があります。

♦ **リモートデータベースとしての SQL Anywhere データベースの使用**

1. 既存の SQL Anywhere データベースを指定して起動するか、新しいデータベースを作成してテーブルを追加します。
2. リモートデータベース内で 1 つまたは複数のパブリケーションを作成します。

   「パブリケーション」73 ページを参照してください。
3. リモートデータベースに Mobile Link ユーザーを作成します。

   「Mobile Link ユーザーの作成」82 ページを参照してください。
4. ユーザーを統合データベースに登録します。

   「統合データベースへの Mobile Link ユーザー名の追加」5 ページを参照してください。
5. リモートデータベースで同期サブスクリプションを作成します。

   「同期サブスクリプションの作成」84 ページを参照してください。

リモートデータベースの配備

SQL Anywhere リモートデータベースを配備するには、データベースを作成し、適切なパブリケーションを追加する必要があります。これを行うには、プロトタイプのリモートデータベースをカスタマイズします。

スターターデータベースを複数のロケーションに配備する場合は、リモート ID に NULL が設定されているデータベースを配備するのが最も安全です。事前に移植するようにデータベースを同期した場合は、配備前にリモート ID を NULL に設定し直すことができます。この方法では、リモートデータベースが初めて同期したときにユニークなリモート ID が割り当てられるため、リモート ID の重複を確実に避けることができます。また、リモート ID はリモートセットアップ手順として設定できますが、ユニークでなければなりません。

同期モデル作成ウィザードを使用して Mobile Link クライアントアプリケーションを作成する場合は、ウィザードを使用してデータベースを配備できます。

♦ プロトタイプをカスタマイズすることによる Mobile Link リモートデータベースの配備

1. プロトタイプとなるリモートデータベースを作成します。

   プロトタイプデータベースには、必要なテーブルとパブリケーションをすべて入れますが、各データベースに固有の情報を入れる必要はありません。通常、この情報は次のとおりです。

   - Mobile Link ユーザー名
   - 同期サブスクリプション
   - グローバルオートインクリメントキー値の始点を提供する global_database_id オプション

2. リモートデータベースごとに、次の操作を実行します。

   - リモートデータベースを保持するディレクトリを作成します。
   - そのディレクトリにプロトタイプのリモートデータベースをコピーします。

     トランザクションログがリモートデータベースと同じディレクトリに保持されている場合、ログファイル名を変更する必要はありません。
   - 個々の情報をデータベースに追加する SQL スクリプトを実行します。

     この SQL スクリプトは、パラメーター化されたスクリプトにすることができます。 パラメーター化されたスクリプトの詳細については、「PARAMETERS 文 [Interactive SQL]」『SQL Anywhere サーバー SQL リファレンス』と「SQL スクリプトファイル」『SQL Anywhere サーバー SQL の使用法』を参照してください。

例

次の SQL スクリプトは、Contact の例から抜粋したものです。このスクリプトは、*%SQLANYSAMP12%¥MobiLink¥Contact¥customize.sql* にあります。

```
PARAMETERS ml_userid, db_id;
go
SET OPTION PUBLIC.global_database_id = {db_id}
go
```


Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
CREATE SYNCHRONIZATION USER {ml_userid}
    TYPE 'TCPIP'
    ADDRESS 'host=localhost;port=2439'
go
CREATE SYNCHRONIZATION SUBSCRIPTION TO "DBA"."Product"
    FOR {ml_userid}
go
CREATE SYNCHRONIZATION SUBSCRIPTION TO "DBA"."Contact"
    FOR {ml_userid}
go
commit work
go
```

次のコマンドは、データソース dsn_remote_1 を指定し、リモートデータベースに対してスクリプトを実行します。

```
dbisql -c "DSN=dsn_remote_1" read customize.sql [SSinger] [2]
```

参照

- 「リモート ID の設定」 71 ページ
- 「同期モデルの展開」『Mobile Link クイックスタート』
- 「SQL Anywhere Mobile Link クライアントの配備」『Mobile Link サーバー管理』
- 「最初の同期は常に行われる」 73 ページ

リモート ID の設定

リモート ID により、Mobile Link 同期システムのリモートデータベースがユニークに識別されます。SQL Anywhere データベースを作成すると、リモート ID は NULL に設定されます。データベースを Mobile Link と同期する場合、Mobile Link は NULL のリモート ID をチェックし、該当するリモート ID が見つかると、GUID をリモート ID として割り当てます。リモート ID を一度設定すると、手動で変更しないかぎり、データベースでは同じリモート ID を保持します。

Mobile Link のイベントスクリプトや別のものからリモート ID を参照する場合は、リモート ID にわかりやすい名前を付けることができます。リモート ID を変更するには、リモートデータベースの ml_remote_id データベースオプションを設定します。ml_remote_id オプションはユーザー定義のオプションで、SYSOPTION システムテーブルに格納されます。このオプションを変更するには、SET OPTION 文や Sybase Central の SQL Anywhere 12 プラグインを使用します。

リモート ID は、同期システム内でユニークでなければなりません。

リモート ID を手動で設定し、その後リモートデータベースを再作成した場合は、再作成したリモートデータベースに古いリモートデータベースとは異なる名前を付けるか、ml_reset_sync_state ストアドプロシージャーを使用して、統合データベース内でリモートデータベースのステータス情報をリセットします。

警告
ほとんどの場合、リモート ID を設定したり、その値を知る必要はありません。ただし、リモート ID を変更する必要がある場合、リモート ID を変更するのに最も安全な時は、最初の同期処理の前です。リモート ID を後で変更する場合は、変更する直前に同期処理の実行を完了しておくようにしてください。同期処理を行わないでリモート ID を変更すると、一部のデータが失われ、データベースの整合性が保たれなくなる可能性があります。

参照

- 「ml_reset_sync_state システムプロシージャー」『Mobile Link サーバー管理』
- 「SET OPTION 文」『SQL Anywhere サーバー SQL リファレンス』
- 「データベースオプション」『SQL Anywhere サーバー データベース管理』
- 「SYSOPTION システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「リモート ID」9 ページ

例

次の SQL 文では、リモート ID を HR001 に設定します。

```
SET OPTION PUBLIC.ml_remote_id = 'HR001'
```

リモートデータベースのアップグレード

新しい SQL Anywhere リモートデータベースをインストールして古いバージョンを上書きすると、統合データベース内の同期進行状況情報が正しくなくなります。この問題を解決するには、ml_reset_sync_state ストアドプロシージャーを使用して、統合データベースでリモートデータベースのステータス情報をリセットします。

参照

- 「ml_reset_sync_state システムプロシージャー」『Mobile Link サーバー管理』
- 「SQL Anywhere Mobile Link クライアントのアップグレード」『SQL Anywhere 12 変更点とアップグレード』

進行オフセット

進行オフセットは、サブスクリプションのすべての操作がアップロードおよび確認されたところまでの時点を示す整数値です。dbmlsync ユーティリティは、オフセットを使用してどのデータをアップロードするか決定します。リモートデータベースでは、オフセットは SYS.ISYSSYNC システムテーブルの progress カラムに格納されます。統合データベースでは、オフセットは ml_subscription テーブルの progress カラムに格納されます。

リモートごとに、リモートデータベースと統合データベースが各サブスクリプションに対するオフセットを管理します。Mobile Link ユーザーが同期を行うと、その Mobile Link ユーザーに関連するすべてのサブスクリプションに対してオフセットが確認されます。これは、その時点でサブスクリプションの同期が取られていない場合でも同様です。このように処理されるのは、複数の

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

パブリケーションに同じデータを含めることができるためです。唯一の例外として、dbmlsync は、アップロードを試みるまでサブスクリプションの進行オフセットをチェックしません。

リモートデータベースのオフセットと統合データベースのオフセットが一致しない場合、デフォルトの動作はリモートデータベースのオフセットを統合データベースの値で更新し、そのオフセットに基づいて新しいアップロードを送信します。通常、このデフォルト動作が適切です。たとえば、統合データベースがバックアップからリストアされ、リモートトランザクションログが変更されていないとき、またはアップロードは成功したが通信エラーによりアップロードの確認が送信されないときに、広く有効です。

進行オフセットが一致しない場合の大部分は、統合データベース進行値を使用することで自動的に解決されます。進行オフセットの問題の修正が必要になることがまれにありますが、この場合は dbmlsync -r オプションを使用できます。

最初の同期は常に行われる

新規に作成したサブスクリプションを最初に同期しようとすると、統合データベースの進行オフセットに対するサブスクリプションの進行オフセットはチェックされません。この機能を使用すると、統合データベースで保持されるステータス情報を削除せずに、リモートデータベースを再作成したり同期したりできます。

リモートデータベースシステムテーブル SYS.ISYSSYNC 内の **progress** カラムの値が **created** カラムの値と同じであり、**log_sent** カラムの値が NULL の場合、dbmlsync ユーティリティは最初の同期を検出します。

ただし、同じアップロードで複数のサブスクリプションを同期し、そのサブスクリプションのいずれかが初めての同期でない場合、初めて同期されるサブスクリプションも含めて、同期対象のすべてのサブスクリプションに関して進行オフセットがチェックされます。たとえば、2 つのサブスクリプション (-s sub1、sub2) に関して dbmlsync -s オプションを指定する場合で、sub1 は以前に同期され、sub2 は同期されていないとします。この場合、両方のサブスクリプションの進行オフセットが統合データベースの値に対してチェックされます。

参照

- 「-r dbmlsync オプション」 127 ページ
- 「ISYSSYNC システムテーブル」『SQL Anywhere サーバー SQL リファレンス』
- 「トランザクションログファイル」 89 ページ

パブリケーション

パブリケーションとは、同期されるデータを識別するデータベースオブジェクトです。パブリケーションは、アップロードされるデータを定義し、ダウンロード先になるテーブルを制限します (ダウンロードについては download_cursor スクリプトで定義されます)。

パブリケーションは、1 つまたは複数のアーティクルから成ります。各アーティクルでは、同期するテーブルのサブセットを指定します。サブセットには、テーブル全体またはテーブルのローやカラムのサブセットを指定できます。パブリケーション内の各アーティクルは、異なるテーブルを参照するようにしてください。

パブリケーションをユーザーにリンクするためのサブスクリプションを作成します。

Sybase Central または CREATE PUBLICATION 文を使用して、パブリケーションを作成します。

Sybase Central では、**[パブリケーション]** フォルダーにすべてのパブリケーションとアーティクルがあります。

デフォルトでは、SQL Anywhere クライアントでのトリガーの動作は Mobile Link サーバーと同期されません。それは、データの同期先のバックエンドデータベースに同じトリガーが存在するという想定に基づいています。 この動作の詳細については、「SendTriggers (st) 拡張オプション」 164 ページを参照してください。

パブリケーションについての注意

- パブリケーションの作成と削除には DBA 権限が必要です。
- 同じテーブルの異なるカラムサブセットが含まれた 2 つのパブリケーションを作成することはできません。
- パブリケーションはどのカラムが選択されているかは確認しますが、それらが送信される順序は確認しません。カラムは、CREATE TABLE 文で定義された順に常に送信されます。
- 各アーティクルは、それぞれが参照するテーブルのプライマリキー内のカラムをすべて含んでいる必要があります。
- アーティクルでは、同期するテーブルのカラムを制限できます。WHERE 句を使用して、ローを制限することもできます。
- パブリケーションにビューとストアドプロシージャーを入れることはできません。
- パブリケーションとサブスクリプションは、Sybase のメッセージベースのレプリケーションテクノロジである SQL Remote でも使用されます。SQL Remote の場合は、統合データベースとリモートデータベースの両方にパブリケーションとサブスクリプションが必要です。これに対して、Mobile Link では、パブリケーションは SQL Anywhere リモートデータベースにのみ存在します。Mobile Link 統合データベースは、同期スクリプトを使用して設定されます。

参照

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「SendTriggers (st) 拡張オプション」 164 ページ

テーブル全体のパブリッシュ

テーブル全体のパブリッシュ

作成できる最も簡単なパブリケーションは、アーティクルのセットで構成されます。各アーティクルには 1 つのテーブルのすべてのローとカラムを含めます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

前提条件

パブリッシュされるテーブルは、あらかじめ用意しておく必要があります。

内容と備考

新しく作成するパブリケーションの名前とパブリッシュするテーブルの名前を指定して、CREATE PUBLICATION 文を実行することによって、パブリケーションを作成することもできます。

♦ 完全テーブルを 1 つ以上パブリッシュする (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、DBA 権限のあるユーザーとしてリモートデータベースに接続します。
2. **[パブリケーション]** フォルダーを開きます。
3. **[ファイル]** » **[新規]** » **[パブリケーション]** を選択します。
4. **[新しいパブリケーションの名前を指定してください。]** フィールドに、新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
5. **[次へ]** をクリックします。
6. **[使用可能なテーブル]** リストでテーブルを選択します。**[追加]** をクリックします。
7. **[完了]** をクリックします。

結果

パブリケーションが作成されます。

次の手順

なし。

例

次の文は、Customers テーブル全体をパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
 TABLE Customers
)
```

次の文は、SQL Anywhere のサンプルデータベースから、テーブルセットの各テーブルのすべてのカラムとローを含むパブリケーションを作成します。

```
CREATE PUBLICATION sales (
 TABLE Customers,
 TABLE SalesOrders,
 TABLE SalesOrderItems,
```

```
    TABLE Products
)
```

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

テーブル内の一部のカラムだけをパブリッシュする

Sybase Central では、テーブルのすべてのローと、一部のカラムだけを含むパブリケーションを作成できます。また、CREATE PUBLICATION 文でカラムのリストを指定しても、同様に作成できます。

前提条件

前提条件

内容と備考

注意

- 異なるカラムのサブセットを持つ同じテーブルが含まれたパブリケーションを 2 つ作成した場合は、片方に対してのみ同期サブスクリプションを作成できます。
- アーティクルには、テーブル内のすべてのプライマリキーを含めてください。

CREATE PUBLICATION 文を使用して、テーブル内の一部のカラムのみをパブリッシュするには、CREATE PUBLICATION 文を実行して、パブリケーション名とテーブル名を指定します。テーブル名の後ろにあるカッコの中に、パブリッシュするカラムをリストします。

♦ テーブル内の一部のカラムだけをパブリッシュする (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、DBA 権限のあるユーザーとしてリモートデータベースに接続します。
2. **[パブリケーション]** フォルダーを開きます。
3. **[ファイル]** » **[新規]** » **[パブリケーション]** を選択します。
4. **[新しいパブリケーションの名前を指定してください。]** フィールドに、新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
5. **[次へ]** をクリックします。
6. **[使用可能なテーブル]** リストでテーブルを選択します。**[追加]** をクリックします。
7. **[次へ]** をクリックします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

8. **[使用可能なカラム]** リストで、使用可能なカラムのリストを展開します。カラムを選択し、**[追加]** をクリックします。
9. **[完了]** をクリックします。

結果

選択したテーブルカラムが、パブリッシュされます。

次の手順

なし。

例

次の文は、Customers テーブルのカラムである id、company_name、city のすべてのローをパブリッシュするパブリケーションを作成します。

```
CREATE PUBLICATION pub_customer (
 TABLE Customers (id, company_name,
  city )
)
```

参照

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

テーブル内の一部のローだけをパブリッシュする

アーティクル定義で WHERE 句の指定がないと、テーブル内で変更されたすべてのローがアップロードされます。パブリケーション内のアーティクルに WHERE 句を追加することで、変更されたローのうち WHERE 句の探索条件に一致するローのみをアップロードするよう制限できます。

WHERE 句内の探索条件では、アーティクルに含まれるカラムだけを参照できます。また、WHERE 句では次のいずれも使用できません。

- サブクエリ
- 変数
- 非決定的関数

これらの条件は強制ではありませんが、従わなかった場合、予期しない結果が発生します。WHERE 句に関連するエラーは、パブリケーションの定義時ではなく、その WHERE 句で参照されたテーブルに対して DML が実行されたときに発生します。

♦ WHERE 句を使用したパブリケーションの作成 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、DBA 権限のあるユーザーとしてリモートデータベースに接続します。

2. **[パブリケーション]** フォルダーを開きます。
3. **[ファイル]** » **[新規]** » **[パブリケーション]** を選択します。
4. **[新しいパブリケーションの名前を指定してください。]** フィールドに、新しいパブリケーションの名前を入力します。**[次へ]** をクリックします。
5. **[次へ]** をクリックします。
6. **[使用可能なテーブル]** リストでテーブルを選択します。**[追加]** をクリックします。
7. **[次へ]** をクリックします。
8. **[次へ]** をクリックします。
9. **[アーティクル]** リストでテーブルを選択し、**[選択したアーティクルには次の WHERE 句があります]** ウィンドウ枠で探索条件を入力します。
10. **[完了]** をクリックします。

♦ WHERE 句を使用したパブリケーションの作成 (SQL の場合)

1. DBA 権限のあるユーザーとしてリモートデータベースに接続します。
2. パブリケーション対象のテーブルと WHERE 条件を含む CREATE PUBLICATION 文を実行します。

参照

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

例

次の例は、Employees テーブル全体を含み、SalesOrders テーブルの中でアーカイブ済みとしてマーク付けされていないすべてのローを含むパブリケーションを作成します。

```
CREATE PUBLICATION main_publication (
TABLE Employees,
TABLE SalesOrders
WHERE archived = 'N'
);
```

テーブル内の archived カラムを N 以外の値から N に変更すると、次回の同期中に Mobile Link サーバーに delete が送信されます。逆に、archived カラムを N から N 以外の値に変更すると、insert が送信されます。archived カラムに対する更新は、Mobile Link サーバーに送信**されません**。

ダウンロード専用のパブリケーション

リモートデータベースへのデータのダウンロードのみを行い、データのアップロードは行わないパブリケーションを作成できます。ダウンロード専用のパブリケーションでは、クライアントのトランザクションログを使用しません。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

異なるダウンロード専用方法における相違点

(アップロードを行わず) ダウンロードのみを行うよう指定するには、2 つの方法があります。

- **ダウンロード専用の同期**　dbmlsync オプションの -e DownloadOnly または -ds を使用します。
- **ダウンロード専用のパブリケーション**　FOR DOWNLOAD ONLY キーワードを使用してパブリケーションを作成します。

これらの 2 つの方法には大きな違いがあります。

ダウンロード専用の同期	ダウンロード専用のパブリケーション
リモートデータベースで変更されているがアップロードはされていないローをダウンロード処理で変更しようとした場合、ダウンロードは失敗します。	リモートデータベースで変更されているがアップロードはされていないローをダウンロード処理で上書きできます。
アップロードやダウンロードが可能な通常のパブリケーションを使用します。ダウンロード専用の同期処理は、dbmlsync のコマンドラインオプションまたは拡張オプションを使用して指定します。	ダウンロード専用パブリケーションを使用します。パブリケーションに対するすべての同期処理はダウンロード専用です。通常のパブリケーションをダウンロード専用に変更することはできません。
ログファイルが必要です。	ログファイルは必要ありません。
サブスクリプションが長期間アップロードされない場合、ログファイルはトランケートされず、大量のディスク領域が使用される場合があります。	ログファイルが存在すると、同期処理によって同期トランケーションポイントは影響されません。このため、パブリケーションが長期間にわたって同期されなくても、ログファイルはトランケートされます。ダウンロード専用のパブリケーションは、ログファイルのトランケーションに影響しません。
ダウンロード専用同期によってスキャンされるログの量を減らすために、時々アップロードを行う必要があります。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。	アップロードを行う必要はありません。

参照

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバー管理』

既存のパブリケーションの変更

パブリケーションを作成してから、アーティクルの追加、修正、削除などの変更を加えたり、パブリケーションの名前を変更したりできます。アーティクルを修正する場合は、そのアーティクル全体の仕様を入力してください。

Sybase Central を使用するか、ALTER PUBLICATION 文によって、これらのタスクを実行できます。

次の点に注意してください。

- DBA 権限を持つユーザーか、パブリケーションの所有者だけがパブリケーションを変更できます。
- 変更には十分注意してください。Mobile Link 設定の実行中にパブリケーションを変更すると、エラーが発生し、データが失われることがあります。変更するパブリケーションにサブスクリプションが含まれている場合は、スキーマのアップグレードとして変更処理を行ってください。 「リモート MobiLink クライアントでのスキーマの変更」64 ページを参照してください。

♦ 既存のパブリケーションまたはアーティクルのプロパティの修正 (Sybase Central の場合)

1. パブリケーションを所有するユーザー、または DBA 権限を持つユーザーとしてリモートデータベースに接続します。
2. 左ウィンドウ枠で、パブリケーションまたはアーティクルをクリックします。プロパティが右ウィンドウ枠に表示されます。
3. プロパティを設定します。

♦ アーティクルの追加 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、パブリケーションを所有するユーザー、または DBA 権限を持つユーザーとしてリモートデータベースに接続します。
2. **[パブリケーション]** フォルダーを展開します。
3. パブリケーションを選択します。
4. **[ファイル]** » **[新規]** » **[アーティクル]** をクリックします。
5. **アーティクル作成ウィザード**で、次の作業を実行します。
   - **[このアーティクルに使用するテーブルを指定してください。]** リストでテーブルを選択します。**[次へ]** をクリックします。
   - **[選択したカラム]** をクリックし、カラムを選択します。**[次へ]** をクリックします。
   - **[このアーティクルに WHERE 句を指定できます。]** ウィンドウ枠で、オプションの WHERE 句を入力します。**[完了]** をクリックします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

♦ アーティクルの削除 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、パブリケーションを所有するユーザー、または DBA 権限を持つユーザーとしてデータベースに接続します。
2. **[パブリケーション]** フォルダーを展開します。
3. パブリケーションを右クリックして、**[削除]** を選択します。
4. **[はい]** をクリックします。

♦ 既存のパブリケーションの修正 (SQL の場合)

1. パブリケーションを所有するユーザー、または DBA 権限を持つユーザーとしてリモートデータベースに接続します。
2. ALTER PUBLICATION 文を実行します。

参照

- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

例

●次の文は、Customers テーブルを pub_contact パブリケーションに追加します。

```
ALTER PUBLICATION pub_contact
 ADD TABLE Customers
```

パブリケーションの削除

Sybase Central または DROP PUBLICATION 文のいずれかを使用して、パブリケーションを削除できます。

パブリケーションを削除する DBA 権限を所有しているか、パブリケーションの所有者でなければなりません。

♦ パブリケーションの削除 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインを使用して、DBA 権限のあるユーザーとしてリモートデータベースに接続します。
2. **[パブリケーション]** フォルダーを開きます。
3. パブリケーションを右クリックして、**[削除]** を選択します。

♦ パブリケーションの削除 (SQL の場合)

1. DBA 権限のあるユーザーとしてリモートデータベースに接続します。

2. DROP PUBLICATION 文を実行します。

参照

- 「DROP PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

例

次の文は、パブリケーション pub_orders を削除します。

```
DROP PUBLICATION pub_orders
```

Mobile Link ユーザーの作成

Mobile Link のユーザー名は、Mobile Link サーバーに接続するときの認証に使用されます。Mobile Link ユーザーをリモートデータベースに作成し、統合データベースに登録してください。

Mobile Link ユーザーは、データベースユーザーとは異なります。データベースユーザー名と一致する Mobile Link ユーザー名を作成することはできますが、Mobile Link も SQL Anywhere も、名前の一致の影響は受けません。

♦ Mobile Link ユーザーのリモートデータベースへの追加 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインから、DBA 権限のあるユーザーとしてデータベースに接続します。
2. **[Mobile Link ユーザー]** フォルダーをクリックします。
3. **[ファイル]** » **[新規]** » **[Mobile Link ユーザー]** をクリックします。
4. **[新しい Mobile Link ユーザーの名前を指定してください。]** フィールドに、Mobile Link ユーザーの名前を入力します。
5. **[完了]** をクリックします。

♦ Mobile Link ユーザーのリモートデータベースへの追加 (SQL の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. CREATE SYNCHRONIZATION USER 文を実行します。Mobile Link ユーザー名はリモートデータベースをユニークに識別します。このため、Mobile Link ユーザー名は同期システム内でユニークである必要があります。

   次は、Mobile Link ユーザー SSinger を追加する例です。

   ```
   CREATE SYNCHRONIZATION USER SSinger
   ```

   Mobile Link ユーザーのプロパティは、CREATE SYNCHRONIZATION USER 文の一部として指定したり、別個に ALTER SYNCHRONIZATION USER 文で指定したりします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「Mobile Link ユーザーの拡張オプションの格納」83 ページ
- 「統合データベースへの Mobile Link ユーザー名の追加」5 ページ

Mobile Link ユーザーの拡張オプションの格納

リモートデータベースで各 Mobile Link ユーザーのオプションを指定するには、拡張オプションを使用します。拡張オプションは、コマンドラインで指定、データベースに格納、sp_hook_dbmlsync_set_extended_options イベントフックで指定できます。

♦ データベースへの Mobile Link 拡張オプションの保存 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインから、DBA 権限のあるユーザーとしてデータベースに接続します。
2. **[Mobile Link ユーザー]** フォルダーを開きます。
3. Mobile Link ユーザー名を右クリックし、**[プロパティ]** を選択します。
4. 必要に応じてプロパティを変更します。

♦ データベースへの Mobile Link 拡張オプションの保存 (SQL の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. ALTER SYNCHRONIZATION USER 文を実行します。

   次は、Mobile Link ユーザー SSinger の拡張オプションをデフォルト値に変更する例です。

   ```
   ALTER SYNCHRONIZATION USER SSinger
   DELETE ALL OPTION
   ```

   Mobile Link ユーザー名を作成するときに、プロパティを指定することも可能です。

♦ クライアントイベントフックを使用した Mobile Link ユーザープロパティの指定

- 次の同期の動作はプログラムを使用してカスタマイズできます。

  詳細については、「sp_hook_dbmlsync_set_extended_options」239 ページを参照してください。

参照

- 「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページ
- 「dbmlsync 拡張オプションの使用」88 ページ
- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

Mobile Link ユーザーの削除

♦ Mobile Link ユーザーのリモートデータベースからの削除 (Sybase Central の場合)

1. SQL Anywhere 12 プラグインから、DBA 権限のあるユーザーとしてデータベースに接続します。
2. **[Mobile Link ユーザー]** フォルダーで、対象の Mobile Link ユーザーを探します。
3. Mobile Link ユーザーを右クリックし、**[削除]** を選択します。

♦ Mobile Link ユーザーのリモートデータベースからの削除 (SQL の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. DROP SYNCHRONIZATION USER 文を実行します。

   次は、データベースから Mobile Link ユーザー SSinger を削除する例です。

   ```
   DROP SYNCHRONIZATION USER SSinger
   ```

参照

- 「DROP SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

同期サブスクリプションの作成

Mobile Link ユーザーとパブリケーションを作成後、1 つまたは複数の既存のパブリケーションに対して、少なくとも 1 人の Mobile Link ユーザーのサブスクリプションを作成してください。これを行うには、同期サブスクリプションを作成します。

注意
Mobile Link ユーザーのすべてのサブスクリプションが、1 つの統合データベースに対してのみ同期されていることを確認する必要があります。複数の統合データベースに同期されていると、データの損失や予期しない動作が発生する場合があります。

同期サブスクリプションは、特定の Mobile Link ユーザーをパブリケーションとリンクします。また、同期に必要なその他の情報を含めることもできます。たとえば、Mobile Link サーバーのアドレスや、同期サブスクリプションに使用する他のオプションを指定できます。特定の同期サブスクリプションの値によって、Mobile Link ユーザーに設定された値が上書きされます。

同期サブスクリプションは、Mobile Link SQL Anywhere リモートデータベース内でのみ必要です。サーバー論理は、統合データベース内の Mobile Link システムテーブルに格納されている同期スクリプトによって実装されます。

単一の SQL Anywhere データベースは複数の Mobile Link サーバーと同期できます。複数のサーバーとの同期を行うには、サーバーごとに異なる Mobile Link ユーザーを作成します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

例

SQL Anywhere サンプルデータベース内の Customers テーブルと SalesOrders テーブルの同期を行うには、次の文を使用します。

1. まず、Customers テーブルと SalesOrders テーブルを含むパブリケーションを作成します。パブリケーション名として testpub を指定します。

```
CREATE PUBLICATION testpub
(TABLE Customers, TABLE SalesOrders)
```

2. 次に Mobile Link ユーザーを作成します。この場合、Mobile Link ユーザーは demo_ml_user です。

```
CREATE SYNCHRONIZATION USER demo_ml_user
```

3. 処理を完了するために、ユーザーとパブリケーションにリンクする my_sub という同期サブスクリプションを作成します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION my_sub TO testpub
 FOR demo_ml_user
 TYPE tcpip
 ADDRESS 'host=localhost;port=2439;'
 SCRIPT VERSION 'version1'
```

参照

- 「パブリケーション」73 ページ
- 「Mobile Link ユーザーの作成」82 ページ

Mobile Link サブスクリプションの変更

Sybase Central を使用するか、ALTER SYNCHRONIZATION SUBSCRIPTION 文を使用すると、同期サブスクリプションを変更できます。構文は CREATE SYNCHRONIZATION SUBSCRIPTION 文に似ていますが、より簡単に追加、修正、削除できるように拡張オプションが用意されています。

♦ 同期サブスクリプションの変更 (Sybase Central の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. **[Mobile Link ユーザー]** フォルダーを開きます。
3. ユーザーをクリックします。サブスクリプションが右ウィンドウ枠に表示されます。
4. 右ウィンドウ枠で変更するサブスクリプションを右クリックし、**[プロパティ]** を選択します。
5. 必要に応じてプロパティを変更します。

♦ 同期サブスクリプションの変更 (SQL の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. ALTER SYNCHRONIZATION SUBSCRIPTION 文を実行します。

参照

- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

Mobile Link サブスクリプションの削除

Sybase Central または DROP SYNCHRONIZATION SUBSCRIPTION 文のいずれかを使用して、同期サブスクリプションを削除できます。

同期サブスクリプションを削除するには、DBA 権限が必要です。

♦ 同期サブスクリプションの削除 (Sybase Central の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. **[Mobile Link ユーザー]** フォルダーを開きます。
3. Mobile Link ユーザーを選択します。
4. サブスクリプションを右クリックして、**[削除]** を選択します。

♦ 同期サブスクリプションの削除 (SQL の場合)

1. DBA 権限のあるユーザーとしてデータベースに接続します。
2. DROP SYNCHRONIZATION SUBSCRIPTION 文を実行します。

例

次の文は、my_sub という同期サブスクリプションを削除します。

```
DROP SYNCHRONIZATION SUBSCRIPTION my_sub
```

参照

- 「DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

同期の開始

Mobile Link 同期を開始するのは、常にクライアントです。SQL Anywhere クライアントの場合は、dbmlsync ユーティリティ、Dbmlsync API または SQL SYNCHRONIZE 文を使用して同期を開

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

始できます。どの方法でもセマンティックは似ていますが、同期のインターフェイスは異なり、独自のアプリケーションに同期を統合する機能も異なります。

同期の動作をカスタマイズするために多くのオプションがありますが、ほぼすべての同期で必要となるいくつかのオプションがあります。ここでは、これらのオプションについて説明します。

-c オプションでは、dbmlsync がリモートデータベースに接続する方法を制御する接続パラメーターを指定できます。SQL SYNCHRONIZE 文を使用するときには、文を実行しているデータベース接続から接続情報が取得されるため、この情報は必要ありません。

-s または Subscription オプションでは、リモートデータベースで定義されているどのサブスクリプションを同期するかを指定できます。

CommunicationAddress と CommunicationType 拡張オプションでは、同期時における Mobile Link サーバーへの dbmlsync の接続方法を決定するネットワークプロトコルオプションを指定できます。

CREATE SYNCHRONIZATION SUBSCRIPTION SQL 文の SCRIPT VERSION 句では、サブスクリプションの同期に使用するスクリプトバージョンを指定できます。スクリプトバージョンによって、同期の制御と処理を行うために Mobile Link サーバーで使用されるスクリプトが決まります。

dbmlsync のパーミッション

同期を行うには、DBA 権限を持つユーザー ID とパスワードを使用して dbmlsync がリモートデータベースに接続されている必要があります。

同期する権限をユーザーに与えても、データベースに関する DBA 権限は付与したくない場合があります。これを行うには、DBA 権限を持たないユーザー ID に REMOTE DBA 権限を付与します。REMOTE DBA 権限を付与されたユーザー ID が DBA 権限を持つのは、dbmlsync ユーティリティから接続が確立された場合のみです。同じユーザー ID を使用する他の接続には、特別な権限は付与されません。

REMOTE DBA のユーザー ID とパスワードは、データベースを同期するどのユーザーにも付与できます。ユーザーは、データベースを同期することはできるようになりますか、データベースに関する他の特殊な権限は付与されません。

同期のカスタマイズ

「dbmlsync の同期のカスタマイズ」97 ページを参照してください。

参照

- 「Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)」 101 ページ
- 「Dbmlsync API」 98 ページ
- 「SYNCHRONIZE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「-c dbmlsync オプション」 111 ページ
- 「-s dbmlsync オプション」 128 ページ
- 「CommunicationAddress (adr) 拡張オプション」 143 ページ
- 「CommunicationType (ctp) 拡張オプション」 144 ページ
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「GRANT REMOTE DBA 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

dbmlsync 拡張オプションの使用

Mobile Link には、同期処理をカスタマイズするための拡張オプションがいくつか用意されています。拡張オプションは、パブリケーション、ユーザー、またはサブスクリプションに設定できます。さらに、dbmlsync コマンドラインまたは sp_hook_dbmlsync_set_extended_options フックプロシージャーでオプションを使用して、拡張オプションの値を上書きすることができます。

♦ dbmlsync コマンドラインでの拡張オプションの上書き

- -e または -eu dbmlsync オプションで、dbmlsync の拡張オプションの値を *option-name=value* の形式で指定します。次に例を示します。

```
dbmlsync -e "v=on;sc=low"
```

♦ サブスクリプション、パブリケーション、またはユーザーの拡張オプションの設定

- SQL Anywhere リモートデータベース内で、CREATE SYNCHRONIZATION SUBSCRIPTION 文または CREATE SYNCHRONIZATION USER 文にオプションを追加します。

  パブリケーションに拡張オプションを追加する場合は、少し異なります。パブリケーションに拡張オプションを追加するには、ALTER/CREATE SYNCHRONIZATION SUBSCRIPTION 文で FOR 句を省略します。

参照

- 「Mobile Link SQL Anywhere クライアントの拡張オプション」 137 ページ
- 「sp_hook_dbmlsync_set_extended_options」 239 ページ

例

次の文は、拡張オプションを使用する同期サブスクリプションを作成します。拡張オプションによって、アップロードストリームを準備するキャッシュサイズを 3 MB に設定し、アップロードのインクリメントサイズを 3 KB に設定します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO my_pub
FOR ml_user
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
ADDRESS 'host=test.internal;port=2439;'
OPTION memory='3m',increment='3k'
```

オプション値は一重引用符で囲む必要がありますが、オプション名は引用符で囲まないでください。

dbmlsync ネットワークプロトコルオプション

dbmlsync 接続情報には、サーバーとの通信に使用するプロトコル、Mobile Link サーバーのアドレスなどの接続パラメーターが含まれます。

参照

- 「CommunicationType (ctp) 拡張オプション」 144 ページ
- 「CommunicationAddress (adr) 拡張オプション」 143 ページ

トランザクションログファイル

通常、何をアップロードするかは、dbmlsync によって SQL Anywhere トランザクションログを使用して決定されます。

SQL Anywhere データベースは、デフォルトでトランザクションログを管理します。データベースの作成時または作成後に dblog ユーティリティを使用して、トランザクションログの場所、またはトランザクションログを使用するかどうかを決定できます。

スクリプト化されたアップロードパブリケーションを同期するか、またはダウンロード専用のパブリケーションのみを使用する場合は、トランザクションログは必要ありません。

アップロードの準備において、dbmlsync ユーティリティは、同期している Mobile Link ユーザーのすべてのサブスクリプションが最後に正常に同期された後に書き込まれたすべてのトランザクションログにアクセスする必要があります。ただし、通常、SQL Anywhere ログファイルは、定期的なデータベース管理作業の中でトランケートされ、名前が変更されます。その場合は、記述されている変更内容がすべて正常に同期されるまで、古いログファイルの名前を変更し、別のディレクトリに保存してください。

dbmlsync コマンドラインで、名前が変更されたログファイルが格納されているディレクトリを指定できます。前回の同期の後で作業ログファイルのトランケートと名前の変更が行われていない場合、または名前が変更されたログファイルがあるディレクトリから dbmlsync を実行する場合は、このパラメーターを省略できます。

参照

- 「バックアップとデータリカバリ」『SQL Anywhere サーバー データベース管理』
- 「進行オフセット」 72 ページ
- 「トランザクションログ」『SQL Anywhere サーバー データベース管理』
- 「初期化ユーティリティ (dbinit)」『SQL Anywhere サーバー データベース管理』
- 「スクリプト化されたアップロード」 311 ページ

例

古いログファイルがディレクトリ *c:¥oldlogs* に格納されているとします。次のコマンドを使用してリモートデータベースを同期することができます。

```
dbmlsync -c "dbn=remote;uid=syncuser" c:¥oldlogs
```

古いログディレクトリへのパスは、コマンドラインに最後の引数として指定してください。

同期中の同時実行性

同期の整合性を保証するため、最後のアップロードが送信された後で変更されたリモートデータベースのサーバー修正ローから変更がダウンロードされていないことを dbmlsync で確認する必要があります。デフォルトでは通常、テーブルがロックされずにこの操作が行われるため、データベース上の他の同時接続ユーザーに与える影響が最小限に抑えられます。スクリプト化されたアップロードを使用するパブリケーションを同期する場合、または sp_hook_dbmlsync_schema_upgrade フックが定義されている場合には、テーブルが IN SHARE MODE でロックされます。

テーブルがロックされていないと、dbmlsync はアップロードの構築後に修正されたローをすべて追跡します。これらのいずれかのローの変更がダウンロードに含まれている場合は、競合とみなされます。

競合が検出された場合は、ダウンロードフェーズがキャンセルされ、新しい変更が上書きされないようにダウンロード操作がロールバックされます。次に、dbmlsync ユーティリティはアップロード手順を含む同期を再試行します。今度はローが同期処理の最初に処理されており、アップロードにこのローが含まれているため、このローを失うことはありません。

デフォルトでは、dbmlsync は正常に実行されるまで同期のリトライを行います。リトライの回数を制限するには、拡張オプション ConflictRetries を使用します。ConflictRetries を -1 に設定すると、正常に実行されるまで dbmlsync によってリトライが実行されます。これを正の整数に設定すると、dbmlsync は指定した回数以内でリトライを実行します。

-d オプション

このロックメカニズムを使用しているときに、データベースに別の接続が存在し、その接続に同期テーブルに対するロックがある場合は、同期が失敗します。別のロックが存在しても、同期がすぐに行われるようにする場合は、dbmlsync で -d オプションを使用します。このオプションを指定すると、同期に影響するロックのある接続はデータベースによって削除されるため、同期を進行できます。削除された接続のコミットされていない変更は、ロールバックされます。

LockTables オプション

LockTables 拡張オプションを使用した同期では、dbmlsync でテーブルを強制的にロックできます。フックプロシージャーに書き込むロジックを簡単にするため、同期時にテーブルをロックするのが望ましい場合があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「ConflictRetries (cr) 拡張オプション」145 ページ
- 「-d dbmlsync オプション」113 ページ
- 「LockTables (lt) 拡張オプション」154 ページ

アプリケーションからの同期の開始

リモートユーザーに別個の実行ファイルを提供するのではなく、アプリケーションに dbmlsync の機能を組み込むことができます。

このようにするには、次の 3 つの方法があります。

●Dbmlsync API

詳細については、「Dbmlsync API」98 ページを参照してください。

●SQL SYNCHRONIZE 文

詳細については、「SYNCHRONIZE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

●DLL を呼び出すことのできる言語で開発している場合は、DBTools インターフェイスを使用して dbmlsync にアクセスできます。C や C++ でプログラムを作成している場合は、SQL Anywhere ディレクトリの *SDK¥Include* サブフォルダーにある dbtools.h ヘッダーファイルを含めることができます。12 このファイルには、a_sync_db 構造体と DBSynchronizeLog 関数の記述があり、dbmlsync 機能をアプリケーションに追加するときに使用します。この解決方法は、Windows や UNIX など、サポート対象となっているすべてのプラットフォームに使用できます。

Dbmlsync API と SQL SYNCHRONIZE 文は、どちらも DBTools インターフェイスより使いやすいため、まずこれを使用することを強くおすすめします。

詳細については、次の項を参照してください。

- ○ 「dbmlsync の DBTools インターフェイス」305 ページ
- ○ DBSynchronizeLog メソッド [データベースツール]『SQL Anywhere サーバー プログラミング』
- ○ a_sync_db 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』

Microsoft ActiveSync との同期

Microsoft ActiveSync は、Microsoft Windows Mobile ハンドヘルドデバイス用の同期ソフトウェアです。Microsoft ActiveSync は、Windows Mobile デバイスとデスクトップコンピューター間の同期を制御します。Microsoft ActiveSync 用の Mobile Link プロバイダーは、Mobile Link サーバーとの同期を制御します。

SQL Anywhere クライアント用の Microsoft ActiveSync 同期を設定するには、次の手順に従います。

●SQL Anywhere リモートデータベースを Microsoft ActiveSync 同期用に設定します。

「Microsoft ActiveSync 用の SQL Anywhere リモートデータベースの設定」92 ページを参照してください。

●Microsoft ActiveSync 用 Mobile Link プロバイダーをインストールします。

「Microsoft ActiveSync 用 Mobile Link プロバイダーのインストール」93 ページを参照してください。

●SQL Anywhere クライアントを、Microsoft ActiveSync で使用できるように登録します。

「Microsoft ActiveSync 用 SQL Anywhere クライアントの登録」94 ページを参照してください。

Microsoft ActiveSync 同期を使用する場合は、Microsoft ActiveSync ソフトウェアから同期を開始します。Microsoft ActiveSync 用の Mobile Link プロバイダーは、dbmlsync を起動するか、スケジュール文字列でのスケジュールに従ってスリープ中の dbmlsync をウェイクアップできます。

リモートデータベース内で遅延フックを使用して dbmlsync をスリープモードに設定することもできますが、Microsoft ActiveSync 用の Mobile Link プロバイダーは、このステータスからは同期を開始できません。

同期スケジュールの詳細については、「同期のスケジュール」95 ページを参照してください。

Microsoft ActiveSync 用の SQL Anywhere リモートデータベースの設定

♦ Microsoft ActiveSync 用の SQL Anywhere リモートデータベースの設定

1. 同期タイプ (TCP/IP、TLS、HTTP、または HTTPS) を選択します。

   同期タイプは、同期パブリケーション、同期ユーザー、または同期サブスクリプション用に設定できます。それぞれの設定方法は似ています。ここでは、典型的な CREATE SYNCHRONIZATION USER 文の一部を示します。

   ```
   CREATE SYNCHRONIZATION USER SSinger
   TYPE tcpip
   ...
   ```

2. ADDRESS 句を使用して、Microsoft ActiveSync 用 Mobile Link プロバイダーと Mobile Link サーバー間の通信を指定します。

   HTTP または TCP/IP 同期の場合は、CREATE SYNCHRONIZATION USER または CREATE SYNCHRONIZATION SUBSCRIPTION 文の ADDRESS 句によって、Mobile Link クライアントとサーバー間の通信を指定します。Microsoft ActiveSync の場合、通信は 2 段階で発生します。つまり、デバイス上の dbmlsync ユーティリティからデスクトップコンピューター上の Microsoft ActiveSync 用 Mobile Link プロバイダーへの通信が発生してから、デスクトップコンピューターから Mobile Link サーバーへの通信が発生します。ADDRESS 句では、Microsoft ActiveSync 用 Mobile Link プロバイダーと Mobile Link サーバー間の通信を指定します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

次の文は、コンピューター kangaroo 上の Mobile Link サーバーへの TCP/IP 通信を指定します。

```
CREATE SYNCHRONIZATION USER SSinger
TYPE tcpip
ADDRESS 'host=kangaroo;port=2439'
```

参照

- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

Microsoft ActiveSync 用 Mobile Link プロバイダーのインストール

インストールユーティリティ (*mlasinst.exe*) を使用して、Microsoft ActiveSync 用 Mobile Link プロバイダーをインストールしてから、SQL Anywhere Mobile Link クライアントを Microsoft ActiveSync 用に登録します。

SQL Anywhere for Windows Mobile のインストーラーによって、Microsoft ActiveSync 用 Mobile Link プロバイダーがインストールされます。SQL Anywhere for Windows Mobile をインストールする場合、この項で説明する手順を実行する必要はありません。

Microsoft ActiveSync 用 Mobile Link プロバイダーをインストールしたら、各アプリケーションを個別に登録してください。

♦ Microsoft ActiveSync 用 Mobile Link プロバイダーのインストール

1. 使用中のコンピューターに Microsoft ActiveSync ソフトウェアがインストールしてあり、Windows Mobile デバイスが接続されていることを確認します。
2. 次のコマンドを実行して、Mobile Link プロバイダーをインストールします。

   ```
   mlasinst -k desk-path -v dev-path
   ```

   desk-path はプロバイダーのデスクトップコンポーネント (*mlasdesk.dll*) のロケーション、*dev-path* はデバイスコンポーネント (*mlasdev.dll*) のロケーションです。

   SQL Anywhere をコンピューターにインストールしてある場合、*mlasdesk.dll* は *%SQLANY12%¥bin32*、*mlasdev.dll* は *%SQLANY12%¥CE* にあります。-v または -k を省略すると、デフォルトでこれらのディレクトリが検索されます。

   リモートプロバイダーが開けないというメッセージが表示された場合は、デバイスのソフトリセットを実行し、コマンドを繰り返します。
3. コンピューターを再起動します。

   コンピューターを再起動すると、Microsoft ActiveSync で新しいプロバイダーが認識されます。
4. Mobile Link プロバイダーを有効にします。

Vista よりも前のバージョンの Windows の場合：

- [Microsoft ActiveSync] ウィンドウで **[オプション]** をクリックします。
- リストにある [Mobile Link] 項目を有効にして **[OK]** をクリックし、Mobile Link プロバイダーをアクティブにします。
- 登録されたアプリケーションのリストを表示するには、もう一度 **[オプション]** をクリックし、Mobile Link プロバイダーを選択して **[設定]** をクリックします。

Windows Vista の場合：

- [Windows Mobile デバイス センター] ウィンドウで、**[モバイル デバイスの設定][コンテンツの設定の変更]** をクリックします。
- **[Mobile Link クライアント]** を選択し、**[保存]** をクリックして Mobile Link プロバイダーをアクティブにします。
- 登録されたアプリケーションのリストを表示するには、**[コンテンツの設定の変更][Mobile Link クライアント][同期の設定]** をクリックします。

参照

- 「Microsoft ActiveSync 用 SQL Anywhere クライアントの登録」94 ページ
- 「Microsoft ActiveSync プロバイダーインストールユーティリティ (mlasinst)」19 ページ

Microsoft ActiveSync 用 SQL Anywhere クライアントの登録

Microsoft ActiveSync で使用するアプリケーションを登録するには、Microsoft ActiveSync プロバイダーのインストールユーティリティを使用する方法と、Microsoft ActiveSync ソフトウェア自体を使用する方法があります。この項では、Microsoft ActiveSync ソフトウェアを使用する方法について説明します。

♦ Microsoft ActiveSync で使用するための SQL Anywhere クライアントの登録

1. Microsoft ActiveSync 用 Mobile Link プロバイダーがインストールされていることを確認します。
2. デスクトップコンピューターで、Microsoft ActiveSync ソフトウェアを起動します。
3. Vista よりも前のバージョンの Windows の場合：

   - **[Microsoft ActiveSync]** ウィンドウで **[オプション]** を選択します。
   - 情報タイプのリストから、**[Mobile Link]** を選択し、**[設定]** をクリックします。
   - **[Mobile Link 同期]** ウィンドウで **[新規]** をクリックします。

   Windows Vista の場合：

   - **[Windows Mobile デバイス センター]** ウィンドウで、**[モバイル デバイスの設定][コンテンツの設定の変更]** をクリックします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

- **[コンテンツの設定の変更]** をクリックします。
- **[Mobile Link クライアント]** をクリックします。
- **[同期の設定]** をクリックします。

4. アプリケーションについて次の情報を入力します。
   - **[アプリケーション名]** Microsoft ActiveSync ユーザーインターフェイスに表示されるアプリケーションを識別する名前。
   - **[クラス名]** -wc オプションを使用して設定した、dbmlsync クライアントのクラス名。
   - **[パス]** デバイス上の dbmlsync アプリケーションのロケーション。
   - **[引数]** Microsoft ActiveSync が dbmlsync の起動時に使用するコマンドライン引数。

     dbmlsync は、2 つのモードのうちの 1 つを使用して開始します。

     ○スケジューリングオプションを指定すると、dbmlsync は停止モードに入ります。この場合、クラス名の設定と一致する値を指定した dbmlsync -wc オプションを使用します。

     ○このように指定しないと、dbmlsync は停止モードに入りません。この場合、-k を使用して dbmlsync を停止します。
5. **[OK]** をクリックしてアプリケーションを登録します。

参照

- 「Microsoft ActiveSync プロバイダーインストールユーティリティ (mlasinst)」 19 ページ
- 「Microsoft ActiveSync 用 Mobile Link プロバイダーのインストール」 93 ページ
- 「-wc dbmlsync オプション」 136 ページ
- 「-k dbmlsync オプション (旧式)」 119 ページ
- 「-wc dbmlsync オプション」 136 ページ
- 「同期のスケジュール」 95 ページ

同期のスケジュール

定義した規則に基づいて定期的に同期を実行するよう dbmlsync を設定できます。dbmlsync を設定する方法は 2 つあります。

- 特定の時刻や曜日、または定期的に同期を開始するには、dbmlsync 拡張オプションの SCHEDULE を使用します。この場合、ユーザーが停止するまで dbmlsync は実行を続けます。

  「dbmlsync オプションを使用したスケジュールの設定」 96 ページを参照してください。

- 定義した論理に基づいて同期を開始するには、dbmlsync イベントフックを使用します。この方法は、不定期またはイベントの応答として同期を起動する場合に適しています。この場合、指定したフックコードによって dbmlsync を自動的に停止できます。

  「イベントフックを使用した同期の開始」 97 ページを参照してください。

Dbmlsync API または SQL SYNCHRONIZE 文が使用されている場合は、このメソッドを使用できません。

停止

停止時には、dbmlsync はデータベーストランザクションログをスキャンし、同期間の遅延時にアップロードを構築します。一部の操作はすでに完了しているため、これにより、同期がトリガーされたときの処理をすばやく行うことができます。

停止時には、dbmlsync はトランザクションログの最後までスキャンしてから、新しいトランザクションに対してログを定期的にポーリングします。PollingPeriod 拡張オプションまたは -pp オプションを使用して、ポーリング間隔を制御できます。 「PollingPeriod (pp) 拡張オプション」159 ページを参照してください。

複数のサブスクリプションを同時に停止している場合は、HoverRescanThreshold 拡張オプションまたは sp_hook_dbmlsync_log_rescan イベントフックを使用し、メモリをリカバリすることによって (リカバリしないとメモリは失われる)、メモリ使用量を制限できます。

DisablePolling 拡張オプションまたは -p オプションを使用して、停止を無効にすることができます。

参照

- 「HoverRescanThreshold (hrt) 拡張オプション」151 ページ
- 「-p dbmlsync オプション」123 ページ
- 「DisablePolling (p) 拡張オプション」146 ページ
- 「sp_hook_dbmlsync_log_rescan」225 ページ

dbmlsync オプションを使用したスケジュールの設定

dbmlsync をバッチ形式で実行して、同期終了後に停止するように設定する代わりに、dbmlsync を継続的に実行してあらかじめ決められた時間に同期を行うように、SQL Anywhere クライアントを設定することもできます。

同期スケジュールは、拡張オプションとして指定します。同期スケジュールを dbmlsync コマンドライン上で指定するか、同期ユーザー、同期サブスクリプション、または同期パブリケーション用にデータベースに同期スケジュールを格納できます。

Dbmlsync API または SQL SYNCHRONIZE 文が使用されている場合は、このメソッドを使用できません。

♦ 同期サブスクリプションへのスケジュールの追加

- 同期サブスクリプション内で Schedule 拡張オプションを設定します。次に例を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
FOR mluser
ADDRESS 'host=localhost'
OPTION schedule='weekday@11:30am-12:30pm'
```

dbmlsync -is オプションを使用すると、スケジュールの設定を無効にし、直ちに同期を行うことができます。-is オプションは、スケジュール拡張オプションで指定したスケジュールを無視するよう dbmlsync に指示します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

♦ **dbmlsync コマンドラインでのスケジュールの追加**

- スケジュール拡張オプションを設定します。拡張オプションは -e または -eu で設定します。次に例を示します。

```
dbmlsync -e "sch=weekday@11:30am-12:30pm" ...
```

同期スケジュールがこのどちらかで指定された場合、dbmlsync は同期終了後も停止せず、継続して実行します。

参照

- 「Schedule (sch) 拡張オプション」160 ページ
- 「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページ
- 「-eu dbmlsync オプション」118 ページ
- 「-is dbmlsync オプション」119 ページ

イベントフックを使用した同期の開始

同期処理のタイミングを制御するために実装できる dbmlsync イベントフックがあります。

sp_hook_dbmlsync_end フックを使用すると、#hook_dict テーブルの Restart ローを使用して、同期処理が終了するたびに dbmlsync が同期を繰り返すかどうかを判断できます。

sp_hook_dbmlsync_delay フックを使用すると、同期処理の開始時に遅延を作成して、同期を続行するタイミングを選択できます。このフックでは、一定の時間遅延させたり、定期的にポーリングを行うことで、ある条件が満たされるまで待機できます。

Dbmlsync API または SQL SYNCHRONIZE 文が使用されている場合は、このメソッドを使用できません。

参照

- 「sp_hook_dbmlsync_end」221 ページ
- 「sp_hook_dbmlsync_delay」210 ページ

dbmlsync の同期のカスタマイズ

dbmlsync クライアントイベントフック

イベントフックでは、SQL ストアドプロシージャーを使用して、dbmlsync のクライアント側の同期処理を管理できます。クライアントイベントフックは、dbmlsync コマンドラインユーティリティや dbmlsync プログラミングインターフェイスで使用できます。

イベントフックを使用して同期イベントのログを取り、処理することができます。たとえば、論理イベントに基づいた同期のスジュール、接続障害のリトライ、または特定のエラーや参照整合性違反の処理などが可能です。

dbmlsync プログラミングインターフェイス

次のプログラミングインターフェイスを使用して、Mobile Link クライアントをアプリケーションに統合し、同期を開始できます。これらのインターフェイスは、dbmlsync コマンドラインユーティリティの代わりに使用できます。

- **Dbmlsync API** Dbmlsync API は、C++ または .NET で記述された Mobile Link クライアントが同期を起動し、要求した同期の進行状況に関するフィードバックを受け取れるようにするプログラミングインターフェイスです。この新しいプログラミングインターフェイスを使用することで、同期の結果に関してアクセスできる情報が大幅に増え、同期のキューイングも可能になるため、同期の管理が容易になります。

- **dbmlsync の DBTools インターフェイス** dbmlsync 用の DBTools インターフェイスを使用することで、SQL Anywhere 同期クライアントアプリケーションに同期機能を統合できます。SQL Anywhere データベース管理ユーティリティはすべて、DBTools によって構築されます。

スクリプト化されたアップロード

また、クライアントトランザクションログの使用を上書きして、独自のアップロードストリームを定義することもできます。

参照

- 「SQL Anywhere クライアントのイベントフック」195 ページ
- 「Dbmlsync API」98 ページ
- 「dbmlsync の DBTools インターフェイス」305 ページ
- 「スクリプト化されたアップロード」311 ページ

Dbmlsync API

Dbmlsync API は、C++ または .NET で記述された Mobile Link クライアントアプリケーションが同期を起動し、要求した同期の進行状況に関するフィードバックを受け取れるようにするプログラミングインターフェイスです。この API は、同期をアプリケーションにシームレスに統合するためのものです。

このプログラミングインターフェイスを使用することで、同期の結果に関してアクセスできる情報が大幅に増え、同期のキューイングも可能になるため、同期の管理が容易になります。

警告
Dbmlsync API はスレッド対応ではありません。DbmlsyncClient クラスの単一インスタンスのすべての呼び出しは、同じスレッドで行う必要があります。DbmlsyncClient の単一インスタンスの関数を異なるスレッドから呼び出すと、予期しないエラーが発生したり結果の信頼性が損なわれたりする可能性があります。

参照

- 「Dbmlsync .NET API リファレンス」278 ページ
- 「Dbmlsync C++ API リファレンス」253 ページ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

SQL Anywhere クライアントのログ

SQL Anywhere リモートデータベースを使用する Mobile Link アプリケーションを作成する場合、注意が必要なクライアントログファイルは 2 種類あります。

- dbmlsync のメッセージログ
- SQL Anywhere トランザクションログ

dbmlsync のメッセージログ

デフォルトでは、dbmlsync のメッセージは dbmlsync のメッセージウィンドウに送信されます。また、-o または -ot オプションを使用して結果をメッセージログファイルにも送信できます。次のコマンドラインの一部では、結果を *dbmlsync.dbs* という名前のログファイルに送ります。

```
dbmlsync -o dbmlsync.dbs ...
```

dbmlsync アクティビティをロギングすると、開発プロセスとトラブルシューティングのときに特に役立ちます。

ログファイルのサイズを制御したり、ファイルが最大サイズに達したときの処理を指定したりできます。

- ログファイルを指定して、結果をログファイルに追加する場合は、-o オプションを使用します。
- ログファイルを指定して、結果をログファイルに追加する前にファイルの内容を削除する場合は、-ot オプションを使用します。
- -o または -ot に加えて -os オプションを使用してサイズを指定すると、そのサイズに達したときに、ログファイルの名前が変更され、元の名前を持つ新しいファイルが使用されます。

メッセージログファイルを指定しなかった場合、すべての出力が dbmlsync のメッセージウィンドウに表示されます。メッセージログファイルを指定した場合、dbmlsync のメッセージウィンドウに送信される出力は指定しなかった場合よりも少なくなります。

-v オプションを使用すると、メッセージログファイルに記録され、dbmlsync のメッセージウィンドウに表示される情報を制御できます。パフォーマンスが低下するため、運用環境の通常の操作には冗長出力を使用しないでください。

SQL Anywhere トランザクションログ

「トランザクションログファイル」89 ページを参照してください。

参照

- 「-o dbmlsync オプション」122 ページ
- 「-ot dbmlsync オプション」122 ページ
- 「-os dbmlsync オプション」122 ページ
- 「-v dbmlsync オプション」135 ページ

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Mobile Link の Mac OS X での実行

Mobile Link サーバーと SQL Anywhere Mobile Link クライアントは、Mac OS X で実行できます。Ultra Light は、Mac OS X では実行できません。

Mac OS X 上の Mobile Link 統合データベースを同期するには、ドライバーマネージャーとして SQL Anywhere ODBC ドライバーを使用します。 「ODBC データソースの作成 (Mac OS X の場合)」『SQL Anywhere サーバー データベース管理』を参照してください。

♦ Mac OS X での Mobile Link サーバーの開始

1. SyncConsole を起動します。

   [Finder] で、SyncConsole をダブルクリックします。SyncConsole アプリケーションは */Applications/SQLAnywhere12* にあります。

2. **[ファイル]** » **[新規]** » **[Mobile Link サーバー]** をクリックします。

3. Mobile Link サーバーを設定します。

   a. **[接続パラメーター]** フィールドに次の文字列を入力します。

      ```
      DSN=dsn-name
      ```

      dsn-name は SQL Anywhere ODBC データソース名です。 ODBC データソースの作成については、「ODBC データソース」『SQL Anywhere サーバー データベース管理』を参照してください。

      dsn-name にスペースが含まれている場合は、文字列を二重引用符で囲みます。次に例を示します。

      ```
      DSN="SQL Anywhere 12 Demo"
      ```

   b. 必要に応じて、**[オプション]** フィールドでオプションを設定します。

      [オプション] フィールドを使用すると、Mobile Link サーバーの動作をさまざまな面から制御できます。 オプションの完全なリストについては、「mlsrv12 構文」『Mobile Link サーバー管理』を参照してください。

4. **[起動]** をクリックして、Mobile Link サーバーを起動します。

   データベースサーバーメッセージウィンドウが開き、サーバーが同期要求を受け付ける準備ができていることを示すメッセージが表示されます。

♦ Mac OS X での dbmlsync の開始

1. SyncConsole を起動します。

   [Finder] で、**SyncConsole** をダブルクリックします。SyncConsole アプリケーションは */Applications/SQLAnywhere12* にあります。

2. **[ファイル]** » **[新規]** » **[Mobile Link クライアント]** をクリックします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

クライアントオプションのウィンドウが表示されます。このウィンドウには設定オプションが多数用意されていますが、それぞれが dbmlsync コマンドラインオプションに対応しています。 完全なリストについては、「dbmlsync 構文」 101 ページを参照してください。

[ログイン]、**[データベース]**、**[ネットワーク]**、**[詳細]** の各タブのオプションはすべて、Mobile Link クライアントから SQL Anywhere リモートデータベースへの接続を定義しています。ほとんどの場合、**[ログイン]** タブの ODBC データソースを指定するだけで接続できます。

[DBMLSync] タブのオプションは、Mobile Link サーバーへの接続について定義します。この機能がリモートデータベースのパブリケーションおよびサブスクリプションで定義されている場合は、このタブのオプションは空のままにできます。

♦ Mac OS X でのサンプルデータベースの実行

1. *sa_config* 設定スクリプトを準備します。

   詳細については、「Unix および Mac OS X 環境変数」『SQL Anywhere サーバー データベース管理』を参照してください。

2. ODBC データソースを設定します。次に例を示します。

   ```
   dbdsn -w  "SQL Anywhere 12 Demo"
   -c "UID=DBA;PWD=sql;DBF=/Applications/SQLAnywhere12/System/demo.db"
   ```

3. Mobile Link サーバーを実行します。次に例を示します。

   ```
   mlsrv12 -c "DSN=SQL Anywhere 12 Demo"
   ```

バージョンに関する考慮事項

dbmlsync が正しく機能するためには、dbmlsync.exe のメジャーバージョンとマイナーバージョンの両方が、データベースサーバーと一致している必要があります。さらに、データベースファイルのメジャーバージョンが dbmlsync.exe のメジャーバージョンと一致していて、データベースファイルのマイナーバージョンが *dbmlsync.exe* のマイナーバージョン以下である必要があります。データベースファイルのバージョンとは、アップグレードされている最新のバージョンのことです。

たとえば、9.02 バージョンの dbmlsync は、9.02 バージョンのデータベースサーバー (*dbeng9.exe*) に対してのみ使用し、9.00、9.01、9.02 のデータベースファイルを使用できます。

Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync)

dbmlsync 構文

dbmlsync ユーティリティを使用して、SQL Anywhere リモートデータベースと統合データベースの同期を行います。

構文

dbmlsync [*options*] [*transaction-logs-directory*]

オプション	説明
@data	指定された環境変数または設定ファイルからオプションを読み込みます。 「@data dbmlsync オプション」 107 ページを参照してください。
-a	エラー時に再入力のプロンプトを表示しません。 「-a dbmlsync オプション」 107 ページを参照してください。
-ap	認証パラメーターを指定します。 「-ap dbmlsync オプション」 107 ページを参照してください。
-ba *filename*	ダウンロードファイルを適用します。 「-ba dbmlsync オプション」 108 ページを参照してください。
-bc *filename*	ダウンロードファイルを作成します。 「-bc dbmlsync オプション」 108 ページを参照してください。
-be *string*	ダウンロードファイルを作成するときに文字列を追加します。 「-be dbmlsync オプション」 109 ページを参照してください。
-bg	ダウンロードファイルを作成するときに、そのファイルを新しいリモートに適合するようにします。 「-bg dbmlsync オプション」 109 ページを参照してください。
-bk	バックグラウンド同期を有効にします。 「-bk dbmlsync オプション」 110 ページを参照してください。
-bkr	バックグラウンド同期が中断された後の dbmlsync の動作を制御します。 「-bkr dbmlsync オプション」 111 ページを参照してください。
-c *connection-string*	*parm1=value1*; *parm2=value2*,... の形式で、リモートデータベースへの接続に使用するデータベース接続パラメーターを指定します。このオプションを指定しなかった場合はウィンドウが表示され、そこで接続情報を指定します。 「-c dbmlsync オプション」 111 ページを参照してください。
-ci *size*	dbmlsync キャッシュの初期サイズを設定します。 「-ci dbmlsync オプション」 112 ページを参照してください。
-cl *size*	dbmlsync キャッシュファイルの最小サイズスレッショルドを設定します。 「-cl dbmlsync オプション」 112 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
-cm *size*	dbmlsync キャッシュファイルの最大サイズ上限値を設定します。 「-cm dbmlsync オプション」 113 ページを参照してください。
-d	ロックされているデータベースへの接続のうち、同期されたアーティクルと競合しているものをすべて削除します。 「-d dbmlsync オプション」 113 ページを参照してください。
-dc	以前に失敗したダウンロードを続行します。 「-dc dbmlsync オプション」 113 ページを参照してください。
-dl	dbmlsync のメッセージウィンドウにログメッセージを表示します。 「-dl dbmlsync オプション」 114 ページを参照してください。
-do	オフライントランザクションログのスキャンを無効にします。 「-do dbmlsync オプション」 114 ページを参照してください。
-drs *bytes*	再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定します。 「-drs dbmlsync オプション」 115 ページを参照してください。
-ds	ダウンロード専用同期を実行します。 「-ds dbmlsync オプション」 116 ページを参照してください。
-e "*option=value*"...	拡張オプションを指定します。 「Mobile Link SQL Anywhere クライアントの拡張オプション」 137 ページを参照してください。
-eh	フック関数で発生したエラーを無視します。
-ek *key*	リモートデータベースの暗号化キーを指定します。 「-ek dbmlsync オプション」 118 ページを参照してください。
-ep	リモートデータベースの暗号化キーを入力するよう要求します。 「-ep dbmlsync オプション」 118 ページを参照してください。
-eu	最新の -n オプションで定義されたアップロードに対して拡張オプションを指定します。 「-eu dbmlsync オプション」 118 ページを参照してください。
-is	スケジュールを無視します。 「-is dbmlsync オプション」 119 ページを参照してください。
-k	完了後、ウィンドウを閉じます。 「-k dbmlsync オプション (旧式)」 119 ページを参照してください。

オプション	説明
-l	使用可能な拡張オプションをリストします。 「-l dbmlsync オプション」 119 ページを参照してください。
-mn *password*	新しい Mobile Link パスワードを指定します。 「-mn dbmlsync オプション」 120 ページを参照してください。
-mp *password*	Mobile Link パスワードを指定します。 「-mp dbmlsync オプション」 120 ページを参照してください。
-n *name*	同期パブリケーション名を指定します。 「-n dbmlsync オプション (旧式)」 121 ページを参照してください。
-o *logfile*	このファイルに出力メッセージのログを取ります。 「-o dbmlsync オプション」 122 ページを参照してください。
-os *size*	メッセージログファイルの最大サイズを指定します (このサイズに達するとログの名前が変更されます)。 「-os dbmlsync オプション」 122 ページを参照してください。
-ot *logfile*	メッセージログファイルの内容を削除してから、このファイルに出力メッセージのログを取ります。 「-ot dbmlsync オプション」 122 ページを参照してください。
-p	ログスキャンのポーリングを無効にします。 「-p dbmlsync オプション」 123 ページを参照してください。
-pc+	同期と同期の間で、Mobile Link サーバーへのオープン接続を維持します。 「-pc+ dbmlsync オプション」 123 ページを参照してください。
-pd *dllname*;...	Windows Mobile 用の指定された DLL をプリロードします。 「-pd dbmlsync オプション」 124 ページを参照してください。
-pi	Mobile Link に接続できるかどうかをテストします。 「-pi dbmlsync オプション」 125 ページを参照してください。
-po	dbmlsync が受信するポートを指定します。 「-po dbmlsync オプション」 125 ページを参照してください。
-pp *number*	ログスキャンのポーリング周期を設定します。 「-pp dbmlsync オプション」 126 ページを参照してください。
-q	最小化ウィンドウで実行します。 「-q dbmlsync オプション」 126 ページを参照してください。
-qc	同期が終了したときに dbmlsync を停止します。 「-qc dbmlsync オプション」 126 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
-qi	dbmlsync をクワイエットモードで起動し、ウィンドウを完全に非表示にします。 「-qi dbmlsync オプション」127 ページを参照してください。
-r[**a** | **b**]	アップロードのリトライにクライアントの進行状況値を使用します。 「-r dbmlsync オプション」127 ページを参照してください。
-s *name*	同期サブスクリプション名を指定します。 「-s dbmlsync オプション」128 ページを参照してください。
-sc	各同期の前にスキーマ情報を再ロードします。「-sc dbmlsync オプション」129 ページを参照してください。
-sm	dbmlsync がサーバーモードで起動するようにします。 「-sm dbmlsync オプション」129 ページを参照してください。
-sp *sync profile*	コマンドラインで指定される同期オプションに、同期プロファイルからのオプションを追加します。 「-sp dbmlsync オプション」130 ページを参照してください。
-tu	トランザクション単位のアップロードを実行します。 「-tu dbmlsync オプション」130 ページを参照してください。
-u *ml_username*	同期する Mobile Link ユーザーを指定します。 「-u dbmlsync オプション (旧式)」132 ページを参照してください。
-ud	UNIX 専用です。dbmlsync をデーモンとして実行します。 「-ud dbmlsync オプション」132 ページを参照してください。
-ui	X Window がサポートされている Linux で、使用可能なディスプレイがない場合にシェルモードで dbmlsync を起動します。 「-ui dbmlsync オプション」133 ページを参照してください。
-uo	アップロード専用同期を実行します。 「-uo dbmlsync オプション」133 ページを参照してください。
-urc *row-estimate*	アップロードされるロー数の推定値を指定します。 「-urc dbmlsync オプション」134 ページを参照してください。
-ux	Solaris と Linux で、dbmlsync のメッセージウィンドウを開きます。 「-ux dbmlsync オプション」134 ページを参照してください。
-v[*levels*]	冗長オペレーション。 「-v dbmlsync オプション」135 ページを参照してください。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
-wc *classname*	ウィンドウクラス名を指定します。「-wc dbmlsync オプション」136 ページを参照してください。
-x	トランザクションログの名前を変更して再起動します。オプションの size パラメーターを -x オプションと一緒に使用して、トランザクションログのサイズを制御します。「-x dbmlsync オプション」136 ページを参照してください。
transaction-logs-directory	トランザクションログのロケーションを指定します。下のトランザクションログファイルを参照してください。

備考

dbmlsync を実行して、SQL Anywhere リモートデータベースと統合データベースの同期を行います。

Mobile Link サーバーを検出して接続するために、dbmlsync はパブリケーション、同期ユーザー、同期サブスクリプション、または dbmlsync コマンドラインの情報を使用します。

トランザクションログファイル　*transaction-logs-directory* には、SQL Anywhere リモートデータベースのトランザクションログが格納されているディレクトリを指定します。アクティブなトランザクションログファイルと 0 個以上のトランザクションログアーカイブファイルがあります。dbmlsync がアップロードするデータを判別するには、このすべてのファイルが必要です。次のすべての条件を満たす場合、このパラメーターを指定してください。

○前回の同期の後で、作業ログファイルの内容がトランケートされ、ファイルの名前が変更されている場合

○名前が変更されたログファイルが格納されているディレクトリ以外のディレクトリから、dbmlsync ユーティリティを実行する場合

dbmlsync イベントフック　同期処理のカスタマイズに役立つ dbmlsync クライアントストアドプロシージャーもあります。

dbmlsync の使用　dbmlsync の使用の詳細については、「同期の開始」86 ページを参照してください。

参照

- 「トランザクションログファイル」89 ページ
- 「SQL Anywhere クライアントのイベントフック」195 ページ
- 「同期の開始」86 ページ
- 「SQL Anywhere クライアントのイベントフック」195 ページ
- 「Dbmlsync API」98 ページ
- 「dbmlsync の DBTools インターフェイス」305 ページ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

@data dbmlsync オプション

指定された環境変数または設定ファイルからオプションを読み込みます。

構文

dbmlsync @*data* ...

備考

このオプションを指定すると、環境変数または設定ファイル内にコマンドラインオプションを記述できます。指定された名前に環境変数と設定ファイルの両方が存在する場合、環境変数が使用されます。

設定ファイルの詳細については、「設定ファイル」『SQL Anywhere サーバー データベース管理』を参照してください。

設定ファイル内のパスワードなどの情報を保護する場合は、ファイル難読化ユーティリティを使用して、設定ファイルの内容を難読化できます。

「ファイル非表示ユーティリティ (dbfhide)」『SQL Anywhere サーバー データベース管理』を参照してください。

-a dbmlsync オプション

通常、特定の種類のエラー (間違った Mobile Link パスワードなど) が発生すると、正しい値の入力をユーザーに要求するウィンドウが dbmlsync から表示されます。-a オプションは、このようなエラーが発生しても dbmlsync がプロンプトを表示しないようにします。

構文

dbmlsync -a ...

-ap dbmlsync オプション

Mobile Link サーバーで authenticate_parameters スクリプトと認証パラメーターに渡されるパラメーターを指定します。

構文

dbmlsync -ap "*parameters,...***"** ...

備考

サーバーで authenticate_parameters 接続スクリプトや認証パラメーターを使用するときに使用します。次に例を示します。

```
dbmlsync -ap "parm1,parm2,parm3"
```

パラメーターは Mobile Link サーバーに送信され、authenticate_parameters スクリプトや統合データベース上のその他のイベントに渡されます。

参照

- 「認証パラメーター」『Mobile Link サーバー管理』
- 「authenticate_parameters 接続イベント」『Mobile Link サーバー管理』
- 「AuthParms 同期プロファイルオプション」179 ページ

-ba dbmlsync オプション

ダウンロードファイルを適用します。

構文

dbmlsync -ba "*filename***" ...**

備考

リモートデータベースに適用する既存のダウンロードファイルの名前を指定します。オプションでパスを指定できます。パスを指定しない場合、デフォルトロケーションは dbmlsync が起動されたディレクトリです。

参照

- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』
- 「-bc dbmlsync オプション」108 ページ
- 「-be dbmlsync オプション」109 ページ
- 「-bg dbmlsync オプション」109 ページ
- 「ApplyDnldFile 同期プロファイルオプション」180 ページ

-bc dbmlsync オプション

ダウンロードファイルを作成します。

構文

dbmlsync -bc "*filename***" ...**

備考

指定された名前でダウンロードファイルを作成します。ダウンロードファイルにはファイル拡張子 *.df* を使用してください。

オプションでパスを指定できます。パスを指定しない場合、デフォルトロケーションは dbmlsync の現在の作業ディレクトリ (dbmlsync が起動されたディレクトリ) です。

オプションで、ダウンロードファイルを作成するとき、-be オプションを使用してリモートデータベースで検証できる文字列を指定したり、-bg オプションを使用して新しいリモートデータベースのダウンロードファイルを作成したりできます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』
- 「-ba dbmlsync オプション」108 ページ
- 「-be dbmlsync オプション」109 ページ
- 「-bg dbmlsync オプション」109 ページ
- 「CreateDnldFile 同期プロファイルオプション」182 ページ

-be dbmlsync オプション

ダウンロードファイルを作成するとき、このオプションはファイルに含まれる追加の文字列を指定します。

構文

dbmlsync -bc "*filename***" -be "***string***" ...**

備考

文字列は、認証や他の目的に使用できます。文字列は、ダウンロードファイルが適用されるときに、リモートデータベース上の sp_hook_dbmlsync_validate_download_file ストアドプロシージャーに渡されます。

参照

- 「sp_hook_dbmlsync_validate_download_file」250 ページ
- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』
- 「-bc dbmlsync オプション」108 ページ
- 「-ba dbmlsync オプション」108 ページ
- 「DnldFileExtra 同期プロファイルオプション」183 ページ

-bg dbmlsync オプション

ダウンロードファイルを作成するとき、このオプションはまだ同期していないリモートデータベースで使用できるファイルを作成します。

構文

dbmlsync -bc "*filename***" -bg ...**

備考

-bg オプションを使用すると、ダウンロードファイルによってリモートデータベースの世代番号が更新されます。

このオプションを使用すると、同期していないリモートデータベースに適用できるダウンロードファイルを構築できます。このオプションを使用しない場合は、同期を行ってからダウンロードファイルを適用する必要があります。

-bg オプションで構築したダウンロードファイルは、スナップショットダウンロードです。新しいリモートデータベースの最終ダウンロードタイムスタンプは、デフォルトでは 1900 年 1 月 1 日になっており、これはダウンロードファイル内の最終ダウンロードタイムスタンプより前となるため、タイムスタンプベースのダウンロードは同期していないリモートデータベースと連携しません。タイムスタンプベースでファイルベースのダウンロードが動作するには、ダウンロードファイル内の最終ダウンロードタイムスタンプがリモートデータベースと同じか、それより前である必要があります。

このオプションは、世代番号による機能を回避するため、そのような機能に依存するシステムの場合は、すでに同期されたリモートデータベースに -bg ダウンロードファイルを適用しないでください。

参照

- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』
- 「-ba dbmlsync オプション」 108 ページ
- 「-bc dbmlsync オプション」 108 ページ
- 「Mobile Link の世代番号」『Mobile Link サーバー管理』
- 「新しいリモートの同期」『Mobile Link サーバー管理』
- 「UpdateGenNum 同期プロファイルオプション」 193 ページ

-bk dbmlsync オプション

バックグラウンド同期を有効にします。

構文

dbmlsync -bk "*connection-string***"** ...

備考

バックグラウンド同期中に、dbmlsync でロックされたデータベースリソースへのアクセスを別の接続が待機している場合、データベースエンジンはリモートデータベースへの dbmlsync 接続を削除し、コミットされていない dbmlsync 操作をロールバックします。これにより、他の接続は同期の完了を待機しないで先に進むことができます。接続の切断時に dbmlsync で未処理だった操作により、データベースが dbmlsync のコミットされていない変更をロールバックするときに、待機している接続に大幅な遅延が発生する場合があります。

dbmlsync 接続が削除されると、進行中の同期は失敗し、エラーが通知されます。

参照

- 「-bkr dbmlsync オプション」 111 ページ
- 「Mobile Link 同期プロファイル」 176 ページ
- 「Background 同期プロファイルオプション」 180 ページ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-bkr dbmlsync オプション

バックグラウンド同期が中断された後の dbmlsync の動作を制御します。

構文

dbmlsync -bkr *num*...

備考

num は -1 以上の整数です。

num が -1 の場合、dbmlsync は、成功または失敗にかかわらず、中断された同期が完了するまで中断されることなく再試行します。*num* が 0 の場合、dbmlsync は中断された同期を再試行しません。*num* が 0 より大きい場合、dbmlsync は、同期が完了するまで *num* 回だけ再試行します。*num* 回試行しても同期が完了しない場合は、中断せずに完了させるため、フォアグラウンドの同期として実行します。

デフォルトでは BackgroundRetry は 0 です。Background オプションが TRUE に設定されていない場合に BackgroundRetry を 0 以外の値に設定すると、エラーになります。 「-bk dbmlsync オプション」110 ページを参照してください。

Dbmlsync API または SQL SYNCHRONIZE 文が使用されている場合、BackgroundRetry は無視されます。

参照

- 「-bk dbmlsync オプション」110 ページ
- 「BackgroundRetry 同期プロファイルオプション」181 ページ

-c dbmlsync オプション

リモートデータベースの接続パラメーターを指定します。

構文

dbmlsync -c "*connection-string***"** ...

備考

接続文字列では、DBA や REMOTE DBA 権限を使用して SQL Anywhere リモートデータベースに接続するための dbmlsync パーミッションを指定します。この場合は、REMOTE DBA 権限を持つユーザー ID を使用することをおすすめします。

keyword=value の形式で、複数のパラメーターをセミコロンで区切って接続文字列を指定します。いずれかのパラメーター名にスペースが含まれる場合は、接続文字列を二重引用符で囲んでください。

-c を指定しないと、[DBMLSync 設定] ウィンドウが表示されます。この接続ウィンドウのフィールドで、残りのコマンドラインオプションを指定できます。

SQL Anywhere データベースに接続するための接続パラメーターの完全なリストについては、「接続パラメーター」『SQL Anywhere サーバー データベース管理』を参照してください。

-ci dbmlsync オプション

dbmlsync キャッシュの初期サイズを設定します。

構文

dbmlsync -ci *size* [**K** | **M** | **P**]...

備考

size は、dbmlsync で同期データの格納に使用される、バイト単位の初期キャッシュサイズです。キロバイトまたはメガバイトの単位を指定するには、オプションでそれぞれサフィックス K、M を使用します。

サイズをシステムの総物理メモリ量のパーセンテージとして指定するには、0 ～ 100 の数字の後に文字 p の数字を指定します。たとえば、-ci 30p は、初期キャッシュサイズを物理メモリの 30% に設定します。

参照

- 「-cm dbmlsync オプション」113 ページ
- 「-cl dbmlsync オプション」112 ページ

-cl dbmlsync オプション

dbmlsync キャッシュファイルを縮小できる最小サイズを設定します。

構文

dbmlsync -cl *size* [**K** | **M** | **P**]...

備考

size は、dbmlsync キャッシュファイルを縮小できる、バイト単位の最小サイズです。キロバイトまたはメガバイトの単位を指定するには、オプションでそれぞれサフィックス K、M を使用します。

サイズをシステムの総物理メモリ量のパーセンテージとして指定するには、0 ～ 100 の数字の後に文字 p を指定します。たとえば、-cl 5p は、キャッシュサイズが物理メモリの 5% を下回らないようにします。

参照

- 「-cm dbmlsync オプション」113 ページ
- 「-ci dbmlsync オプション」112 ページ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-cm dbmlsync オプション

dbmlsync キャッシュの最大サイズ上限値を設定します。

構文

dbmlsync -cm *size* [**K** | **M** | **P**]...

備考

size は、dbmlsync キャッシュを拡張できる、バイト単位の最大サイズです。キロバイトまたはメガバイトの単位を指定するには、オプションでそれぞれサフィックス K、M を使用します。

サイズをシステムの総物理メモリ量のパーセンテージとして指定するには、0 ～ 100 の数字の後に文字 p を指定します。たとえば、-cm 60p は、キャッシュの最大サイズを物理メモリの 60% に制限します。

参照

- 「-ci dbmlsync オプション」 112 ページ
- 「-cl dbmlsync オプション」 112 ページ

-d dbmlsync オプション

リモートデータベースに対する競合ロックを削除します。

構文

dbmlsync -d ...

備考

同期対象のテーブルに対するロックを dbmlsync で取得する必要がある場合、別の接続でこれらのいずれかのテーブルにロックがあると、同期は失敗したり遅延したりする可能性があります。このオプションを指定すると、SQL Anywhere は競合ロックを保持するリモートデータベースへの他の接続をすべて強制的に削除するため、同期はただちに実行されます。

参照

- 「同期中の同時実行性」 90 ページ
- 「KillConnections 同期プロファイルオプション」 186 ページ

-dc dbmlsync オプション

以前に失敗したダウンロードを再起動します。

構文

dbmlsync -dc ...

備考

デフォルトでは、ダウンロード中に dbmlsync で障害が発生すると、ダウンロードデータはリモートデータベースに適用されません。ただし、受信したダウンロードの一部がリモートデバイスのテンポラリファイルに保存されているため、次に同期するときに -dc オプションを指定すると、短時間でダウンロードを完了できます。-dc オプションを指定すると、dbmlsync は前のダウンロードで受信しなかった部分をダウンロードしようとします。残りのデータをダウンロードできる場合は、完全なダウンロードがリモートデータベースに適用されます。ダウンロードできない場合は、同期に失敗します。

-dc オプションを使用した場合、アップロードされる新しいデータがあると、再起動可能なダウンロードは失敗します。

また、ContinueDownload 拡張オプションや sp_hook_dbmlsync_end フックを使用して、失敗したダウンロードを再起動することもできます。

参照

- 「ダウンロードの失敗の再開」『Mobile Link サーバー管理』
- 「ContinueDownload (cd) 拡張オプション」 145 ページ
- 「sp_hook_dbmlsync_end」 221 ページ
- 「DownloadReadSize (drs) 拡張オプション」 148 ページ
- 「ContinueDownload 同期プロファイルオプション」 182 ページ

-dl dbmlsync オプション

メッセージを dbmlsync のメッセージウィンドウまたはコマンドプロンプトに表示し、メッセージログファイルに書き込みます。

構文

dbmlsync -dl ...

備考

通常、ファイルに出力のログを取るときは、dbmlsync ウィンドウに書き込まれるよりも多くのメッセージがログファイルに書き込まれます。このオプションを使用すると、dbmlsync は通常はファイルにのみ書き込まれる情報をウィンドウにも出力します。このオプションを使用すると、同期速度が低下する場合があります。

-do dbmlsync オプション

オフライントランザクションログのスキャンを無効にします。

構文

dbmlsync -do ...

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

1 つのディレクトリに複数のデータベースのトランザクションログファイルが格納されている場合、これらのいずれかのデータベースのオフライントランザクションログファイルがなくても、dbmlsync はどのデータベースからの同期もできない可能性があります。dbmlsync は、-d オプションが指定された場合、オフライントランザクションログをスキャンせず、単一のディレクトリに他のすべてのデータベースと一緒に格納されたデータベースから同期することができます。

このオプションを使用しても、オフライントランザクションログが必要な場合、dbmlsync は同期できません。

このオプションは、-x オプションと同時に使用できません。

-drs dbmlsync オプション

再起動可能なダウンロードについて、通信障害の後に再送する必要がある最大バイト数を指定します。

構文

dbmlsync -drs *bytes* ...

備考

-drs オプションは、再起動可能なダウンロードを実行するときにだけ有用なダウンロードの読み込みサイズを指定します。

dbmlsync は、チャンク単位でダウンロードを読み込みます。ダウンロードの読み込みサイズは、このチャンクのサイズを定義します。通信エラーが発生すると、処理中のチャンク全体が失われます。エラーが発生した時点によって、失われるバイト数は 0 ～ ダウンロードの読み込みサイズ -1 のいずれかになります。たとえば、DownloadReadSize が 100 バイトで、497 バイトを読み込んだ後でエラーが発生した場合は、読み込んだ最後の 97 バイトが失われます。このようにして失われたバイト数は、ダウンロードが再起動されたときに再送信されます。

通常は、ダウンロード読み込みサイズの値を大きくすると、正常な同期でのパフォーマンスが向上しますが、エラーが発生したときに再送信されるデータが多くなります。

このオプションの一般的な用途は、通信の信頼性が低いときにデフォルトのサイズを減らすことです。

デフォルトは **32767** です。このオプションを 32767 より大きな値に設定すると、32767 が使用されます。

また、DownloadReadSize 拡張オプションを使用して、ダウンロードの読み込みサイズを指定することもできます。

参照

- 「DownloadReadSize (drs) 拡張オプション」 148 ページ
- 「ダウンロードの失敗の再開」『Mobile Link サーバー管理』
- 「ContinueDownload (cd) 拡張オプション」 145 ページ
- 「sp_hook_dbmlsync_end」 221 ページ
- 「-dc dbmlsync オプション」 113 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -drs 100
```

-ds dbmlsync オプション

ダウンロード専用同期を実行します。

構文

dbmlsync -ds ...

備考

ダウンロード専用同期が発生する場合、dbmlsync はデータベースの変更をアップロードしません。しかし、スキーマと進行オフセットについての情報はアップロードします。

さらに、dbmlsync は、ダウンロード専用同期中に、リモートデータベースでの変更が上書きされないようにします。これを実行するには、ログをスキャンし、アップロードされるのを待機している操作に関連するローを検出します。これらのローのいずれかがダウンロードによって修正されると、ダウンロードはロールバックされ、同期は失敗します。この理由で同期が失敗した場合は、問題を訂正するために完全な同期を行う必要があります。

ダウンロード専用同期で同期されたリモートがある場合、ダウンロード専用同期がスキャンするログの量を減らすために、定期的に完全な双方向同期を行ってください。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。

-ds を使用すると、ConflictRetries 拡張オプションが無視され、dbmlsync はダウンロード専用同期をリトライしません。ダウンロード専用同期は、失敗した場合、通常の同期が行われるまで失敗し続けます。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

参照

- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバー管理』
- 「DownloadOnly (ds) 拡張オプション」 147 ページ
- 「ダウンロード専用のパブリケーション」 78 ページ
- 「DownloadOnly 同期プロファイルオプション」 183 ページ

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-e dbmlsync オプション

拡張オプションを指定します。

構文

dbmlsync -e *extended-option=value*; ...

備考

拡張オプションは、長形式または省略形で指定できます。

すべての拡張オプションのリストを取得するには、dbmlsync -l オプションを使用します。

コマンドラインで -e オプションを使用して指定したオプションは、同じコマンドラインで要求したすべての同期に適用されます。たとえば、次のコマンドラインでは、拡張オプション drs=512 は sub1 と sub2 の両方の同期に適用されます。

```
dbmlsync -e "drs=512" -s sub1 -s sub2
```

拡張オプションは、dbmlsync メッセージログと SYSSYNC システムビューで確認できます。

単一のアップロードに拡張オプションを指定するには、-eu オプションを使用します。

参照

- 「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページ
- 「-l dbmlsync オプション」119 ページ
- 「-eu dbmlsync オプション」118 ページ
- 「SYSSYNC システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「sp_hook_dbmlsync_set_extended_options」239 ページ
- 「ExtOpt 同期プロファイルオプション」185 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を起動するときの拡張オプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost;dir=c:¥db¥logs"...
```

-eh dbmlsync オプション

フック関数で発生したエラーを無視します。

構文

dbmlsync -eh ...

参照

- 「IgnoreHookErrors 同期プロファイルオプション」185 ページ

-ek dbmlsync オプション

強力に暗号化されたリモートデータベースの暗号化キーをコマンドラインで直接指定できます。

構文

dbmlsync -ek *key* ...

備考

高度に暗号化されたリモートデータベースを扱う場合、データベースやトランザクションログ (オフライントランザクションなど) を使用するには、常に暗号化キーを使用します。強力な暗号化が適用されたデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

-ep dbmlsync オプション

リモートデータベースで使用する暗号化キーの入力を要求します。

構文

dbmlsync -ep ...

備考

このオプションを指定すると、暗号化キーを入力するためのウィンドウが表示されます。クリアテキストでは暗号化キーを見ることができないようにすることで、高いセキュリティが得られます。強力な暗号化が適用されたリモートデータベースの場合、-ek または -ep のどちらかを指定します。両方同時には指定できません。強力に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

-eu dbmlsync オプション

拡張アップロードオプションを指定します。

構文

dbmlsync -s *subscription-name* **-eu** *keyword=value*;...

dbmlsync -n *publication-name* **-eu** *keyword=value*;...

備考

コマンドラインで -eu オプションを使用して指定した拡張オプションは、その直前の -n オプションまたは -s オプションで指定される同期のみに適用されます。たとえば、次のコマンドラインでは、拡張オプション eh=on は サブスクリプション sub2 の同期のみに適用されます。

```
dbmlsync -s sub1 -s sub2 -eu eh=on
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

拡張オプションが複数設定された場合の処理方法および拡張オプションの完全なリストについては、「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページを参照してください。

-is dbmlsync オプション

Schedule 拡張オプションを無視します。

構文

dbmlsync -is ...

備考

同期をスケジュールする拡張オプションを無視します。

スケジュールの詳細については、「同期のスケジュール」95 ページを参照してください。

参照

- 「IgnoreScheduling 同期プロファイルオプション」186 ページ

-k dbmlsync オプション (旧式)

同期が終了したときに dbmlsync を停止します。 -c または -ot オプションが一緒に指定されていないかぎり、同期中にエラーが発生しても dbmlsync は停止しません。

このオプションは推奨されなくなりました。代わりに -qc を使用してください。

構文

dbmlsync -k ...

参照

- 「-qc dbmlsync オプション」126 ページ

-l dbmlsync オプション

使用可能な拡張オプションをリストします。

構文

dbmlsync -l ...

参照

- 「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページ

-mn dbmlsync オプション

同期する Mobile Link ユーザーの新しいパスワードを指定します。

構文

dbmlsync -mn *password* ...

備考

Mobile Link ユーザーのパスワードを変更します。

参照

- 「Mobile Link ユーザー」4 ページ
- 「MobiLinkPwd (mp) 拡張オプション」156 ページ
- 「NewMobiLinkPwd (mn) 拡張オプション」157 ページ
- 「-mp dbmlsync オプション」120 ページ
- 「NewMobiLinkPwd 同期プロファイルオプション」188 ページ

-mp dbmlsync オプション

同期する Mobile Link ユーザーのパスワードを指定します。

構文

dbmlsync -mp *password* ...

備考

Mobile Link ユーザー認証用のパスワードを指定します。

参照

- 「Mobile Link ユーザー」4 ページ
- 「MobiLinkPwd (mp) 拡張オプション」156 ページ
- 「NewMobiLinkPwd (mn) 拡張オプション」157 ページ
- 「-mn dbmlsync オプション」120 ページ

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-n dbmlsync オプション (旧式)

注意
このオプションは廃止される予定です。代わりに -s dbmlsync オプションを使用することをおすすめします。 「-s dbmlsync オプション」128 ページを参照してください。

dbmlsync -s オプションを使用するには、同期するサブスクリプションのサブスクリプション名を確認する必要があります。サブスクリプション名は次のクエリを使用して確認できます。

```
SELECT subscription_name
FROM syssync JOIN sys.syspublication
WHERE site_name = <ml_user> AND publication_name = <pub_name>;
```

<ml_user> を同期する Mobile Link ユーザーに置き換えます。この値は、dbmlsync コマンドラインの -u オプションで指定される値です。 「-u dbmlsync オプション (旧式)」132 ページを参照してください。

<pub_name> を同期されるパブリケーションの名前で置き換えます。この値は、dbmlsync コマンドラインの -n オプションで指定される値です。 「-n dbmlsync オプション (旧式)」121 ページを参照してください。

同期させるパブリケーションを指定します。

構文

dbmlsync -n *pubname ...*

備考

-n オプションを複数指定すると、複数の同期パブリケーションを同期できます。

-n を使用して複数のパブリケーションを同期するには、次の 2 つの方法があります。

- -n pub1,pub2,pub3 と指定して、1 つのアップロードで pub1、pub2、pub3 をアップロードし、その後に 1 つのダウンロードを行う。

  この方法では、各パブリケーションまたはサブスクリプションで拡張オプションを設定した場合、リスト内の最初のパブリケーションとそのサブスクリプションで設定したオプションのみが使用されます。2 番目以降のパブリケーションとサブスクリプションで設定した拡張オプションは無視されます。

- -n pub1 -n pub2 -n pub3 と指定して、pub1、pub2、pub3 の 3 つを別個の逐次同期で同期する。

  -n pub1 -n pub2 を指定するなど、連続した同期が非常に速く行われると、サーバーがまだ前の同期を処理しているときに、dbmlsync が同期の処理を開始する場合があります。この場合、2 番目の同期は、同時同期ができないことを示すエラーで失敗します。この状況が発生した場合は、sp_hook_dbmlsync_delay ストアドプロシージャーを定義して、各同期の前に遅延を作成できます。通常は数秒から 1 分の遅延で十分です。

参照

- 「sp_hook_dbmlsync_delay」210 ページ
- 「Publication 同期プロファイルオプション」190 ページ

-o dbmlsync オプション

dbmlsync メッセージログファイルの名前を指定します。

構文

dbmlsync -o *filename* ...

備考

出力をログファイルに追加します。デフォルトでは、画面に出力が表示されます。

参照

- 「-os dbmlsync オプション」 122 ページ
- 「-ot dbmlsync オプション」 122 ページ

-os dbmlsync オプション

dbmlsync メッセージログファイルの最大サイズを指定します (このサイズに達するとログの名前が変更されます)。

構文

dbmlsync -os *size* [**K** | **M** | **G**]...

備考

size には、dbmlsync メッセージログファイルの最大サイズを、バイト単位で指定します。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれサフィックス k、m、または g を使用してください。デフォルトでは、サイズは無制限となります。最小のサイズ制限は 10K です。

dbmlsync ユーティリティは、現在のファイルサイズを確認してから、ファイルに出力メッセージのログを取ります。ログメッセージのファイルサイズが指定したサイズを超えた場合、dbmlsync ユーティリティは出力ファイルの名前を *yymmddxx*.dbs に変更します。*yymmdd* は年月日を表し、*xx* は 00 から始まって連続して増える数字を表します。

このオプションを使用すると、古いログファイルを手動で削除してディスク領域を解放できます。

参照

- 「-o dbmlsync オプション」 122 ページ
- 「-ot dbmlsync オプション」 122 ページ

-ot dbmlsync オプション

指定されたファイルの内容を削除してから、このファイルに出力メッセージのログを取ります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

dbmlsync -ot *logfile* ...

備考

機能は -o オプションと同じですが、dbmlsync の起動時にメッセージログファイルの内容が削除されてからメッセージが書き込まれます。

参照

- 「-o dbmlsync オプション」122 ページ
- 「-os dbmlsync オプション」122 ページ

-p dbmlsync オプション

ログスキャンのポーリングを無効にします。

構文

dbmlsync -p ...

備考

アップロードを構築するには、dbmlsync はトランザクションログをスキャンする必要があります。通常、これは同期の開始時に実行されます。しかし、同期がスケジュールされている場合、または sp_hook_dbmlsync_delay フックが使用されている場合、dbmlsync はデフォルトでは同期の前に発生する一時停止でログをスキャンします。同期が開始されるとき、ログはすでに少なくとも一部がスキャンされているため、この動作はより効率的です。このデフォルトの動作は、ログスキャンのポーリングと呼ばれます。

ログスキャンのポーリングはデフォルトではオンになっていますが、Schedule 拡張オプションを使用して同期がスケジュールされているとき、または sp_hook_dbmlsync_delay フックが使用されているときしか有効になりません。有効な場合は、設定された間隔でポーリングが行われます。デフォルトではポーリング間隔は 1 分ですが、dbmlsync -pp オプションで変更できます。

このオプションは、拡張オプションの DisablePolling=on とまったく同じです。

参照

- 「DisablePolling (p) 拡張オプション」146 ページ
- 「PollingPeriod (pp) 拡張オプション」159 ページ
- 「-pp dbmlsync オプション」126 ページ

-pc+ dbmlsync オプション

同期と同期の間で、Mobile Link サーバーへの永続的な接続を維持します。

構文

dbmlsync -pc+ ...

備考

このオプションを指定すると、dbmlsync は通常どおり Mobile Link サーバーに接続しますが、以降の同期で使用できるようにその接続を開いたままにします。永続的な接続は、次のいずれかが発生すると閉じます。

- エラーが発生し、同期に失敗した場合。
- 活性チェックのタイムアウトが発生した場合。

  「timeout」55 ページを参照してください。
- 同期が、異なる通信タイプまたはアドレスで開始された場合。つまり、設定が異なったり (別のホストが指定された場合など)、異なる方法で指定された場合 (同じホストとポートが指定されたが、順序が異なる場合など) です。

永続的な接続が閉じている場合は、新しい接続 (永続的) が開きます。

このオプションが最も役に立つのは、クライアントが高い頻度で同期するために、サーバーへの接続確立コストが高くなっている場合です。

デフォルトでは、永続的な接続は維持されません。

-pd dbmlsync オプション

Windows Mobile 用の指定された DLL をプリロードします。

構文

dbmlsync -pd *dllname*;...

備考

dbmlsync を Windows Mobile で実行するときに暗号化された通信ストリームを使用している場合は、-pd オプションを使用して起動時に適切な DLL がロードされるようにしてください。そうしなかった場合、dbmlsync では必要になるまで DLL をロードしません。Windows Mobile ではリソースが制限されるため、後で DLL をロードするとエラーが発生しやすくなります。

次に、各通信プロトコル用にロードする必要がある DLL を示します。

プロトコル	DLL
ECC	mlcecc12.dll
RSA	mlcrsa12.dll
FIPS	mlcrsafips12.dll

複数の DLL は、セミコロンで区切ったリストで指定します。次に例を示します。

```
-pd mlcrsafips12.dll;mlcrsa12.dll
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-pi dbmlsync オプション

Mobile Link サーバーを ping します。

構文

dbmlsync -pi -c *connection_string* ...

備考

-pi を使用すると、dbmlsync はリモートデータベースに接続し、Mobile Link サーバーへの接続に必要な情報を取り出し、サーバーに接続し、指定した Mobile Link ユーザーを認証します。Mobile Link サーバーは、ping を受信すると、統合データベースに接続し、ユーザーを認証し、ユーザー認証ステータスと値をクライアントに送信します。Mobile Link サーバーがコマンドラインオプション -zu+ を指定して実行されていると、Mobile Link ユーザー名が ml_user システムテーブルに見つからない場合は Mobile Link サーバーがユーザーを ml_user Mobile Link システムテーブルに追加します。

適切に接続をテストするには、dbmlsync との同期に使用するすべての同期オプションと一緒に -pi オプションを使用します。-pi を使用すると、dbmlsync は同期を実行しません。

ping に成功した場合、Mobile Link サーバーは情報メッセージを発行します。ping に失敗した場合は、エラーメッセージを発行します。

-pi を指定して dbmlsync を起動した場合、Mobile Link サーバーが実行できるのは、次のスクリプト (存在する場合) だけです。

- begin_connection
- authenticate_user
- authenticate_user_hashed
- authenticate_parameters
- end_connection

参照

- 「Ping 同期プロファイルオプション」189 ページ

-po dbmlsync オプション

dbmlsync がサーバーモードの場合、このオプションは、dbmlsync がクライアントから接続を受信するポートを指定します。

構文

dbmlsync -po *port number* ...

備考

このオプションは、-sm を指定した場合にのみ使用できます。

参照

- 「-sm dbmlsync オプション」 129 ページ

-pp dbmlsync オプション

ログスキャンの頻度を指定します。

構文

dbmlsync -pp *number* **[h | m | s]**...

備考

アップロードを構築するには、dbmlsync はトランザクションログをスキャンする必要があります。通常、これは同期の開始時に実行されます。しかし、同期がスケジュールされている場合、または sp_hook_dbmlsync_delay フックが使用されている場合、dbmlsync はデフォルトでは同期の前に発生する一時停止でログをスキャンします。同期が開始されるとき、ログはすでに少なくとも一部がスキャンされているため、この動作はより効率的です。このデフォルトの動作は、ログスキャンのポーリングと呼ばれます。

このオプションは、ログスキャンの間隔を指定します。サフィックス s、m、h、d を使用して、それぞれ秒、分、時間、日を指定します。デフォルトは **1** 分です。サフィックスを指定しない場合、デフォルトの時間単位は分です。

参照

- 「PollingPeriod (pp) 拡張オプション」 159 ページ
- 「DisablePolling (p) 拡張オプション」 146 ページ
- 「-p dbmlsync オプション」 123 ページ

-q dbmlsync オプション

Mobile Link 同期クライアントを最小化ウィンドウで起動します。

構文

dbmlsync -q ...

-qc dbmlsync オプション

同期が終了したときに dbmlsync を停止します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

dbmlsync -qc ...

備考

このオプションを使用すると、同期が成功するか、-o オプションまたは -ot オプションを使用してメッセージログファイルが指定されている場合に、同期の完了後に dbmlsync を終了します。

参照

- 「-o dbmlsync オプション」 122 ページ
- 「-ot dbmlsync オプション」 122 ページ

-qi dbmlsync オプション

dbmlsync システムトレイアイコンとメッセージウィンドウを表示するかどうかを制御します。

構文

dbmlsync -qi ...

備考

このオプションは、起動エラーウィンドウを除いて、dbmlsync の実行中に視覚的な表示が出ないようにします。-o で指定したログファイルを使用してエラーを診断できます。

-qi オプションを指定すると、同期の完了後に dbmlsync は終了します。

参照

- 「-o dbmlsync オプション」 122 ページ
- 「-ot dbmlsync オプション」 122 ページ

-r dbmlsync オプション

リモートデータベースと統合データベースのオフセットが一致しないとき、リモートオフセットを使用するように指定します。

-rb オプションは、リモートオフセットが統合オフセットよりも小さい場合 (リモートデータベースがバックアップからリストアされたときなど)、リモートオフセットを使用することを示します。-r オプションは下位互換のために提供されており、-rb と同じです。-ra オプションは、リモートオフセットが統合オフセットよりも大きい場合、リモートオフセットを使用することを示します。このオプションは、非常にまれな環境だけのために提供されており、データ損失の原因となる可能性があります。

構文

dbmlsync { **-r** | **-ra** | **-rb** } ...

備考

進行オフセットの詳細については、「進行オフセット」72 ページを参照してください。

-rb　リモートデータベースがバックアップからリストアされると、デフォルトの動作がデータ損失の原因となることがあります。この場合、リモートデータベースをリストアした後、初めて dbmlsync を実行するときに -rb を指定します。-rb を使用すると、リモートデータベースに記録されたオフセットが統合データベースから取得したオフセットよりも小さい場合、リモートデータベースに記録されているオフセットからアップロードが継続されます。-rb を使用し、リモートデータベースのオフセットが統合データベースからのオフセットよりも小さくない場合、エラーがレポートされ、同期がアボートされます。

-rb オプションでは、すでにアップロードされたデータがアップロードされることがあります。これにより統合データベースで競合が起こる可能性があり、これは適切な競合解決スクリプトを使用して処理する必要があります。

-ra　-ra オプションは、非常にまれな場合にのみ使用してください。-ra を使用すると、リモートのオフセットが統合データベースから取得したオフセットよりも大きい場合、アップロードはリモートデータベースから取得したオフセットからリトライされます。-ra を使用し、リモートデータベースのオフセットが統合データベースからのオフセットよりも大きくない場合、エラーがレポートされ、同期がアボートされます。

-ra オプションの使用には注意してください。統合データベースをリストアした結果、オフセットが一致しなければ、2 つのオフセット間の差のうちリモートデータベース内で発生した変更が失われます。-ra オプションは、統合データベースがバックアップからリストアされ、リモートデータベースのトランザクションログがリモートのオフセットと同じポイントでトランケートされた場合に役立ちます。この場合、リモートデータベースからアップロードされたすべてのデータは、統合オフセットのポイントからリモートオフセットのポイントまで失われます。

参照

- 「RemoteProgressGreater 同期プロファイルオプション」190 ページ
- 「RemoteProgressLess 同期プロファイルオプション」191 ページ

-s dbmlsync オプション

同期するサブスクリプションを指定します。

構文

dbmlsync -s *subname* ...

備考

このオプションは、-n dbmlsync オプションに置き換わります。

-s を使用して複数のサブスクリプションを同期するには、次の 2 つの方法があります。

- -s sub1,sub2,sub3 と指定して、1 つのアップロードで sub1、sub2、sub3 を同期し、その後に 1 つのダウンロードを行う。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

この方法では、各サブスクリプションで拡張オプションを設定した場合、リスト内の最初のサブスクリプションで設定したオプションのみが使用されます。2 番目以降のサブスクリプションで設定した拡張オプションは無視されます。

- -s sub1 -s sub2 -s sub3 と指定して、sub1、sub2、sub3 の 3 つを別個の逐次同期で同期し、それぞれに独自のアップロードとダウンロードを行う。

  -s sub1 -s sub2 を指定するなど、連続した同期が非常に速く行われると、サーバーがまだ前の同期を処理しているときに、dbmlsync が同期の処理を開始する場合があります。この場合、2 番目の同期は、同時同期ができないことを示すエラーで失敗します。この状況が発生した場合は、sp_hook_dbmlsync_delay ストアドプロシージャーを定義して、各同期の前に遅延を作成できます。通常は数秒から 1 分の遅延で十分です。

参照

- 「sp_hook_dbmlsync_delay」 210 ページ
- 「Subscription 同期プロファイルオプション」 192 ページ

-sc dbmlsync オプション

dbmlsync が各同期の前にスキーマ情報を再ロードするように指定します。

構文

dbmlsync -sc ...

備考

バージョン 9.0 以前では、dbmlsync は各同期の前にデータベースからスキーマ情報を再ロードしていました。再ロードされた情報には、外部キー関係、パブリケーションの定義、データベースに格納された拡張オプション、データベースの設定に関する情報が含まれていました。このような情報のロードには時間がかかります。また、通常、同期と同期の間で情報が変更されることはありません。

バージョン 9.0 以降では、dbmlsync はデフォルトで起動時にのみスキーマ情報をロードします。各同期の前にこの情報をロードする場合は、-sc を指定します。

-sm dbmlsync オプション

dbmlsync がサーバーモードで起動するようにします。

構文

dbmlsync -sm ...

備考

サーバーモードの場合、Dbmlsync API または SQL SYNCHRONIZE 文を使用して dbmlsync が起動し、アプリケーションからの接続を待機します。

このオプションは、コマンドラインから dbmlsync サーバーを起動する場合にのみ使用します。

通常、dbmlsync は、Dbmlsync API または SQL SYNCHRONIZE 文を使用して直接起動されます。このオプションは、これらのいずれかのメソッドを使用する場合には「使用しません」。

参照

- 「-po dbmlsync オプション」 125 ページ

-sp dbmlsync オプション

-sp を使用すると、指定された同期プロファイルにあるオプションが、その同期に対してコマンドラインで指定されたオプションに追加されます。

構文

dbmlsync -sp *sync profile* ...

備考

コマンドラインと同期プロファイルで同等のオプションが指定された場合、コマンドラインにあるオプションが、プロファイルにあるオプションよりも優先されます。

参照

- 「CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「Mobile Link 同期プロファイル」 176 ページ

-tu dbmlsync オプション

リモートデータベースの各トランザクションを、1 つの同期内で独立したトランザクションとしてアップロードするように指定します。

構文

dbmlsync -tu ...

備考

-tu オプションを使用するときは、「トランザクションアップロード」を作成します。こうすると、dbmlsync はリモートデータベースの各トランザクションを別個のトランザクションとしてアップロードします。Mobile Link サーバーは各トランザクションを受信したときに別々に適用およびコミットします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

-tu オプションを使用すると、リモートデータベースのトランザクションの順序は、常に統合データベースで保持されます。ただし、トランザクションでの操作の順序は保持されないことがあります。これには、以下の 2 つの理由があります。

- Mobile Link は、常に外部キー関係に基づいて更新を適用します。たとえば、外部キーテーブルとプライマリキーテーブルでデータが変更されると、Mobile Link はデータの挿入をプライマリキーテーブル、外部キーテーブルの順に行いますが、データの削除は外部キーテーブル、プライマリキーテーブルの順に行います。リモートの操作がこの順序に従っていない場合は、統合データベースでは操作の順序が異なります。

- トランザクション内の操作は結合されます。つまり、1 つのトランザクションで同じローを 3 回変更しても、最後の形式のローのみがアップロードされます。

トランザクションアップロードが中断された場合、送信されなかったデータは、次の同期で送信されます。通常、正常に完了しなかったトランザクションだけが、この時点で送信されます。また、サブスクリプションの最初の同期中にアップロードが失敗した場合などは、dbmlsync はすべてのトランザクションを再送します。

-tu オプションを使用しないと、Mobile Link はリモートデータベースでのすべての変更を結合し、アップロードの 1 つのトランザクションにします。つまり、同期と同期の間に同じローを 3 回変更しても、リモートトランザクションの数に関係なく、最後の形式のローのみがアップロードされます。このデフォルトの動作は、効率がよく、多くの状況に最適です。

ただし、特定の状況では、統合データベースでリモートトランザクションを保持したい場合があります。たとえば、リモートデータベースでトランザクションが発生したときに、そのトランザクションに基づいてアクションを行うトリガーを統合データベースで定義したいことがあります。

さらに、アップロードを小さいトランザクションに分割することには利点があります。多くの統合データベースは、小さなトランザクションを対象として最適化されているため、巨大なトランザクションは効率的でなく、多数の競合が発生することがあります。また、-tu オプションを使用すると、アップロード中に通信エラーが発生してもアップロード全体を失わずに済むことがあります。-tu オプションを使用している場合にアップロードエラーが発生しても、正常にアップロードされたすべてのトランザクションが適用されます。

-tu オプションを指定すると、Mobile Link は SQL Remote とほぼ同じように動作します。主な相違点は、SQL Remote がすべての変更を発生順にリモートデータベースにレプリケートし、結合を行わないことです。このように動作させるには、リモートデータベースで各データベース操作の後にコミットしてください。

Increment 拡張オプションやスクリプト化されたアップロードに -tu を使用することはできません。

参照

- 「-tx mlsrv12 オプション」『Mobile Link サーバー管理』
- 「自己参照テーブルからのデータのアップロード」『Mobile Link サーバー管理』
- 「TransactionalUpload 同期プロファイルオプション」192 ページ

-u dbmlsync オプション (旧式)

注意
このオプションは廃止される予定です。代わりに -s dbmlsync オプションを使用することをおすすめします。 「-s dbmlsync オプション」128 ページを参照してください。

dbmlsync -s オプションを使用するには、同期するサブスクリプションのサブスクリプション名を確認する必要があります。サブスクリプション名は次のクエリを使用して確認できます。

```
SELECT subscription_name
FROM syssync JOIN sys.syspublication
WHERE site_name = <ml_user> AND publication_name = <pub_name>;
```

<ml_user> を同期する Mobile Link ユーザーに置き換えます。この値は、dbmlsync コマンドラインの -u オプションで指定される値です。 「-u dbmlsync オプション (旧式)」132 ページを参照してください。

<pub_name> を同期されるパブリケーションの名前で置き換えます。この値は、dbmlsync コマンドラインの -n オプションで指定される値です。 「-n dbmlsync オプション (旧式)」121 ページを参照してください。

Mobile Link ユーザー名を指定します。

構文

dbmlsync -u *ml_username* ...

備考

dbmlsync コマンドラインでユーザーを 1 人だけ指定できます。この場合、*ml_username* は、処理するサブスクリプションに対応する CREATE SYNCHRONIZATION SUBSCRIPTION 文の FOR 句に使用されているユーザー名です。

このオプションを -n *publication* と併用して、dbmlsync が操作するサブスクリプションを識別します。各サブスクリプションは、*ml_username*, *publication* の組み合わせでユニークに識別されます。

コマンドラインで指定できるユーザー名は 1 つだけです。1 回の処理で同期するサブスクリプションはすべて、同じユーザーと関連していなければなりません。-n オプションを使用してコマンドラインで指定した各パブリケーションにサブスクリプションが 1 つしかない場合は、-u オプションを省略できます。

参照

- 「MLUser 同期プロファイルオプション (旧式)」188 ページ

-ud dbmlsync オプション

UNIX プラットフォームの場合にのみ、dbmlsync をデーモンとして実行するよう指示します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

dbmlsync -ud ...

備考

UNIX プラットフォーム専用です。

dbmlsync をデーモンとして実行する場合は、-o または -ot オプションのいずれかを指定して、出力情報のログを取ってください。

dbmlsync をデーモンとして起動すると、現在のユーザーの umask 設定によってパーミッションが制御されます。dbmlsync に適切なパーミッションを付与するため、dbmlsync を起動する前に umask 値を設定することをおすすめします。

参照

- 「-o dbmlsync オプション」 122 ページ
- 「-ot dbmlsync オプション」 122 ページ

-ui dbmlsync オプション

X Window Server がサポートされている Linux で、使用可能なディスプレイがない場合にシェルモードで dbmlsync を起動します。

構文

dbmlsync -ui ...

備考

このオプションを使用すると、X Window で dbmlsync の起動が試みられます。それが失敗した場合は、シェルモードで起動されます。

-ui を指定すると、dbmlsync は使用可能なディスプレイを見つけようとします。X Window Server が実行されていなかったなどの理由で、使用可能なディスプレイが見つからなかった場合は、dbmlsync はシェルモードで起動されます。

-uo dbmlsync オプション

同期がアップロードだけを含むように指定します。

構文

dbmlsync -uo...

備考

アップロード専用の同期中に、dbmlsync は通常の完全な同期とまったく同じように、Mobile Link へのアップロードを準備し送信します。しかし、ダウンロードを返信する代わりに、Mobile Link はアップロードのコミットが成功したかどうかを示す確認だけを送信します。

アップロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

参照

- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバー管理』
- 「DownloadOnly (ds) 拡張オプション」147 ページ
- 「UploadOnly (uo) 拡張オプション」167 ページ
- 「UploadOnly 同期プロファイルオプション」193 ページ

-urc dbmlsync オプション

同期でアップロードされるロー数の推定値を指定します。

構文

dbmlsync -urc *row-estimate* ...

備考

パフォーマンスを改善するために、同期でアップロードするロー数の推定値を指定できます。この設定は、多数のローをアップロードするときに特に便利です。推定値が大きいほど、アップロードが高速になりますが、多くのメモリを消費します。

指定した推定値に関係なく、同期は正しく続行されます。

参照

- 「大規模なアップロードのロー数の推定」『Mobile Link サーバー管理』
- 「UploadRowCnt 同期プロファイルオプション」194 ページ

-ux dbmlsync オプション

Linux で、メッセージを表示する、dbmlsync のメッセージウィンドウを開きます。

構文

dbmlsync -ux...

備考

-ux が指定されている場合、dbmlsync は使用可能なディスプレイを見つけます。たとえば、DISPLAY 環境変数が設定されていなかったり、X Window Server が実行されていなかったりしたために、使用可能なディスプレイが見つからなかった場合、dbmlsync は起動できません。

クワイエットモードで dbmlsync のメッセージウィンドウを実行するには、-q を使用します。

Windows では、dbmlsync のメッセージウィンドウが自動的に表示されます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「-q dbmlsync オプション」 126 ページ

-v dbmlsync オプション

メッセージログファイルにログを取り、同期ウィンドウに表示する情報を指定できます。冗長レベルが高すぎるとパフォーマンスに影響する可能性があるため、通常は、冗長レベルを高くするのは開発段階のみとしてください。

構文

dbmlsync -v [*levels*] ...

備考

-v オプションはメッセージログファイルと同期ウィンドウに影響します。dbmlsync コマンドラインで -o または -ot を指定すると、メッセージログが記録されるだけです。

-v を単独で指定すると、少量の情報のログが取られますが、-v を省略した場合よりも情報量は多くなります。

levels の値は次のとおりです。たとえば -vnrsu や -v+cp などのように、次のオプションを一度に 1 つまたは複数指定できます。

- **+** c と p 以外のすべてのログオプションをオンにする
- **c** ログ内の接続文字列を公開する。
- **p** ログ内の Mobile Link パスワードを公開する。
- **n** アップロードとダウンロードされたロー数のログを取る
- **o** 指定したコマンドラインオプションと拡張オプションに関する情報のログを取る
- **r** アップロードとダウンロードされたローの値のログを取る
- **s** フックスクリプトに関連するメッセージのログを取る
- **u** アップロードに関連する情報のログを取る

-v オプションと同様の機能を持つ拡張オプションがあります。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。-v コマンドラインオプションを使用すると、冗長オプションがすぐに有効になります。拡張オプションを使用する場合、最初の同期が始まるまで、冗長オプションは有効になりません。最初の同期の後、オプションの指定内容に関わらず、動作は同じになるはずです。

参照

- 「Verbose (v) 拡張オプション」168 ページ
- 「VerboseHooks (vs) 拡張オプション」169 ページ
- 「VerboseMin (vm) 拡張オプション」170 ページ
- 「VerboseOptions (vo) 拡張オプション」171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」172 ページ
- 「VerboseRowValues (vr) 拡張オプション」173 ページ
- 「-o dbmlsync オプション」122 ページ
- 「-ot dbmlsync オプション」122 ページ
- 「Verbosity 同期プロファイルオプション」194 ページ

-wc dbmlsync オプション

ウィンドウクラス名を指定します。

構文

dbmlsync -wc *class-name* ...

備考

このオプションは、スケジュールが有効になっているときや、サーバー起動同期を使用しているときなどに、停止モードの dbmlsync をウェイクアップするのに使用可能なウィンドウクラス名を指定します。

また、このウィンドウクラス名によって、Microsoft ActiveSync 同期用のアプリケーションが識別されます。Microsoft ActiveSync 同期に使用するアプリケーションの登録時に、クラス名を指定してください。

このオプションが適用されるのは、Windows だけです。

参照

- 「Microsoft ActiveSync 用 SQL Anywhere クライアントの登録」94 ページ
- 「Microsoft ActiveSync との同期」91 ページ
- 「Schedule (sch) 拡張オプション」160 ページに示されている INFINITE キーワード
- 「同期のスケジュール」95 ページ

例

```
dbmlsync -wc dbmlsync_$message_end...
```

-x dbmlsync オプション

トランザクションログの名前を変更して再起動します。

構文

dbmlsync -x [*size* [**K** | **M** | **G**]] ...

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

サイズ値は常に -x オプションと一緒に指定することを強くおすすめします。具体的には、**-x 0** を指定することをおすすめします。サイズを指定することで、潜在的なあいまいさと意図しない動作が回避されます。-x オプションをオフラインログディレクトリの直前に指定するときは、エラーまたは意図しない動作を回避するためにサイズを指定する必要があります。

オプションのサイズが指定された場合、ログが指定されたサイズ (バイト単位) より大きいと、名前が変更されます。単位をキロバイト、メガバイト、またはギガバイトで指定するには、それぞれサフィックス k、m、または g を使用してください。デフォルトのサイズは 0 です。

リモートデータベース側でバックアップを定期的に実行しないと、トランザクションログが大きくなっていきます。トランザクションログのサイズを制御するには、-x オプションを使用する代わりに、SQL Anywhere のイベントハンドラーを使用する方法があります。

参照

- 「スケジュールとイベントの使用によるタスクの自動化」『SQL Anywhere サーバー データベース管理』
- 「delete_old_logs オプション [SQL Remote]」『SQL Anywhere サーバー データベース管理』
- 「CREATE EVENT 文」『SQL Anywhere サーバー SQL リファレンス』

Mobile Link SQL Anywhere クライアントの拡張オプション

拡張オプションは、dbmlsync コマンドライン上で -e オプションまたは -eu オプションを使用して指定するか、データベースに格納できます。拡張オプションをデータベースに格納するには、Sybase Central を使用するか、sp_hook_dbmlsync_set_extended_options イベントフックを使用するか、次のいずれかの文で OPTION 句を使用します。

- CREATE SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- CREATE SYNCHRONIZATION USER
- ALTER SYNCHRONIZATION USER
- 同期ユーザーを指定しない CREATE SYNCHRONIZATION SUBSCRIPTION (これによって、拡張オプションがパブリケーションに対応付けられる)

優先順位

dbmlsync は、データベースに格納されるオプションとコマンドラインで指定されるオプションを結合します。競合するオプションが指定されている場合、dbmlsync は次のようにして競合を解決します。次のリストで先に示す方法で指定されているオプションが、後に示すものよりも優先されます。

1. sp_hook_dbmlsync_set_extended_options イベントフックで指定されているオプション。

2. 拡張オプションではないコマンドラインで指定されたオプション(たとえば、-ds は -e "ds=off" を上書きします。)
3. コマンドラインで -eu オプションを使用して指定されているオプション
4. コマンドラインで -e オプションを使用して指定されているオプション
5. SQL 文または Sybase Central でサブスクリプションに対して指定されているオプション。**同期モデル展開ウィザード**を使用して Mobile Link モデルを展開すると、拡張オプションが自動的に設定され、サブスクリプションで指定されます。
6. SQL 文または Sybase Central で Mobile Link ユーザーに対して指定されているオプション
7. SQL 文または Sybase Central でパブリケーションに対して指定されているオプション

注意
この優先順位は、前述の SQL 文の TYPE と ADDRESS の各オプションで指定しているような接続パラメーターにも適用されます。

拡張オプションは、ログと SYSSYNC システムビューで確認できます。

拡張オプションを使用して同期をチューニングする方法については、「dbmlsync 拡張オプションの使用」88 ページを参照してください。

拡張オプションの概要

拡張オプションのリストは以下のとおりです。

オプション	説明
BufferDownload={**ON** | **OFF**}; ...	Mobile Link サーバーからのダウンロードを、リモートデータベースに適用する前に、すべてキャッシュに読み込むかどうかを指定します。 「BufferDownload (bd) 拡張オプション」143 ページを参照してください。
CommunicationAddress=*protocol-option*; ...	Mobile Link サーバーに接続するネットワークプロトコルオプションを指定します。 「CommunicationAddress (adr) 拡張オプション」143 ページを参照してください。
CommunicationType=*network-protocol*; ...	Mobile Link サーバーへの接続に使用するネットワークプロトコルの種類を指定します。 「CommunicationType (ctp) 拡張オプション」144 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
ConflictRetries=*number*; ...	競合のためにダウンロードが失敗した場合のリトライの回数を指定します。「ConflictRetries (cr) 拡張オプション」 145 ページを参照してください。
ContinueDownload={ **ON** | **OFF** }; ...	以前に失敗したダウンロードを再起動します。「ContinueDownload (cd) 拡張オプション」 145 ページを参照してください。
DisablePolling={**ON** | **OFF**}; ...	ログスキャンの自動ポーリングを無効にします。「DisablePolling (p) 拡張オプション」 146 ページを参照してください。
DownloadOnly={ **ON** | **OFF** }; ...	同期がダウンロード専用であることを指定します。「DownloadOnly (ds) 拡張オプション」 147 ページを参照してください。
DownloadReadSize=*number*[**K**]; ...	再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定します。「DownloadReadSize (drs) 拡張オプション」 148 ページを参照してください。
ErrorLogSendLimit=*number*[**K** | **M**]; ...	同期エラーが発生した場合、dbmlsync からサーバーに送信するリモートメッセージログファイルのサイズを指定します。「ErrorLogSendLimit (el) 拡張オプション」 149 ページを参照してください。
FireTriggers={ **ON** | **OFF** }; ...	ダウンロードが適用されたときにリモートデータベースでトリガーが起動されるように指定します。「FireTriggers (ft) 拡張オプション」 151 ページを参照してください。
HoverRescanThreshold=*number*[**K** | **M**]; ...	スケジュールを使用している場合、再スキャンの実行までに累積可能な廃棄メモリ量を制限します。「HoverRescanThreshold (hrt) 拡張オプション」 151 ページを参照してください。
IgnoreHookErrors={ **ON** | **OFF** }; ...	フック関数内で発生したエラーを無視するように指定します。「IgnoreHookErrors (eh) 拡張オプション」 152 ページを参照してください。
IgnoreScheduling={ **ON** | **OFF** }; ...	Schedule 拡張オプションを無視するように指定します。「IgnoreScheduling (isc) 拡張オプション」 152 ページを参照してください。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
Increment=*number*[**K** | **M**]; ...	インクリメンタルアップロードを有効にし、インクリメンタルアップロードのサイズを制御します。 「Increment (inc) 拡張オプション」 153 ページを参照してください。
LockTables={ **ON** | **OFF** | **SHARE** | **EXCLUSIVE** }; ...	同期されるパブリケーション内のテーブルを同期する前にロックするよう指定します。 「LockTables (lt) 拡張オプション」 154 ページを参照してください。
MirrorLogDirectory=*dir*; ...	古いトランザクションログのミラーファイルを削除できるようにその場所を指定します。 「MirrorLogDirectory (mld) 拡張オプション」 155 ページを参照してください。
MobiLinkPwd=*password*; ...	Mobile Link パスワードを指定します。 「MobiLinkPwd (mp) 拡張オプション」 156 ページを参照してください。
NewMobiLinkPwd=*new-password*; ...	新しい Mobile Link パスワードを指定します。 「NewMobiLinkPwd (mn) 拡張オプション」 157 ページを参照してください。
NoSyncOnStartup={ **on** | **off** }; ...	dbmlsync が起動時に同期するのを防ぎます。このオプションを指定しない場合は、スケジューリングオプションにより、dbmlsync が起動時に同期されます。 「NoSyncOnStartup (nss) 拡張オプション」 157 ページを参照してください。
OfflineDirectory=*path*; ...	オフライントランザクションのログを含むパスを指定します。 「OfflineDirectory (dir) 拡張オプション」 158 ページを参照してください。
PollingPeriod=*number*[**S** | **M** | **H** | **D**]; ...	ログスキャンのポーリング周期を指定します。 「PollingPeriod (pp) 拡張オプション」 159 ページを参照してください。
Schedule=*schedule*; ...	同期のスケジュールを指定します。 「Schedule (sch) 拡張オプション」 160 ページを参照してください。
ScriptVersion=*version-name*; ...	スクリプトバージョンを指定します。 「ScriptVersion (sv) 拡張オプション」 162 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション	説明
SendColumnNames={ **ON** | **OFF** }; ...	ダイレクトローハンドリングや mlreplay で使用するために、アップロード時にカラム名が送信されるように指定します。 「SendColumnNames (scn) 拡張オプション」 162 ページを参照してください。
SendDownloadAck={ **ON** | **OFF** }; ...	クライアントからサーバーにダウンロード確認が送信されるように指定します。 「SendDownloadAck (sa) 拡張オプション」 163 ページを参照してください。
SendTriggers={ **ON** | **OFF** }; ...	アップロード時にトリガーの動作が送信されるように指定します。 「SendTriggers (st) 拡張オプション」 164 ページを参照してください。
TableOrder=*tables*; ...	アップロードでのテーブルの順序を指定します。 「TableOrder (tor) 拡張オプション」 165 ページを参照してください。
TableOrderChecking={ OFF | ON }; ...	TableOrder 拡張オプションで指定されたテーブルの順序について、参照整合性のチェックを無効にすることができます。 「TableOrderChecking (toc) 拡張オプション」 166 ページを参照してください。
UploadOnly={ **ON** | **OFF** }; ...	同期がアップロードだけを含むように指定します。 「UploadOnly (uo) 拡張オプション」 167 ページを参照してください。
Verbose={ **ON** | **OFF** }; ...	完全冗長を指定します。 「Verbose (v) 拡張オプション」 168 ページを参照してください。
VerboseHooks={ **ON** | **OFF** }; ...	フックスクリプトに関するメッセージのログを取るように指定します。 「VerboseHooks (vs) 拡張オプション」 169 ページを参照してください。
VerboseMin={ **ON** | **OFF** }; ...	少量の情報のログを取るように指定します。 「VerboseMin (vm) 拡張オプション」 170 ページを参照してください。
VerboseOptions={ **ON** | **OFF** }; ...	指定したコマンドラインオプション (拡張オプションを含む) に関する情報のログを取るように指定します。 「VerboseOptions (vo) 拡張オプション」 171 ページを参照してください。

オプション	説明
VerboseRowCounts={ ON | OFF }; ...	アップロードおよびダウンロードされるローの数のログを取るように指定します。 「VerboseRowCounts (vn) 拡張オプション」 172 ページを参照してください。
VerboseRowValues={ ON | OFF }; ...	アップロードおよびダウンロードされるローの値のログを取るように指定します。 「VerboseRowValues (vr) 拡張オプション」 173 ページを参照してください。
VerboseUpload={ ON | OFF }; ...	アップロードストリームに関する情報のログを取るように指定します。 「VerboseUpload (vu) 拡張オプション」 174 ページを参照してください。

参照

- 「-e dbmlsync オプション」 117 ページ
- 「-eu dbmlsync オプション」 118 ページ
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「SYSSYNC システムビュー」『SQL Anywhere サーバー SQL リファレンス』
- 「sp_hook_dbmlsync_set_extended_options」 239 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を起動するときの拡張オプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost;dir=c:¥db¥logs"...
```

次の SQL 文は、拡張オプションをデータベースに格納する方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
  FOR mluser
  ADDRESS 'host=localhost'
  OPTION schedule='weekday@11:30am-12:30pm', dir='c:¥db¥logs'
```

次の dbmlsync コマンドラインは、オプションとその構文をリストする使用画面を開きます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
dbmlsync -l
```

BufferDownload (bd) 拡張オプション

Mobile Link サーバーからのダウンロードを、リモートデータベースに適用する前に、すべてキャッシュに読み込むかどうかを指定します。

構文

bd={ON | OFF}; ...

BufferDownload={ON | OFF}; ...

備考

デフォルトは ON です。デフォルトを使用すると、Mobile Link サーバーへの負荷が低下し、サーバー側のスループットが向上します。

BufferDownload が OFF に設定されている場合、ダウンロードが読み込まれると dbmlsync によって適用されます。

CommunicationAddress (adr) 拡張オプション

Mobile Link サーバーに接続するネットワークプロトコルオプションを指定します。

構文

adr=*protocol-option*; ...

CommunicationAddress=*protocol-option*; ...

備考

パラメーターについては、「Mobile Link クライアントネットワークプロトコルオプション」24 ページを参照してください。

Mobile Link ユーザーのすべてのサブスクリプションが、1 つの統合データベースに対してのみ同期されていることを確認する必要があります。複数の統合データベースに同期されていると、データの損失や予期しない動作が発生する場合があります。

CommunicationType 拡張オプションを使用してネットワークプロトコルのタイプを指定します。

参照

- 「Mobile Link クライアントネットワークプロトコルオプション」24 ページ
- 「Relay Server 設定ファイル」『Relay Server』
- 「CommunicationType (ctp) 拡張オプション」144 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost"
```

コマンドラインで複数のネットワークプロトコルオプションを指定するには、それらを一重引用符で囲みます。次に例を示します。

```
dbmlsync -e "adr='host=somehost;port=5001'"
```

データベースに Address または CommunicationType を格納するには、拡張オプションを使用するか、ADDRESS 句を使用できます。次に例を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  ADDRESS 'host=localhost;port=2439'
```

CommunicationType (ctp) 拡張オプション

Mobile Link サーバーへの接続に使用するネットワークプロトコルの種類を指定します。

構文

ctp=*network-protocol*; ...

CommunicationType=*network-protocol*; ...

備考

network-protocol には、**tcpip**、**tls**、**http**、**https** のいずれかを指定します。デフォルトは **tcpip** です。

Mobile Link ユーザーのすべてのサブスクリプションが、1 つの統合データベースに対してのみ同期されていることを確認する必要があります。複数の統合データベースに同期されていると、データの損失や予期しない動作が発生する場合があります。

参照

- 「Mobile Link サーバー／クライアント通信の暗号化」『SQL Anywhere サーバー データベース管理』
- 「CommunicationAddress (adr) 拡張オプション」 143 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ctp=https"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ctp='tcpip'
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

ConflictRetries (cr) 拡張オプション

競合のためにダウンロードが失敗した場合のリトライの回数を指定します。

構文

cr=*number*; ...

ConflictRetries=*number*; ...

備考

同期時にテーブルがロックされている場合、アップロードの構築からダウンロードの適用までの間に、データベースに操作を適用することができます。ダウンロードでも変更されるローに変更が影響する場合、dbmlsync では競合として処理され、ダウンロードストリームには変更内容は適用されません。競合が発生すると、dbmlsync は同期処理全体をリトライします。通常、同期は次の試行で成功しますが、新しい競合によって新しいリトライが必要となることもあります。このオプションは、実行するリトライの最大回数を制御します。

このオプションは、LockTables オプションが OFF (デフォルトは ON) の場合にだけ役に立ちます。

デフォルトは **-1** です (リトライは無制限に続行されます)。

参照

- 「競合の解決」『Mobile Link サーバー管理』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "cr=5"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION cr='5';
```

ContinueDownload (cd) 拡張オプション

以前に失敗したダウンロードを再起動します。

構文

cd={ ON | OFF }; ...

ContinueDownload={ ON | OFF }; ...

備考

dbmlsync でサーバーからダウンロード全体を受信しないと、ダウンロードデータはリモートデータベースに適用されません。ただし、受信したダウンロードの一部はリモートデバイスのテンポラリファイルに格納されるため、後でダウンロードを再起動できます。拡張オプション cd=on を設定すると、dbmlsync はダウンロードを再起動し、前のダウンロードで受信しなかった部分をダウンロードします。残りのデータをダウンロードできる場合は、完全なダウンロードがリモートデータベースに適用されます。

cd=on を設定した場合、アップロードされる新しいデータがあると、ダウンロードを再起動せずに同期が失敗します。ダウンロードを再起動できない場合は、同期が失敗します。

また、-dc オプションや sp_hook_dbmlsync_end フックを使用して、ダウンロードを再起動することもできます。

参照

- 「ダウンロードの失敗の再開」『Mobile Link サーバー管理』
- 「sp_hook_dbmlsync_set_extended_options」239 ページ
- 「-dc dbmlsync オプション」113 ページ
- 「sp_hook_dbmlsync_end」221 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "cd=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION cd='on';
```

DisablePolling (p) 拡張オプション

ログスキャンの自動ポーリングを無効にします。

構文

p={ON | OFF}; ...

DisablePolling={ON | OFF}; ...

備考

アップロードを構築するには、dbmlsync はトランザクションログをスキャンする必要があります。通常、これは同期直前に行います。しかし、同期がスケジュールされている場合、または sp_hook_dbmlsync_delay フックが使用されている場合、dbmlsync はデフォルトにより同期の間でログをスキャンします。同期が始まるときログはすでに一部がスキャンされているので、この動作はより効率的です。このデフォルトの動作は、ログスキャンのポーリングと呼ばれます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

ログスキャンのポーリングはデフォルトでは on になっていますが、同期がスケジュールされているとき、または sp_hook_dbmlsync_delay フックが使用されているときしか効果がありません。これが有効な場合、ポーリングは決まった間隔で実行されます。dbmlsync はログの最後までスキャンし、ポーリング周期の間待機した後、ログの新しいトランザクションをスキャンします。デフォルトではポーリング周期は 1 分ですが、dbmlsync -pp オプションまたは PollingPeriod 拡張オプションで変更できます。

デフォルトでは、ログスキャンのポーリングを無効にしません **(OFF)**。

このオプションは、**dbmlsync -p** と同じです。

参照

- 「PollingPeriod (pp) 拡張オプション」 159 ページ
- 「-p dbmlsync オプション」 123 ページ
- 「-pp dbmlsync オプション」 126 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "p=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION p='on';
```

DownloadOnly (ds) 拡張オプション

同期がダウンロード専用であることを指定します。

構文

ds={ ON | OFF }; ...

DownloadOnly={ ON | OFF }; ...

備考

ダウンロード専用同期が発生する場合、dbmlsync はローの操作またはデータをアップロードしません。しかし、スキーマと進行オフセットについての情報はアップロードします。

さらに、dbmlsync は、ダウンロード専用同期中に、アップロードされていないリモートデータベースでの変更が上書きされないようにします。これを実行するには、ログをスキャンし、アップロードされるのを待機している操作に関連するローを検出します。これらのローのいずれかがダウンロードによって修正されると、ダウンロードはロールバックされ、同期は失敗します。この理由で同期が失敗した場合は、問題を訂正するために完全な同期を行う必要があります。

ダウンロード専用同期で同期されたリモートがある場合、ダウンロード専用同期がスキャンするログの量を減らすために、定期的に完全な同期を行ってください。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。これが問題になる場合は、ダウンロード専用パブリケーションを使用すると、同期中のログの問題を防ぐことができます。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

デフォルトは **OFF** です (アップロードとダウンロード両方の実行)。

参照

- 「-ds dbmlsync オプション」 116 ページ
- 「ダウンロード専用のパブリケーション」 78 ページ
- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバー管理』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ds=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ds='ON';
```

DownloadReadSize (drs) 拡張オプション

再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定します。

構文

drs=*number*[**K**]; ...

DownloadReadSize=*number*[**K**]; ...

備考

DownloadReadSize オプションが有用なのは、再起動可能なダウンロードを実行するときだけです。

ダウンロードの読み込みサイズは、バイト単位で指定します。キロバイトの単位を指定するには、サフィックス k を使用します。

dbmlsync は、チャンク単位でダウンロードを読み込みます。このチャンクのサイズを定義するのが DownloadReadSize です。通信エラーが発生すると、処理中のチャンク全体が失われます。エラーが発生した時点によって、失われるバイト数は 0 ～ DownloadReadSize -1 のいずれかにな

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

ります。たとえば、DownloadReadSize が 100 バイトで、497 バイトを読み込んだ後でエラーが発生した場合は、読み込んだ最後の 97 バイトが失われます。このようにして失われたバイト数は、ダウンロードが再起動されたときに再送信されます。

通常は、DownloadReadSize の値を大きくすると、正常な同期でのパフォーマンスが向上しますが、エラーが発生したときに再送信されるデータが多くなります。

このオプションの一般的な用途は、通信の信頼性が低いときにデフォルトのサイズを減らすことです。

デフォルトは **32767** です。このオプションを 32767 より大きな値に設定すると、32767 が使用されます。

参照

- 「-drs dbmlsync オプション」 115 ページ
- 「ダウンロードの失敗の再開」『Mobile Link サーバー管理』
- 「ContinueDownload (cd) 拡張オプション」 145 ページ
- 「sp_hook_dbmlsync_end」 221 ページ
- 「-dc dbmlsync オプション」 113 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "drs=100"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION drs='100';
```

ErrorLogSendLimit (el) 拡張オプション

同期エラーが発生した場合、dbmlsync からサーバーに送信するリモートメッセージログファイルのサイズを指定します。

構文

el=*number*[**K** | **M**]; ...

ErrorLogSendLimit=*number*[**K** | **M**]; ...

備考

このオプションはバイト単位で指定します。キロバイトまたはメガバイトの単位を指定するには、それぞれサフィックス k、m を使用します。

このオプションは、同期中にエラーが発生した場合に、dbmlsync が Mobile Link サーバーに送信するメッセージログのバイト数を指定します。dbmlsync メッセージログを送信しない場合は、このオプションを **0** に設定します。

デフォルトは **32K** です。

このオプションを 0 以外に設定すると、クライアント側のエラーが発生したときにエラーログがアップロードされます。クライアント側のすべてのエラーで、ログが送信されるわけではありません。通信エラーや、dbmlsync が Mobile Link サーバーに接続されていないときに発生したエラーでは、ログは送信されません。アップロード送信後にエラーが発生した場合、エラーログがアップロードされるのは、SendDownloadAck 拡張オプションが ON に設定された場合だけです。

ErrorLogSendLimit に十分な大きさの値を設定すると、dbmlsync は現在のセッションのメッセージログ全体を、Mobile Link サーバーに送信します。たとえば、メッセージログのメッセージが古いメッセージログファイルに追加された場合、dbmlsync は現在のセッション中に生成された新しいメッセージだけを送信します。新しいメッセージ全体の長さが ErrorLogSendLimit を超える場合、dbmlsync は指定されたサイズまでのメッセージログのみをアップロードします。

メッセージログのサイズは、指定されている冗長性の設定の影響を受けます。冗長性を調節するには、dbmlsync -v オプションを使用するか、"verbose" で始まる dbmlsync 拡張オプションを使用します。詳細については、次に示す -e 冗長性オプションを参照してください。

参照

- 「-v dbmlsync オプション」135 ページ
- 「-e dbmlsync オプション」117 ページ
- 「-eu dbmlsync オプション」118 ページ
- 「Verbose (v) 拡張オプション」168 ページ
- 「VerboseHooks (vs) 拡張オプション」169 ページ
- 「VerboseMin (vm) 拡張オプション」170 ページ
- 「VerboseOptions (vo) 拡張オプション」171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」172 ページ
- 「VerboseRowValues (vr) 拡張オプション」173 ページ
- 「VerboseUpload (vu) 拡張オプション」174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "el=32k"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION el='32k';
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

FireTriggers (ft) 拡張オプション

ダウンロードが適用されたときにリモートデータベースでトリガーが起動されるように指定します。

構文

ft={ ON | OFF }; ...

FireTriggers={ ON | OFF }; ...

備考

デフォルトは **ON** です。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ft=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ft='off';
```

HoverRescanThreshold (hrt) 拡張オプション

スケジュールを使用している場合、再スキャンの実行までに累積可能な廃棄メモリ量を制限します。

構文

hrt=*number*[**K** | **M**]; ...

HoverRescanThreshold=*number*[**K** | **M**]; ...

備考

メモリをバイト単位で指定します。キロバイトまたはメガバイトの単位を指定するには、それぞれサフィックス k、m を使用します。デフォルトは **1m** です。

コマンドラインで複数の -n オプションまたは -s オプションを指定すると、メモリ破棄の原因となる断片化が dbmlsync で起きる可能性があります。破棄されたメモリは、データベーストランザクションログの再スキャンによってのみリカバリできます。このオプションでは、ログを再スキャンしてメモリをリカバリするまでに累積可能な廃棄メモリ量を制限できます。破棄されたメモリのリカバリを制御するもう 1 つの方法は、sp_hook_dbmlsync_log_rescan ストアドプロシージャーを実行することです。

参照

- 「sp_hook_dbmlsync_log_rescan」225 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "hrt=2m"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION hrt='2m';
```

IgnoreHookErrors (eh) 拡張オプション

フック関数内で発生したエラーを無視するように指定します。

構文

eh={ ON | OFF }; ...

IgnoreHookErrors={ ON | OFF }; ...

備考

デフォルトは **OFF** です。

このオプションは、dbmlsync -eh オプションと同じです。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "eh=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION eh='off';
```

IgnoreScheduling (isc) 拡張オプション

Schedule 拡張オプションを無視するように指定します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

isc={ ON | OFF }; ...

IgnoreScheduling={ ON | OFF }; ...

備考

ON に設定すると、dbmlsync は Schedule 拡張オプションを無視して、すぐに同期を行います。デフォルトは **OFF** です。

このオプションは、dbmlsync -is オプションと同じです。

参照

- 「同期のスケジュール」95 ページ
- 「Schedule (sch) 拡張オプション」160 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "isc=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION isc='off';
```

Increment (inc) 拡張オプション

インクリメンタルアップロードを有効にし、インクリメンタルアップロードのサイズを制御します。

構文

inc=*number***[K | M]; ...**

Increment=*number***[K | M]; ...**

備考

このオプションの値は、大まかな各アップロード部分のサイズをバイト単位で指定します。キロバイトまたはメガバイトの単位を指定するには、それぞれサフィックス k、m を使用します。

このオプションをゼロ以外の値に設定すると、アップロードは 1 つ以上の部分に分けて Mobile Link に送信されます。これは、dbmlsync が完全なアップロードを完了するだけの長い時間、Mobile Link サーバーへの接続を維持できない場合に役立ちます。デフォルトは 0 です。

オプションの値は、次のように各アップロード部分のサイズを制御します。dbmlsync は、データベーストランザクションログをスキャンして、アップロードを構築します。このオプションを

指定すると、dbmlsync はオプションで設定されたバイト数をスキャンし、未処理トランザクションがない最初の時点、つまりすべてのトランザクションがコミットまたはロールバックされた次の時点までスキャンを続けます。dbmlsync は、スキャンした部分をアップロード部分として送信し、中断したところからログのスキャンを再開します。

Increment 拡張オプションは、スクリプト化されたアップロードやトランザクション単位のアップロードでは使用できません。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "inc=32000"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION inc='32k';
```

LockTables (lt) 拡張オプション

同期されるパブリケーション内のテーブルを同期する前にロックするよう指定します。

構文

lt={ ON | OFF | SHARE | EXCLUSIVE }; ...

LockTables={ ON | OFF | SHARE | EXCLUSIVE }; ...

備考

SHARE は、dbmlsync が共有モードのすべての同期テーブルをロックすることを意味します。EXCLUSIVE は、dbmlsync が排他モードのすべての同期テーブルをロックすることを意味します。Windows Mobile 以外のプラットフォームでは、ON は SHARE と同じです。Windows Mobile デバイスでは、ON は EXCLUSIVE と同じです。

デフォルトは OFF です。つまり、デフォルトでは、次の場合を除いて dbmlsync は同期テーブルをロックしません。

- 現在の同期でスクリプトベースのアップロードを使用するパブリケーションがある場合、およびリモートデータベースで sp_hook_dbmlsync_schema_upgrade フックが定義されている場合、dbmlsync は同期テーブルを SHARE でロックします。

同期中の修正を禁止するには、ON に設定します。

共有ロックと排他ロックの詳細については、「ロックの仕組み」『SQL Anywhere サーバー SQL の使用法』と「LOCK TABLE 文」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Mobile Link アプリケーションでのテーブルのロックの詳細については、「同期中の同時実行性」 90 ページを参照してください。

同期テーブルが排他モードでロックされているとき (Windows Mobile デバイスではデフォルト)、他の接続はそのテーブルにアクセスできません。このため、別個の接続で実行する dbmlsync ストアドプロシージャーは、同期テーブルのいずれかにアクセスする必要がある場合は実行できません。

別個の接続で実行するフックについては、「SQL Anywhere クライアントのイベントフック」 195 ページを参照してください。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "lt=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION lt='on';
```

MirrorLogDirectory (mld) 拡張オプション

古いトランザクションログのミラーファイルを削除できるようにその場所を指定します。

構文

mld=*dir*; ...

MirrorLogDirectory=*dir*; ...

備考

このオプションを指定すると、次のどちらかの状況になった場合に、dbmlsync は古いトランザクションログミラーファイルを削除できます。

- オフライントランザクションログミラーが、トランザクションログミラーとは異なるディレクトリに置かれる

  または

- dbmlsync がリモートデータベースサーバーとは異なるコンピューターで実行されている

通常の設定では、アクティブなトランザクションログミラーと名前の変更されたトランザクションログミラーファイルは同じディレクトリに置かれ、dbmlsync はリモートデータベースと同じコンピューターで実行されるため、このオプションは不要であり、古いトランザクションログミラーファイルは自動的に削除されます。

このディレクトリ内のトランザクションログが影響を受けるのは、delete_old_logs データベースオプションが On、Delay、または *n* 日に設定されている場合だけです。

参照

- 「delete_old_logs オプション [SQL Remote]」『SQL Anywhere サーバー データベース管理』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mld=c:¥tmp¥file"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mld='c:¥tmp¥file';
```

MobiLinkPwd (mp) 拡張オプション

Mobile Link パスワードを指定します。

構文

mp=*password*; ...

MobiLinkPwd=*password*; ...

備考

Mobile Link サーバーに接続するためのパスワードを指定します。このパスワードは、サブスクリプションが同期されている Mobile Link ユーザーの正しいパスワードにする必要があります。デフォルト値は NULL です。

Mobile Link ユーザーがすでにパスワードを持っている場合は、拡張オプション **-e mn** で変更できます。

参照

- 「NewMobiLinkPwd (mn) 拡張オプション」 157 ページ
- 「-mn dbmlsync オプション」 120 ページ
- 「-mp dbmlsync オプション」 120 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mp=password"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mp='password';
```

NewMobiLinkPwd (mn) 拡張オプション

新しい Mobile Link パスワードを指定します。

構文

mn=*new-password*; ...

NewMobiLinkPwd=*new-password*; ...

備考

サブスクリプションが同期されている Mobile Link ユーザーの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。デフォルトでは、パスワードは変更されません。

参照

- 「MobiLinkPwd (mp) 拡張オプション」156 ページ
- 「-mn dbmlsync オプション」120 ページ
- 「-mp dbmlsync オプション」120 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mp=oldpassword;mn=newpassword"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mp='oldpassword';mn='newpassword'
```

NoSyncOnStartup (nss) 拡張オプション

dbmlsync が起動時に同期するのを防ぎます。このオプションを指定しない場合は、スケジューリングオプションにより、dbmlsync が起動時に同期されます。

構文

nss={ on | off }; ...

NoSyncOnStartup={ on | off }; ...

備考

このオプションが有効なのは、スケジュール拡張オプションが EVERY または INFINITE 句と一緒に使用されている場合だけです。これらのスケジュールオプションでは、dbmlsync は起動時に自動的に同期します。

デフォルトは off です。

NoSyncOnStartup を on に設定し、INFINITE 句と一緒にスケジュールを使用すると、ウィンドウメッセージが受信されるまで同期は行われません。

NoSyncOnStartup を on に設定して、EVERY 句と一緒にスケジュールを使用すると、起動後の最初の同期は、EVERY 句で指定した期間の経過後に行われます。

この設定は、dbmlsync 起動時以外、スケジュールの動作に影響することはありません。

参照

- 「Schedule (sch) 拡張オプション」 160 ページ
- 「同期のスケジュール」 95 ページ

例

次の dbmlsync コマンドラインの一部は、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "schedule=EVERY:01:00;nss=off"...
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION nss='off', schedule='EVERY:01:00';
```

OfflineDirectory (dir) 拡張オプション

オフライントランザクションのログを含むパスを指定します。

構文

dir=*path*; ...

OfflineDirectory=*path*; ...

備考

dbmlsync はデフォルトで、オンライントランザクションログと同じディレクトリ内で、名前の変更されたログを確認できます。このオプションは、名前の変更されたオフライントランザクションログが別のディレクトリにある場合にのみ指定する必要があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "dir=c:¥db¥logs"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION dir='c:¥db¥logs';
```

PollingPeriod (pp) 拡張オプション

ログスキャンのポーリング周期を指定します。

構文

pp=*number*[**S** | **M** | **H** | **D**]; ...

PollingPeriod=*number*[**S** | **M** | **H** | **D**]; ...

備考

このオプションは、ログスキャンの間隔を指定します。サフィックス s、m、h、d を使用して、それぞれ秒、分、時間、日を指定します。デフォルトは **1** 分です。サフィックスを指定しない場合、デフォルトの時間単位は分です。

ログスキャンのポーリングは、同期をスケジュールしているとき、または sp_hook_dbmlsync_delay フックを使用しているときだけ実行されます。

ログスキャンのポーリングの説明は、「DisablePolling (p) 拡張オプション」146 ページを参照してください。

このオプションは、**dbmlsync -pp** と同じです。

参照

- 「DisablePolling (p) 拡張オプション」146 ページ
- 「-pp dbmlsync オプション」126 ページ
- 「-p dbmlsync オプション」123 ページ
- 「sp_hook_dbmlsync_delay」210 ページ
- 「Schedule (sch) 拡張オプション」160 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "pp=5"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION pp='5';
```

Schedule (sch) 拡張オプション

同期のスケジュールを指定します。

構文

sch=*schedule*; ...

Schedule=*schedule*; ...

schedule : { **EVERY**:*hhhh*:*mm* | **INFINITE** | *singleSchedule* }

hhhh : **00** ... **9999**

mm : **00** ... **59**

singleSchedule : *day* **@***hh*:*mm*[**AM** | **PM**] [**-***hh*:*mm*[**AM** | **PM**]] ,...

hh : **00** ... **24**

mm : **00** ... **59**

day :
EVERYDAY | **WEEKDAY** | **MON** | **TUE** | **WED** | **THU** | **FRI** | **SAT** | **SUN** | *dayOfMonth*

dayOfMonth : **0**... **31**

備考

EVERY　EVERY キーワードを指定すると、同期は起動時に発生し、その後は指定した期間の経過後に無限に繰り返されます。指定した期間より同期処理に時間がかかる場合、同期はすぐに再開されます。

dbmlsync の起動時に同期が発生するのを防ぐには、拡張オプション NoSyncOnStartup を使用します。

singleSchedule　1 つ以上の単一スケジュールを指定すると、同期は指定した日時にのみ発生します。

間隔は *@hh*:*mm*-*hh*:*mm* (AM または PM はオプション指定) のように指定します。AM または PM を指定しない場合は、24 時間制とみなされます。24:00 は翌日の午前 00:00 とみなされます。間隔を指定すると、同期は間隔中の任意の時点から始まります。間隔を指定すると同期を行う時間帯に幅ができるため、同じスケジュールを持つ複数のリモートデータベースが、まったく同じ時刻に同期を行うことがなく、Mobile Link サーバー側では輻輳が生じなくなります。

間隔の終了時刻は常に開始時刻より後の時刻として解釈されます。間隔に真夜中が含まれている場合は、翌日に終了します。dbmlsync が間隔の途中で始まる場合、同期は終了時刻より前の任意の時点で発生します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

EVERYDAY　EVERYDAY とは週の 7 日間すべてのことです。

WEEKDAY　WEEKDAY とは月曜日から金曜日までのことです。Monday のような完全形も使用できます。使用言語が英語でない場合、クライアントによって接続文字列で要求された言語でない場合、サーバーメッセージウィンドウに表示される言語でない場合は、完全形を使用します。

dayOfMonth　月の長さに関係なく、月の最終日を指定するには、*dayOfMonth* を 0 に設定します。

INFINITE　INFINITE キーワードを指定すると、dbmlsync の同期は起動時に発生します。その後同期が発生するのは、ウィンドウメッセージを dbmlsync に送信する別のプログラムから同期が開始されたときだけです。dbmlsync 拡張オプション NoSyncOnStartup を使用すると、初期同期を防ぐことができます。

このオプションを dbmlsync -wc オプションと組み合わせて使用すると、dbmlsync をウェイクアップし、同期を行うことができます。

直前の同期がスケジュール時刻にまだ完了していない場合は、その同期が完了した時点で、スケジュールされた同期が開始されます。

デフォルトは、スケジュール設定なしです。

Dbmlsync API が使用されている場合、または SQL SYNCHRONIZE 文が使用されている場合、Schedule オプションは無視されます。

IgnoreScheduling 拡張オプションと dbmlsync -is オプションは、dbmlsync がスケジュールを無視して即時の同期を行えるようにします。

参照

- 「NoSyncOnStartup (nss) 拡張オプション」 157 ページ
- 「-wc dbmlsync オプション」 136 ページ
- 「同期のスケジュール」 95 ページ
- 「IgnoreScheduling (isc) 拡張オプション」 152 ページ
- 「-is dbmlsync オプション」 119 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sch=WEEKDAY@8:00am,SUN@9:00pm"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sch='WEEKDAY@8:00am,SUN@9:00pm';
```

ScriptVersion (sv) 拡張オプション

スクリプトバージョンを指定します。

警告
スクリプトバージョンの指定には、ScriptVersion 拡張オプションではなく、CREATE SYNCHRONIZATION SUBSCRIPTION 文と ALTER SYNCHRONIZATION SUBSCRIPTION 文の SCRIPT VERSION 句を使用することを強くおすすめします。これは、SCRIPT VERSION 句を使用すると、スキーマのアップグレードを行う場合の問題が大幅に単純化されるからです。

ScriptVersion (sv) 拡張オプションは、SCRIPT VERSION 句を使用して格納された値を上書きします。したがって、ScriptVersion (sv) 拡張オプションは、下位互換のためにのみ使用するか、同期に使用するスクリプトバージョンを明示的に指定する必要があるようなまれな場合にのみ使用してください。

「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』と「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

構文

sv=*version-name*; ...

ScriptVersion=*version-name*; ...

備考

スクリプトバージョンは、同期中に統合データベースの Mobile Link によって実行されるスクリプトを決定します。デフォルトのスクリプトバージョンは **default** です。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sv=SysAd001"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sv='SysAd001';
```

SendColumnNames (scn) 拡張オプション

ダイレクトローハンドリングや mlreplay で使用するために、アップロード時にカラム名が送信されるように指定します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

構文

scn={ ON | OFF }; ...

SendColumnNames={ ON | OFF }; ...

備考

デフォルトは **ON** です。

カラム名はデフォルトで送信されるため、ほとんどの配備では ml_add_column ストアドプロシージャーは必要ありません。ml_add_column 名を使用する場合、クライアントから送信される名前は無視されます。

カラム名は、Mobile Link サーバーによってダイレクトローハンドリングで使用されます。ダイレクトローハンドリング API を使用して、インデックスではなく、名前でカラムを参照する場合は、このオプションを **ON** に設定してください。

mlreplay ユーティリティを使用している場合は、リプレイ API のカラム名が意味のあるものになるように、このオプションを **ON** に設定します。

参照

- 「ml_add_column system procedure」『Mobile Link サーバー管理』
- 「ダイレクトローハンドリング」『Mobile Link サーバー管理』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "scn=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION scn='on';
```

SendDownloadAck (sa) 拡張オプション

クライアントからサーバーにダウンロード確認が送信されるように指定します。

構文

sa={ ON | OFF }; ...

SendDownloadAck={ ON | OFF }; ...

備考

ダウンロード確認を使用すると、リモートデータベースにダウンロードが適用されたことを Mobile Link サーバーで確実に判断できます。統合データベースで同期スクリプトを作成して、

確認を処理し、確認時にビジネスロジックを実行できます。クライアントがダウンロードを適用した後でネットワークセッションが削除されると、ダウンロード確認が送信されないことがあるため、この可能性をスクリプトで許可する必要があります。 「nonblocking_download_ack 接続イベント」『Mobile Link サーバー管理』を参照してください。

注意 : SendDownloadAck が ON に設定され、冗長モードである場合、受信確認行はクライアントログに書き込まれます。

デフォルトは **OFF** です。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sa=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sa='on';
```

SendTriggers (st) 拡張オプション

アップロード時にトリガーの動作が送信されるように指定します。

構文

st={ ON | OFF }; ...

SendTriggers={ ON | OFF }; ...

備考

カスケード削除もトリガーの動作と見なされます。

デフォルトは **OFF** です。

2 つのサブスクリプションに同じテーブルが 1 つ以上含まれている場合は、SendTriggers オプションについては同じ設定を使用して、両方のサブスクリプションを同期する必要があります。

> **注意**
> 同期のダウンロードフェーズの一部として行われたデータベースの変更の結果生じるトリガーアクションは、SendTriggers オプションの値にかかわらず、同期されることはありません。

SendTrigger オプションが **ON** に設定されている場合にトリガーでの操作が同期されるようにするには、そのトリガーでの操作が、同期されているパブリケーションのテーブルに対して行われる必要があります。トリガーを起動した基本の操作もパブリケーション内に存在することが必要なわけではありません。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

dbmlsync は単一のローに対して行われる操作を結合しますが、SendTrigger オプションが **ON** に設定されている場合は、トリガーを起動した基本の操作が別の操作に結合されていても、トリガー内で起動されるすべての操作が送信されます。

外部キーに対する組み込み参照整合性アクション (ON DELETE CASCADE と ON UPDATE CASCADE) はトリガーの動作と見なされ、SendTrigger オプションが **ON** に設定されないかぎり同期されません。1 つの例外は、参照整合性アクションが、すでにアップロードストリーム内にあるローに対して行われる場合です。この場合、システム生成トリガー内の参照整合性アクションは、次の例で示すように、すでにアップロードストリーム内にあるローのステータスと結合されます。

この例では、**Parent** テーブルと **Child** テーブル間に外部キーがあり、参照整合性アクション ON DELETE CASCADE が使用されます。

```
INSERT INTO Parent (pid, pname) values ( 100, 'Amy' );
INSERT INTO Child (cid, pid, cname) values ( 2000, 100, 'Alex' );
COMMIT;
DELETE FROM Parent WHERE pid = 100;
COMMIT;
```

Parent テーブルからロー 100 を 削除すると、参照整合性アクションによって **Child** テーブルからもロー 2000 が削除されることに注意してください。この時点で dbmlsync が実行された場合、**Parent** テーブルのロー 100 での INSERT とその後の DELETE によって、ローは同期されなくなります。ただし、SendTrigger オプションが **OFF** に設定されている場合、**Child** テーブルへの挿入は送信されます。この場合、dbmlsync は **Child** テーブルのロー 2000 をすでにアップロードストリームに追加しているため、**Parent** テーブルのロー 100 の削除での参照整合性アクション (これにより **Child** テーブルのロー 2000 が削除される) は、アップロードストリーム内の既存のローと結合されます。その結果、どちらのローも同期されなくなり、両方の側でデータの一貫性が維持されます。

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "st=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION st='on';
```

TableOrder (tor) 拡張オプション

アップロードでのテーブルの順序を指定します。

構文

tor=*tables*; ...

TableOrder=*tables*; ...

tables = *table-name* [,*table-name*], ...

備考

このオプションでは、テーブルをアップロードする順序は指定できます。アップロードされるすべてのテーブルを指定する必要があります。同期に含まれていないテーブルを指定すると、そのテーブルは無視されます。

指定するテーブルの順序は、参照整合性を保つようにします。つまり、Table1 が Table2 を外部キー参照する場合、Table2 は Table1 の前にアップロードする必要があります。適切な順序でテーブルを指定しないと、次の 2 つの場合を除いてエラーが発生します。

- TableOrderChecking=OFF を設定した場合。
- テーブルが環状外部キーに関連付けられている場合(この場合、ルールを満たす順序はないので、循環されるテーブルはどんな順序でもアップロードできます)

TableOrder を指定しないと、dbmlsync は、参照整合性を満たす順序を選択します。

ダウンロードのテーブルの順序は、アップロードと同じです。アップロードテーブルの順序の制御により、サーバー側のスクリプトを簡単に記述することができます。特に、リモートデータベースと統合データベースの外部キー制約が異なる場合に効果を発揮します。

参照

- 「TableOrderChecking (toc) 拡張オプション」 166 ページ
- 「アップロードの処理方法」『Mobile Link クイックスタート』
- 「参照整合性と同期」『Mobile Link クイックスタート』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "tor=admin,primary,foreign"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION tor='admin,primary,foreign;
```

TableOrderChecking (toc) 拡張オプション

TableOrder 拡張オプションで指定されたテーブルの順序について、参照整合性のチェックを無効にすることができます。

構文

toc={ OFF | ON }; ...

TableOrderChecking={ OFF | ON }; ...

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

ほとんどのアプリケーションで、リモートデータベースと統合データベースのテーブルには、同じ外部キーが関連付けられています。このような場合、TableOrderChecking をデフォルトの値 ON のままにしておくと、dbmlsync によって、テーブルが別のテーブルに対する外部キーを持つときに、別のテーブルよりも前にアップロードされないようにすることができます。これにより、参照整合性が確保されます。

このオプションは、統合データベースとリモートデータベースの外部キーの関係が異なる場合に便利です。TableOrder 拡張オプションと一緒に使用すると、リモートでの参照整合性制約に違反していても、サーバーでの参照整合性制約を満たすテーブルの順序を指定できます。

このオプションは、TableOrder 拡張オプションが指定された場合のみ役立ちます。

デフォルトは **ON** です。

参照

- 「TableOrder (tor) 拡張オプション」 165 ページ
- 「アップロードの処理方法」『Mobile Link クイックスタート』
- 「参照整合性と同期」『Mobile Link クイックスタート』

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "toc=OFF"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION toc='Off';
```

UploadOnly (uo) 拡張オプション

同期がアップロードだけを含むように指定します。

構文

uo={ ON | OFF }; ...

UploadOnly={ ON | OFF }; ...

備考

アップロード専用の同期中に、dbmlsync は通常の完全な同期とまったく同じように、Mobile Link サーバーへのアップロードを準備し送信します。しかし、ダウンロードを返信する代わりに、Mobile Link サーバーはアップロードのコミットが成功したかどうかを示す確認だけを送信します。

アップロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

デフォルトは **OFF** です。

参照

- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link サーバー管理』
- 「DownloadOnly (ds) 拡張オプション」 147 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "uo=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION uo='on';
```

Verbose (v) 拡張オプション

完全冗長を指定します。

構文

v={ ON | OFF }; ...

Verbose={ ON | OFF }; ...

備考

このオプションが指定する高いレベルの冗長性は、パフォーマンスに影響する場合があるので、通常は開発段階にだけ使用してください。

このオプションは、**dbmlsync -v+** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「-v dbmlsync オプション」 135 ページ
- 「VerboseHooks (vs) 拡張オプション」 169 ページ
- 「VerboseMin (vm) 拡張オプション」 170 ページ
- 「VerboseOptions (vo) 拡張オプション」 171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 172 ページ
- 「VerboseRowValues (vr) 拡張オプション」 173 ページ
- 「VerboseUpload (vu) 拡張オプション」 174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "v=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION v='on';
```

VerboseHooks (vs) 拡張オプション

フックスクリプトに関するメッセージのログを取るように指定します。

構文

vs={ ON | OFF }; ...

VerboseHooks={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -vs** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

参照

- 「-v dbmlsync オプション」135 ページ
- 「SQL Anywhere クライアントのイベントフック」195 ページ
- 「Verbose (v) 拡張オプション」168 ページ
- 「VerboseMin (vm) 拡張オプション」170 ページ
- 「VerboseOptions (vo) 拡張オプション」171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」172 ページ
- 「VerboseRowValues (vr) 拡張オプション」173 ページ
- 「VerboseUpload (vu) 拡張オプション」174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vs=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vs='on';
```

VerboseMin (vm) 拡張オプション

少量の情報のログを取るように指定します。

構文

vm={ ON | OFF }; ...

VerboseMin={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -v** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「-v dbmlsync オプション」 135 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「VerboseOptions (vo) 拡張オプション」 171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 172 ページ
- 「VerboseRowValues (vr) 拡張オプション」 173 ページ
- 「VerboseUpload (vu) 拡張オプション」 174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vm=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vm='on';
```

VerboseOptions (vo) 拡張オプション

指定したコマンドラインオプション (拡張オプションを含む) に関する情報のログを取るように指定します。

構文

vo={ ON | OFF }; ...

VerboseOptions={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -vo** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

参照

- 「-v dbmlsync オプション」 135 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「VerboseMin (vm) 拡張オプション」 170 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 172 ページ
- 「VerboseRowValues (vr) 拡張オプション」 173 ページ
- 「VerboseUpload (vu) 拡張オプション」 174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vo=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vo='on';
```

VerboseRowCounts (vn) 拡張オプション

アップロードおよびダウンロードされるローの数のログを取るように指定します。

構文

vn={ ON | OFF }; ...

VerboseRowCounts={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -vn** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

参照

- 「-v dbmlsync オプション」 135 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「VerboseMin (vm) 拡張オプション」 170 ページ
- 「VerboseOptions (vo) 拡張オプション」 171 ページ
- 「VerboseRowValues (vr) 拡張オプション」 173 ページ
- 「VerboseUpload (vu) 拡張オプション」 174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vn=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vn='on';
```

VerboseRowValues (vr) 拡張オプション

アップロードおよびダウンロードされるローの値のログを取るように指定します。

構文

vr={ ON | OFF }; ...

VerboseRowValues={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -vr** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

参照

- 「-v dbmlsync オプション」 135 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「VerboseMin (vm) 拡張オプション」 170 ページ
- 「VerboseOptions (vo) 拡張オプション」 171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 172 ページ
- 「VerboseUpload (vu) 拡張オプション」 174 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vr=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vr='on';
```

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

VerboseUpload (vu) 拡張オプション

アップロードストリームに関する情報のログを取るように指定します。

構文

vu={ ON | OFF }; ...

VerboseUpload={ ON | OFF }; ...

備考

このオプションは、**dbmlsync -vu** と同じです。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。拡張オプションでロギングの冗長性を設定しても、ロギングは直ちに有効にならないため、起動情報のログは取られません。最初の同期が行われる時点では、-v オプションを指定した場合と拡張オプションを指定した場合のロギングの動作は同じです。

デフォルトは **OFF** です。

参照

- 「-v dbmlsync オプション」 135 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「Verbose (v) 拡張オプション」 168 ページ
- 「VerboseMin (vm) 拡張オプション」 170 ページ
- 「VerboseOptions (vo) 拡張オプション」 171 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 172 ページ
- 「VerboseRowValues (vr) 拡張オプション」 173 ページ

例

次の dbmlsync コマンドラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vu=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION vu='on';
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Mobile Link SQL 文

次に、Mobile Link SQL Anywhere クライアントの構成と実行に使用する SQL 文を示します。

- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「GRANT REMOTE DBA 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「START SYNCHRONIZATION DELETE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「STOP SYNCHRONIZATION DELETE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「STOP SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「SYNCHRONIZE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

Ultra Light クライアント

「Ultra Light SQL 文」『Ultra Light データベース管理とリファレンス』を参照してください。

Mobile Link 同期プロファイル

同期プロファイルを使用すると、いくつかの dbmlsync オプションをデータベースに配置できます。作成するプロファイルには、さまざまな同期オプションを含めることができます。

同期プロファイルを作成、変更、削除するには、次の文を使用します。

- 「CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』

同期プロファイルには、dbmlsync -sp オプション、Dbmlsync API の Sync メソッド、SQL SYNCHRONIZE 文、リモートデータベースの集中管理で作成されたリモートタスクからアクセスできます。いずれの場合も、同期プロファイルのオプションに統合される追加オプションを指定する機能があります。指定した追加オプションと同期プロファイルで指定されたオプションが競合する場合は、追加オプションで指定した値が使用されます。

dbmlsync -sp オプションを使用する場合、コマンドラインの他のすべてのオプションは追加オプションとして処理されます。他のインターフェイスには、追加オプションを指定するための特定のパラメーターまたは句があります。

Ultra Light での同期プロファイルの使用については、「同期プロファイルオプション」『Ultra Light データベース管理とリファレンス』を参照してください。

同期プロファイルでは次のオプションを指定できます。

オプション名	省略名	指定可能な値	説明
AuthParms	ap	文字列	authenticate_parameters スクリプトと認証パラメーターにパラメーターを入力します。「AuthParms 同期プロファイルオプション」179 ページを参照してください。
ApplyDnldFile	ba	文字列	ダウンロードファイルを適用します。「ApplyDnldFile 同期プロファイルオプション」180 ページを参照してください。
Background	bk	文字列	TRUE に設定されている場合、バックグラウンド同期を有効にします。「Background 同期プロファイルオプション」180 ページを参照してください。
BackgroundRetry	bkr	整数	バックグラウンド同期が中断された後の dbmlsync の動作方法を制御します。「BackgroundRetry 同期プロファイルオプション」181 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション名	省略名	指定可能な値	説明
ContinueDownload	dc	ブール式	以前に失敗したダウンロードを再起動します。「ContinueDownload 同期プロファイルオプション」182 ページを参照してください。
CreateDnldFile	bc	文字列	ダウンロードファイルを作成します。「CreateDnldFile 同期プロファイルオプション」182 ページを参照してください。
DnldFileExtra	be	文字列	ダウンロードファイルを作成するとき、このオプションはファイルに含まれる追加の文字列を指定します。「DnldFileExtra 同期プロファイルオプション」183 ページを参照してください。
DownloadOnly	ds	ブール式	ダウンロード専用同期を実行します。「DownloadOnly 同期プロファイルオプション」183 ページを参照してください。
DownloadReadSize	drs	整数	再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定します。「DownloadReadSize 同期プロファイルオプション」184 ページを参照してください。
ExtOpt	e	文字列	拡張オプションを指定します。「ExtOpt 同期プロファイルオプション」185 ページを参照してください。
IgnoreHookErrors	eh	ブール式	フック関数で発生したエラーを無視します。「IgnoreHookErrors 同期プロファイルオプション」185 ページを参照してください。
IgnoreScheduling	is	ブール式	スケジュールの指示を無視して、直ちに同期を行います。「IgnoreScheduling 同期プロファイルオプション」186 ページを参照してください。
KillConnections	d	ブール式	リモートデータベースに対する競合ロックを削除します。「KillConnections 同期プロファイルオプション」186 ページを参照してください。
LogRenameSize	x	整数。オプションで後ろに K または M が付きます。	アップロードデータがスキャンされた後、トランザクションログの名前を変更し、再起動します。「LogRenameSize 同期プロファイルオプション」187 ページを参照してください。

オプション名	省略名	指定可能な値	説明
MobiLinkPwd	mp	文字列	Mobile Link ユーザーのパスワードを指定します。「MobiLinkPwd 同期プロファイルオプション」 187 ページを参照してください。
MLUser	u	文字列	Mobile Link ユーザー名を指定します。「MLUser 同期プロファイルオプション (旧式)」 188 ページを参照してください。
NewMobiLinkPwd	mn	文字列	Mobile Link ユーザーの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。「NewMobiLinkPwd 同期プロファイルオプション」 188 ページを参照してください。
Ping	pi	ブール式	Mobile Link サーバーに対して ping を実行し、クライアントと Mobile Link 間の通信を確認します。「Ping 同期プロファイルオプション」 189 ページを参照してください。
Publication	n	文字列	このオプションは推奨されなくなりました。同期させるパブリケーションを指定します。パブリケーションは同期プロファイル内で 1 回しか指定できませんが、コマンドラインオプションは複数回指定できます。「Publication 同期プロファイルオプション」 190 ページを参照してください。
RemoteProgressGreater	ra	ブール式	リモートオフセットが統合オフセットよりも大きい場合にリモートオフセットが使用されるように指定します。これは、-ra オプションと同じです。「RemoteProgressGreater 同期プロファイルオプション」 190 ページを参照してください。
RemoteProgressLess	rb	ブール式	リモートオフセットが統合オフセットよりも小さい場合に (リモートデータベースがバックアップからリストアされたときなど) リモートオフセットが使用されるように指定します。これは、-rb オプションと同じです。「RemoteProgressLess 同期プロファイルオプション」 191 ページを参照してください。
Subscription	s	文字列	同期させるサブスクリプションを指定します。サブスクリプションは、同期プロファイルで 1 回しか指定できません。「Subscription 同期プロファイルオプション」 192 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプション名	省略名	指定可能な値	説明
TransactionalUpload	tu	ブール式	リモートデータベースの各トランザクションを、1つの同期内で独立したトランザクションとしてアップロードするように指定します。「TransactionalUpload 同期プロファイルオプション」192 ページを参照してください。
UpdateGenNum	bg	ブール式	ダウンロードファイルを作成するとき、このオプションはまだ同期していないリモートデータベースで使用できるファイルを作成します。「UpdateGenNum 同期プロファイルオプション」193 ページを参照してください。
UploadOnly	uo	ブール式	同期がアップロードだけを含み、ダウンロードが発生しないように指定します。 「UploadOnly 同期プロファイルオプション」193 ページを参照してください。
UploadRowCnt	urc	整数	同期でアップロードされるロー数の推定値を指定します。 「UploadRowCnt 同期プロファイルオプション」194 ページを参照してください。
Verbosity		文字列 (オプションのカンマ区切りのリスト)	dbmlsync の冗長性を制御します。 「Verbosity 同期プロファイルオプション」194 ページに似ています。 値は、次のオプションの 1 つ以上を含むカンマ区切りのリストにします。次に示すように、各オプションは既存の -v オプションに対応しています。 ● BASIC は -v と同じです。 ● HIGH は -v+ と同じです。 ● CONNECT_STR は -vc と同じです。 ● ROW_CNT は -vn と同じです。 ● OPTIONS は -vo と同じです。 ● ML_PASSWORD は -vp と同じです。 ● ROW_DATA は -vr と同じです。

AuthParms 同期プロファイルオプション

authenticate_parameters スクリプトと認証パラメーターにパラメーターを入力します。

構文

ap=*parameters*

Authparms=*parameters*

備考

authenticate_parameters 接続スクリプトや認証パラメーターを使用するときに使用します。

パラメーターは Mobile Link サーバーに送信され、authenticate_parameters スクリプトや統合データベース上のその他のイベントに渡されます。

参照

- 「-ap dbmlsync オプション」 107 ページ

例

```
AuthParms=p1,p2,p3
```

ApplyDnldFile 同期プロファイルオプション

ダウンロードファイルを適用します。

構文

ba=*filename*

ApplyDnldFile=*filename*

備考

リモートデータベースに適用する既存のダウンロードファイルの名前を指定します。

参照

- 「-ba dbmlsync オプション」 108 ページ

例

```
ApplyDnldFile=filename
```

Background 同期プロファイルオプション

ON に設定されている場合、バックグラウンド同期を有効にします。

構文

bk={ON|OFF}

Background={ON|OFF}

備考

バックグラウンド同期中に、dbmlsync でロックされたデータベースリソースへのアクセスを別の接続が待機している場合、データベースエンジンはリモートデータベースへの dbmlsync 接続を削除し、コミットされていない dbmlsync 操作をロールバックします。これにより、他の接続は同期の完了を待機しないで先に進むことができます。接続の切断時に dbmlsync で未処理だっ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

た操作により、データベースが dbmlsync のコミットされていない変更をロールバックするときに、待機している接続に大幅な遅延が発生する場合があります。

参照

- 「-bk dbmlsync オプション」 110 ページ

例

次の例は、Background 同期プロファイルオプションを使用する方法を示します。

```
Background=on
```

BackgroundRetry 同期プロファイルオプション

整数。バックグラウンド同期が中断された後の dbmlsync の動作方法を制御します。

構文

bkr=*integer*

BackgroundRetry=*integer*

備考

このオプションには省略形と長形式があり、bkr または BackgroundRetry を使用できます。

この値は -1 以上の整数に設定します。値が -1 の場合、dbmlsync は、成功または失敗にかかわらず、中断された同期が完了するまで中断されることなく再試行します。値が 0 の場合、dbmlsync は中断された同期を再試行しません。値が 0 より大きい場合、dbmlsync は、同期が完了するまで指定された回数だけ再試行します。指定された回数を試行しても同期が完了しない場合は、中断せずに完了させるため、フォアグラウンドの同期として実行します。

デフォルトでは BackgroundRetry は 0 です。

Background オプションが ON に設定されていない場合に BackgroundRetry を 0 以外の値に設定すると、エラーになります。

Dbmlsync API または SQL SYNCHRONIZE 文を使用して同期が開始されている場合、このオプションは無視されます。

参照

- 「-bkr dbmlsync オプション」 111 ページ

例

次の例は、BackgroundRetry 同期プロファイルオプションを使用する方法を示します。

```
BackgroundRetry=4
```

ContinueDownload 同期プロファイルオプション

以前に失敗したダウンロードを再起動します。

構文

dc={ON|OFF}

ContinueDownload={ON|OFF}

備考

デフォルトでは、ダウンロード中に Mobile Link で障害が発生すると、ダウンロードデータはリモートデータベースに適用されません。ただし、受信したダウンロードの一部がリモートデバイスのテンポラリファイルに保存されているため、次に dbmlsync を同期するときに ContinueDownload=on を指定すると、短時間でダウンロードを完了できます。

ContinueDownload=on を指定すると、dbmlsync は前のダウンロードで受信しなかった部分をダウンロードしようとします。残りのデータをダウンロードできる場合は、完全なダウンロードがリモートデータベースに適用されます。ダウンロードできない場合は、同期に失敗します。

ContinueDownload=on を設定した場合、アップロードされる新しいデータがあると、再起動可能なダウンロードは失敗します。また、ContinueDownload 拡張オプションや sp_hook_dbmlsync_end フックを使用して、失敗したダウンロードを再起動することもできます。

参照

- 「-dc dbmlsync オプション」 113 ページ
- 「sp_hook_dbmlsync_end」 221 ページ
- 「ContinueDownload (cd) 拡張オプション」 145 ページ

例

次の例は、ContinueDownload 同期プロファイルオプションを使用する方法を示します。

```
ContinueDownload=on
```

CreateDnldFile 同期プロファイルオプション

指定された名前でダウンロードファイルを作成します。

構文

bc=*filename*

CreateDnldFile=*filename*

備考

ダウンロードファイルにはファイル拡張子 *.df* を使用してください。

オプションでパスを指定できます。パスを指定しない場合、デフォルトロケーションは dbmlsync の現在の作業ディレクトリ (dbmlsync が起動されたディレクトリ) です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オプションで、ダウンロードファイルを作成するとき、-be オプションを使用してリモートデータベースで検証できる文字列を指定したり、-bg オプションを使用して新しいリモートデータベースのダウンロードファイルを作成したりできます。

参照

- 「-bc dbmlsync オプション」 108 ページ
- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』

例

```
CreateDnldFile=dnldl.df
```

DnldFileExtra 同期プロファイルオプション

ダウンロードファイルを作成するとき、このオプションはファイルに含まれる追加の文字列を指定します。

構文

be=*string*

DnldFileExtra=*string*

備考

文字列は、認証や他の目的に使用できます。文字列は、ダウンロードファイルが適用されるときに、リモートデータベース上の sp_hook_dbmlsync_validate_download_file ストアドプロシージャーに渡されます。

文字列にセミコロンを含めることはできません。

参照

- 「-be dbmlsync オプション」 109 ページ
- 「sp_hook_dbmlsync_validate_download_file」 250 ページ

例

次の例は、DnldFileExtra 同期プロファイルオプションを使用する方法を示します。

```
DnldFileExtra=val1,val2,val3
```

DownloadOnly 同期プロファイルオプション

on に設定されている場合、ダウンロード専用同期を実行します。

構文

ds={ON|OFF}

DownloadOnly={ON|OFF}

備考

ダウンロード専用同期が発生する場合、dbmlsync はデータベースの変更をアップロードしません。しかし、スキーマと進行オフセットについての情報はアップロードします。

さらに、dbmlsync は、ダウンロード専用同期中に、リモートデータベースでの変更が上書きされないようにします。これを実行するには、ログをスキャンし、アップロードされるのを待機している操作に関連するローを検出します。これらのローのいずれかがダウンロードによって修正されると、ダウンロードはロールバックされ、同期は失敗します。この理由で同期が失敗した場合は、問題を訂正するために完全な同期を行う必要があります。

ダウンロード専用同期で同期されたリモートがある場合、ダウンロード専用同期がスキャンするログの量を減らすために、定期的に完全な双方向同期を行ってください。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。

参照

- 「-ds dbmlsync オプション」 116 ページ
- 「DownloadOnly (ds) 拡張オプション」 147 ページ

例

次の例は、DownloadOnly 同期プロファイルオプションを使用する方法を示します。

```
DownloadOnly=on
```

DownloadReadSize 同期プロファイルオプション

再起動可能なダウンロードについて、通信障害の後に再送する必要がある最大バイト数を指定します。

構文

drs=*size*

DownloadReadSize=*size*

備考

dbmlsync は、チャンク単位でダウンロードを読み込みます。ダウンロードの読み込みサイズは、このチャンクのサイズを定義します。通信エラーが発生すると、処理中のチャンク全体が失われます。エラーが発生した時点によって、失われるバイト数は 0 ～ ダウンロードの読み込みサイズ -1 のいずれかになります。たとえば、DownloadReadSize が 100 バイトで、497 バイトを読み込んだ後でエラーが発生した場合は、読み込んだ最後の 97 バイトが失われます。このようにして失われたバイト数は、ダウンロードが再起動されたときに再送信されます。

通常は、ダウンロード読み込みサイズの値を大きくすると、正常な同期でのパフォーマンスが向上しますが、エラーが発生したときに再送信されるデータが多くなります。このオプションの一般的な用途は、通信の信頼性が低いときにデフォルトのサイズを減らすことです。

デフォルトは 32767 です。このオプションを 32767 より大きな値に設定すると、32767 が使用されます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

また、DownloadReadSize 拡張オプションを使用して、ダウンロードの読み込みサイズを指定することもできます。

参照

- 「-drs dbmlsync オプション」115 ページ
- 「DownloadReadSize (drs) 拡張オプション」148 ページ

例

次の例は、DownloadReadSize 同期プロファイルオプションを使用する方法を示します。

```
DownloadReadSize=100
```

ExtOpt 同期プロファイルオプション

拡張オプションを指定します。

構文

e={*option=value***; ...}**

ExtOpt={*option=value***; ...}**

備考

拡張オプションは、長形式または省略形で指定できます。「Mobile Link SQL Anywhere クライアントの拡張オプション」137 ページを参照してください。

拡張オプションは、「オプション＝値」のペアで指定します。設定する拡張オプションのリストは、中カッコ {} で囲む必要があります。

参照

- 「-e dbmlsync オプション」117 ページ

例

次の例は、ExtOpt 同期プロファイルオプションを使用する方法を示します。

```
ExtOpt={lt=exclusive;tableorder=t1,t2,t3}
```

IgnoreHookErrors 同期プロファイルオプション

on に設定されている場合、フック関数で発生したエラーを無視します。

構文

eh={ON|OFF}

IgnoreHookErrors={ON|OFF}

参照

- 「-eh dbmlsync オプション」 117 ページ

例

次の例は、IgnoreHookErrors 同期プロファイルオプションを使用する方法を示します。

```
IgnoreHookErrors=on
```

IgnoreScheduling 同期プロファイルオプション

Schedule 拡張オプションを無視して、直ちに同期を行います。

構文

is={ON|OFF}

IgnoreScheduling={ON|OFF}

備考

このオプションには省略形と長形式があり、is または IgnoreScheduling を使用できます。

参照

- 「-is dbmlsync オプション」 119 ページ
- 「同期のスケジュール」 95 ページ

例

次の例は、IgnoreScheduling 同期プロファイルオプションを使用する方法を示します。

```
IgnoreScheduling=on
```

KillConnections 同期プロファイルオプション

リモートデータベースに対する競合ロックを削除します。

構文

d={ON|OFF}

KillConnections={ON|OFF}

備考

同期対象のテーブルに対するロックを dbmlsync で取得する必要がある場合、別の接続でこれらのいずれかのテーブルにロックがあると、同期は失敗したり遅延したりする可能性があります。このオプションを指定すると、SQL Anywhere は競合ロックを保持するリモートデータベースへの他の接続をすべて強制的に削除するため、同期はただちに実行されます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「-d dbmlsync オプション」113 ページ
- 「同期中の同時実行性」90 ページ

例

次の例は、KillConnections 同期プロファイルオプションを使用する方法を示します。

```
KillConnections=on
```

LogRenameSize 同期プロファイルオプション

トランザクションログの名前を変更して再起動します。

構文

x=*size*[**K** | **M**]

LogRenameSize=*size*[**K** | **M**]

備考

このオプションを設定すると、トランザクションログが指定されたサイズ (バイト単位) より大きい場合に、同期中に名前が変更されて再起動されます。キロバイトまたはメガバイトの単位を指定するには、それぞれサフィックス K、M を使用します。

サイズに関係なくトランザクションログの名前を変更するには、サイズを 0 に設定します。

参照

- 「-x dbmlsync オプション」136 ページ

例

次の例は、LogRenameSize 同期プロファイルオプションを使用する方法を示します。

```
LogRenameSize=512k
```

MobiLinkPwd 同期プロファイルオプション

Mobile Link ユーザーのパスワードを指定します。

構文

mp=*password*

MobiLinkPwd=*password*

参照

- 「-mp dbmlsync オプション」 120 ページ
- 「MobiLinkPwd (mp) 拡張オプション」 156 ページ
- 「NewMobiLinkPwd (mn) 拡張オプション」 157 ページ
- 「-mn dbmlsync オプション」 120 ページ

例

次の例は、MobiLinkPwd 同期プロファイルオプションを使用する方法を示します。

```
MobiLinkPwd=mypassword
```

MLUser 同期プロファイルオプション (旧式)

Mobile Link ユーザー名を指定します。

構文

u=*username*

MLUser=*username*

備考

このオプションは推奨されなくなりました。代わりに Subscription 同期プロファイルオプションを使用します。

参照

- 「Subscription 同期プロファイルオプション」 192 ページ
- 「Mobile Link ユーザー」 4 ページ
- 「-u dbmlsync オプション (旧式)」 132 ページ

例

次の例は、MLUser 同期プロファイルオプションを使用する方法を示します。

```
MLUser=my_user_name
```

NewMobiLinkPwd 同期プロファイルオプション

Mobile Link ユーザーの新しいパスワードを指定します。このオプションは、既存のパスワードを変更する場合に使用します。

構文

mn=*new_password*

NewMobiLinkPwd=*new_password*

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「-mp dbmlsync オプション」 120 ページ
- 「-mn dbmlsync オプション」 120 ページ
- 「Mobile Link ユーザー」 4 ページ

例

次の例は、NewMobiLinkPwd 同期プロファイルオプションを使用する方法を示します。

```
NewMobiLinkPwd=new_password
```

Ping 同期プロファイルオプション

Mobile Link サーバーを ping します。

構文

pi={ON|OFF}

Ping={ON|OFF}

備考

ping 同期プロファイルオプションを on に設定すると、dbmlsync はリモートデータベースに接続し、Mobile Link サーバーへの接続に必要な情報を取り出し、Mobile Link サーバーに接続し、指定した Mobile Link ユーザーを認証します。

Mobile Link サーバーは、ping を受信すると、統合データベースに接続し、ユーザーを認証し、ユーザー認証ステータスと値をクライアントに送信します。Mobile Link サーバーがコマンドラインオプション -zu+ を指定して実行されていると、Mobile Link ユーザー名が ml_user システムテーブルに見つからない場合は Mobile Link サーバーがユーザーを ml_user Mobile Link システムテーブルに追加します。

適切に接続をテストするには、dbmlsync との同期に使用するすべての同期オプションと一緒に ping 同期プロファイルオプションを使用します。ping 同期プロファイルオプションが含まれている場合、dbmlsync は同期を実行しません。

ping に成功した場合、Mobile Link サーバーは情報メッセージを発行します。ping に失敗した場合は、エラーメッセージを発行します。

参照

- 「-pi dbmlsync オプション」 125 ページ

例

次の例は、Ping 同期プロファイルオプションを使用する方法を示します。

```
Ping=on
```

Publication 同期プロファイルオプション

同期させるパブリケーションを指定します。

構文

n=*pubname*, ...

Publication=*pubname*, ...

備考

publication 同期プロファイルオプションは、同期プロファイルで 1 回しか指定できません。

注意
このオプションは廃止される予定です。代わりに Subscription 同期プロファイルオプションまたは -s dbmlsync オプションを使用することをお勧めします。「Subscription 同期プロファイルオプション」192 ページと「-s dbmlsync オプション」128 ページを参照してください。

dbmlsync -s オプションを使用するには、同期するサブスクリプションのサブスクリプション名を確認する必要があります。サブスクリプション名は次のクエリを使用して確認できます。

```
SELECT subscription_name
FROM syssync JOIN sys.syspublication
WHERE site_name = <ml_user> AND publication_name = <pub_name>;
```

<ml_user> を同期する Mobile Link ユーザーに置き換えます。この値は、dbmlsync コマンドラインの -u オプションで指定される値です。 「-u dbmlsync オプション (旧式)」132 ページを参照してください。

<pub_name> を同期されるパブリケーションの名前で置き換えます。この値は、dbmlsync コマンドラインの -n オプションで指定される値です。 「-n dbmlsync オプション (旧式)」121 ページを参照してください。

参照

● 「-n dbmlsync オプション (旧式)」121 ページ

例

```
Publication=overnight
```

RemoteProgressGreater 同期プロファイルオプション

リモートオフセットが統合オフセットよりも大きい場合にリモートオフセットが使用されるように指定します。これは、-ra オプションと同じです。

構文

ra={ON|OFF}

RemoteProgressGreater={ON|OFF}

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

このオプションは、非常にまれな場合にのみ使用してください。このオプションを使用すると、リモートのオフセットが統合データベースから取得したオフセットよりも大きい場合、アップロードはリモートデータベースから取得したオフセットからリトライされます。このオプションを使用し、リモートデータベースのオフセットが統合データベースからのオフセットよりも大きくない場合、エラーがレポートされ、同期がアボートされます。

このオプションの使用には十分注意してください。統合データベースをリストアした結果、オフセットが一致しなければ、2 つのオフセット間の差のうちリモートデータベース内で発生した変更が失われます。このオプションは、統合データベースがバックアップからリストアされ、リモートデータベースのトランザクションログがリモートのオフセットと同じポイントでトランケートされた場合に役立ちます。この場合、リモートデータベースからアップロードされたすべてのデータは、統合オフセットのポイントからリモートオフセットのポイントまで失われます。

参照

- 「-r dbmlsync オプション」 127 ページ

例

RemoteProgressGreater=on

RemoteProgressLess 同期プロファイルオプション

リモートオフセットが統合オフセットよりも小さい場合に (リモートデータベースがバックアップからリストアされたときなど) リモートオフセットが使用されるように指定します。これは、-rb オプションと同じです。

構文

rb={ON|OFF}

RemoteProgressLess={ON|OFF}

備考

リモートデータベースがバックアップからリストアされると、デフォルトの動作がデータ損失の原因となることがあります。このオプションを使用すると、リモートデータベースに記録されたオフセットが統合データベースから取得したオフセットよりも小さい場合、リモートデータベースに記録されているオフセットからアップロードが継続されます。このオプションを使用し、リモートデータベースのオフセットが統合データベースからのオフセットよりも小さくない場合、エラーがレポートされ、同期がアボートされます。

このオプションでは、すでにアップロードされたデータがアップロードされることがあります。これにより統合データベースで競合が起こる可能性があり、これは適切な競合解決スクリプトを使用して処理する必要があります。

参照

- 「-r dbmlsync オプション」 127 ページ

例

RemoteProgressLess=on

Subscription 同期プロファイルオプション

同期させるサブスクリプションを指定します。

構文

s=*pubname*, ...

Subscription=*subname*, ...

備考

サブスクリプションは同期プロファイル内で 1 回しか指定できませんが、コマンドラインオプションは複数回指定できます。

参照

- 「-s dbmlsync オプション」 128 ページ

例

Subscription=mySubscription

TransactionalUpload 同期プロファイルオプション

ブール式リモートデータベースの各トランザクションを、1 つの同期内で独立したトランザクションとしてアップロードするように指定します。

構文

tu={ON|OFF}

TransactionalUpload={ON|OFF}

備考

TransactionalUpload 同期プロファイルオプションを使用するときは、トランザクションアップロードを作成します。こうすると、dbmlsync はリモートデータベースの各トランザクションを別個のトランザクションとしてアップロードします。Mobile Link サーバーは各トランザクションを受信したときに別々に適用およびコミットします。

参照

- 「-tu dbmlsync オプション」 130 ページ

例

TransactionalUpload=on

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

UpdateGenNum 同期プロファイルオプション

ダウンロードファイルを作成するとき、このオプションはまだ同期していないリモートデータベースで使用できるファイルを作成します。このオプションを使用しない場合は、同期を行ってからダウンロードファイルを適用する必要があります。

構文

bg={ON|OFF}

UpdateGenNum={ON|OFF}

備考

このオプションを使用すると、ダウンロードファイルによってリモートデータベースの世代番号が更新されます。

このオプションで構築したダウンロードファイルは、スナップショットダウンロードです。新しいリモートデータベースの最終ダウンロードタイムスタンプは、デフォルトでは 1900 年 1 月 1 日になっており、これはダウンロードファイル内の最終ダウンロードタイムスタンプより前となるため、タイムスタンプベースのダウンロードは同期していないリモートデータベースと連携しません。タイムスタンプベースでファイルベースのダウンロードが動作するには、ダウンロードファイル内の最終ダウンロードタイムスタンプがリモートと同じか、それより前である必要があります。

このオプションは、世代番号による機能を回避するため、そのような機能に依存するシステムの場合は、すでに同期されたリモートデータベースに UpdateGenNum ダウンロードファイルを適用しないでください。

参照

- 「-bg dbmlsync オプション」 109 ページ
- 「Mobile Link の世代番号」『Mobile Link サーバー管理』
- 「新しいリモートの同期」『Mobile Link サーバー管理』

例

```
UpdateGenNum=on
```

UploadOnly 同期プロファイルオプション

同期がアップロードだけを含み、ダウンロードが発生しないように指定します。

構文

uo={ON|OFF}

UploadOnly={ON|OFF}

備考

アップロード専用の同期中に、dbmlsync は通常の完全な同期とまったく同じように、Mobile Link へのアップロードを準備し送信します。しかし、ダウンロードを返信する代わりに、Mobile Link はアップロードのコミットが成功したかどうかを示す確認だけを送信します。

アップロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link サーバー管理』を参照してください。

参照

- 「-uo dbmlsync オプション」 133 ページ
- 「UploadOnly (uo) 拡張オプション」 167 ページ

例

```
UploadOnly=on
```

UploadRowCnt 同期プロファイルオプション

整数。同期でアップロードされるロー数の推定値を指定します。

構文

urc=*rowcount*

UploadRowCnt=*rowcount*

備考

パフォーマンスを改善するために、同期でアップロードするロー数の推定値を指定できます。この設定は、多数のローをアップロードするときに特に便利です。推定値が大きいほど、アップロードが高速になりますが、多くのメモリを消費します。

指定した推定値に関係なく、同期は正しく続行されます。

参照

- 「-urc dbmlsync オプション」 134 ページ

例

```
UploadRowCnt=100
```

Verbosity 同期プロファイルオプション

dbmlsync の冗長性を制御します。

構文

v={BASIC|HIGH|CONNECT_STR|ROW_CNT|OPTIONS|ML_PASSWORD|ROW_DATA|HOOK}, ...

Verbosity={BASIC|HIGH|CONNECT_STR|ROW_CNT|OPTIONS|ML_PASSWORD|ROW_DATA|HOOK}, ...

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

指定可能な値

値は、次のオプションの 1 つ以上を含むカンマ区切りのリストにします。各オプションは、異なるタイプの冗長性を有効にします。

- **BASIC** 制限された冗長性を生成する
- **HIGH** 可能な最高の冗長レベルを生成する
- **CONNECT_STR** ログ内の接続文字列を公開する
- **ROW_CNT** アップロードとダウンロードされたロー数のログを取る
- **OPTIONS** 同期の指定に使用するオプションのログを取る
- **ML_PASSWORD** ログ内の Mobile Link パスワードを公開する。
- **ROW_DATA** アップロードとダウンロードされたローのログを取る
- **HOOK** フックスクリプトに関連するメッセージのログを取る

備考

verbosity 拡張オプションと verbosity 同期プロファイルをオプションを指定して、競合が発生した場合は、verbosity 同期プロファイルオプションが verbosity 拡張オプションよりも優先されます。

-v オプションと verbosity 拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログオプションが追加され、指定したすべてのオプションが使用されます。-v コマンドラインオプションを使用すると、冗長オプションがすぐに有効になります。拡張オプションを使用する場合、最初の同期が始まるまで、冗長オプションは有効にならないため、起動情報のログは取られません。最初の同期の後、オプションの指定内容に関わらず、動作は同じになるはずです。

参照

- 「-v dbmlsync オプション」 135 ページ

例

```
Verbosity=OPTIONS, ML_PASSWORD
```

SQL Anywhere クライアントのイベントフック

SQL Anywhere 同期クライアント dbmlsync には、一連のイベントフックがオプションで用意されています。これを使用して、同期処理をカスタマイズできます。フックを実装した場合は、同期処理における特定の時点で呼び出されます。

イベントフックを実装するには、特定の名前の SQL ストアドプロシージャーを作成します。ほとんどのイベントフックストアドプロシージャーは、同期自体と同じ接続で実行されます。

イベントフックを使用して同期イベントのログを取り、処理することができます。たとえば、論理イベントに基づいた同期のスジュール、接続障害のリトライ、またはエラーや参照整合性違反の処理などが可能です。

また、イベントフックを使用して、パブリケーションで簡単に定義できないデータのサブセットを同期することもできます。たとえば、2 つのイベントフックプロシージャーを記述して、テンポラリテーブル内のデータを同期することができます。この場合、一方のイベントフックプロシージャーでは、同期の前にテンポラリテーブルから永久テーブルにデータをコピーし、他方では同期後にデータを逆にコピーします。

警告
同期処理の整合性は、組み込みトランザクションの順序に依存します。イベントフックプロシージャー内では、暗黙的または明示的なコミットもロールバックも実行しないでください。

フック内の接続設定を変更する場合は、設定を以前の値に復元してから、フックを終了してください。設定を復元しないと、予期しない結果になる場合があります。

dbmlsync インターフェイス

クライアントイベントフックは、Dbmlsync コマンドラインユーティリティと一緒に使用できます。または、SQL Anywhere クライアントの同期に使用するいずれのプログラミングインターフェイスとも一緒に使用できます。これには、Dbmlsync API と Dbmlsync 用の DBTools インターフェイスが含まれます。

「dbmlsync の同期のカスタマイズ」97 ページを参照してください。

同期イベントフックの順序

次の疑似コードは、使用可能なイベントと、同期処理中に各イベントが呼び出されるポイントを示します。たとえば、sp_hook_dbmlsync_abort は最初に呼び出されるイベントフックです。

```
sp_hook_dbmlsync_abort //not called when Dbmlsync API or the SQL SYNCHRONIZE STATEMENT is
used
sp_hook_dbmlsync_set_extended_options
loop until return codes direct otherwise (
   sp_hook_dbmlsync_abort
   sp_hook_dbmlsync_delay
)
sp_hook_dbmlsync_abort
// start synchronization
sp_hook_dbmlsync_begin
// upload events
for each upload segment
// a normal synchronization has one upload segment
// a transactional upload has one segment per transaction
// an incremental upload has one segment per upload piece
 sp_hook_dbmlsync_logscan_begin  //not called for scripted upload
 sp_hook_dbmlsync_logscan_end  //not called for scripted upload
 sp_hook_dbmlsync_set_ml_connect_info //only called during first upload
 sp_hook_dbmlsync_upload_begin
 sp_hook_dbmlsync_set_upload_end_progress  //only called for scripted upload
 sp_hook_dbmlsync_upload_end
next upload segment
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
// download events
sp_hook_dbmlsync_validate_download_file (only called
   when -ba option is used)
sp_hook_dbmlsync_download_begin
for each table
   sp_hook_dbmlsync_download_table_begin
   sp_hook_dbmlsync_download_table_end
next table
sp_hook_dbmlsync_download_end

sp_hook_dbmlsync_schema_upgrade
// end synchronization
sp_hook_dbmlsync_end
sp_hook_dbmlsync_process_exit_code
sp_hook_dbmlsync_log_rescan
```

イベントフック

各フックには、プロシージャーの実装時に使用できるパラメーター値が用意されています。パラメーター値の中には、新しい値を返すように変更できるものがあります。それ以外のものは、読み込み専用です。これらのパラメーターは、ストアドプロシージャーの引数ではありません。いずれのイベントフックストアドプロシージャーにも、引数は渡されません。代わりに、#hook_dict テーブル内のローを読み込んだり修正したりすることで、引数がやりとりされます。

たとえば、sp_hook_dbmlsync_begin プロシージャーには Mobile Link ユーザーのパラメーターが 1 つあります。これは、同期している Mobile Link ユーザーです。この値は、#hook_dict テーブルから取り出すことができます。

順序は Mobile Link サーバーでのイベントの順序と類似していますが、統合データベースとリモートデータベースに追加する論理の種類には、重複はほとんどありません。したがって、2 つのインターフェイスは別のものとなります。

*_begin フックが正常に実行されると、*_begin フックの後にどのようなエラーが発生しても、対応する *_end フックが呼び出されます。*_begin フックは定義されていないが、*_end フックが定義されている場合、通常 *_begin フックが呼び出される時点より前にエラーが発生しないかぎり、*_end フックが呼び出されます。

フックがデータベース内のデータを変更すると、sp_hook_dbmlsync_logscan_begin での変更を含むこれまでのすべての変更が、現在の同期セッションで同期されます。それ以降の変更は、次のセッションで同期されます。

イベントフックプロシージャー

この項では、イベントフックプロシージャーの設計と使用におけるいくつかの注意事項について説明します。

注意

- イベントフックプロシージャーでは、COMMIT 操作も ROLLBACK 操作も実行しないでください。プロシージャーは同期と同じ接続で実行されるので、COMMIT または ROLLBACK を実行すると同期が妨害されます。

●フック内の接続設定を変更する場合は、設定を以前の値に復元してから、フックを終了してください。設定を復元しないと、予期しない結果になる場合があります。

●dbmlsync は、ストアドプロシージャーを、所有者で識別せずに呼び出します。したがって、ストアドプロシージャーは、dbmlsync 接続で使用されるユーザー名、またはユーザーがメンバーであるグループのどちらかで所有されている必要があります。

●フックプロシージャーは DBA 権限を持つユーザーが所有してください。

#hook_dict テーブル

フックが呼び出される直前に、dbmlsync は次の CREATE 文を使用してリモートデータベースに #hook_dict テーブルを作成します。テーブル名の前の # は、そのテーブルがテンポラリであることを意味します。

```
CREATE TABLE #hook_dict(
name VARCHAR(128) NOT NULL UNIQUE,
value VARCHAR(10240) NOT NULL)
```

dbmlsync ユーティリティは #hook_dict テーブルを使用してフック関数に値を渡し、フック関数は #hook_dict テーブルを使用して dbmlsync に値を戻します。

各フックは、パラメーター値を受け取ります。パラメーター値の中には、新しい値を返すように変更できるものがあります。それ以外のものは、読み込み専用です。このテーブルの各ローには、1 つのパラメーターの値があります。

たとえば、2 つのサブスクリプションが次のように定義されているとします。

```
CREATE SYNCHRONIZATION SUBSCRIPTION sub1
TO pub1
FOR  MyUser;
SCRIPT VERSION 'v1'

CREATE SYNCHRONIZATION SUBSCRIPTION sub2
TO pub2
FOR  MyUser;
SCRIPT VERSION 'v1'
```

sp_hook_dbmlsync_begin フックが、次の dbmlsync コマンドラインに対して呼び出されるとします。

```
dbmlsync -c 'DSN=MyDsn' -s sub1,sub2
```

#hook_dict テーブルには、次のローが含まれます。

#hook_dict ロー	値
subscription_0	sub1
subscription_1	sub2

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

#hook_dict ロー	値
publication_0	pub1
publication_1	pub2
MobiLink user	MyUser
Script version	v1

注意
publication_*n* ローは廃止される予定であり、今後のリリースで削除される可能性があります。

フックを使用して、#hook_dict テーブルから値を取り出したり、その動作をカスタマイズしたりできます。たとえば、Mobile Link ユーザーを取り出すには、次のように SELECT 文を使用します。

```
SELECT value
FROM #hook_dict
WHERE name = 'MobiLink user'
```

In/out パラメーターは、dbmlsync の動作をフックで修正することによって更新できます。たとえば、次のような文を使用してテーブルの abort synchronization ローを更新することで、同期のアボートを sp_hook_dbmlsync_abort フックから dbmlsync に指示することができます。

```
UPDATE #hook_dict
SET value='true'
WHERE name='abort synchronization'
```

各フックの記述には、#hook_dict テーブルのローがリストされます。

例

次のサンプル sp_hook_dbmlsync_delay プロシージャーは、#hook_dict テーブルでの in/out パラメーターの使用を示します。このプロシージャーでは、Mobile Link システムのスケジュールされたダウン時間 (18:00 ～ 19:00) 以外にのみ同期を行います。

```
CREATE PROCEDURE sp_hook_dbmlsync_delay()
BEGIN
  DECLARE delay_val integer;
 SET delay_val=DATEDIFF(
  second, CURRENT TIME, '19:00');
 IF (delay_val>0 AND
    delay_val<3600)
 THEN
 UPDATE #hook_dict SET value=delay_val
  WHERE name='delay duration';
 END IF;
END
```

次のプロシージャーは、同期の開始時にリモートデータベース内で実行されます。現在の Mobile Link ユーザー名 (sp_hook_dbmlsync_begin イベントに使用可能なパラメーターの 1 つ) を取り出して、SQL Anywhere のメッセージウィンドウに出力します。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin()
BEGIN
  DECLARE MLuser VARCHAR(150);
  SELECT '>>>MLuser = ' || value INTO MLuser
    FROM #hook_dict
    WHERE name ='MobiLink user';
  MESSAGE MLuser TYPE INFO TO CONSOLE;
END
```

イベントフックプロシージャー用の接続

各イベントフックプロシージャーは、同期自体と同じ接続で実行されます。ただし、次のプロシージャーは例外です。

- sp_hook_dbmlsync_all_error
- sp_hook_dbmlsync_communication_error
- sp_hook_dbmlsync_download_log_ri_violation
- sp_hook_dbmlsync_misc_error
- sp_hook_dbmlsync_sql_error

これらのプロシージャーは、同期が失敗する前に呼び出されます。失敗すると、同期アクションがロールバックされます。別の接続で実行すると、これらのプロシージャーを使用して失敗情報のログを取ることができ、このとき、ログアクションは同期アクションとともにはロールバックされません。

イベントフックプロシージャー内でのエラーと警告の処理

イベントフックストアドプロシージャーを作成すると、同期エラー、Mobile Link の接続障害、参照整合性違反を処理することができます。この項では、エラーや警告の処理に使用するイベントフックプロシージャーについて説明します。実装された各プロシージャーは、指定したタイプのエラーが発生するたびに自動的に実行されます。

参照整合性違反の処理

ダウンロード内のローがリモートデータベース上の外部キー関係に違反すると、参照整合性違反が発生します。次のイベントフックを使用して参照整合性違反のログを取り、処理します。

- 「sp_hook_dbmlsync_download_log_ri_violation」
- 「sp_hook_dbmlsync_download_ri_violation」

Mobile Link 接続障害の処理

sp_hook_dbmlsync_ml_connect_failed イベントフックを使用すると、Mobile Link サーバーへの接続が失敗した場合に、異なる通信タイプまたはアドレスを使用してリトライすることができます。リトライしても接続が失敗した場合、dbmlsync は sp_hook_dbmlsync_communication_error と sp_hook_dbmlsync_all_error フックを呼び出します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

「sp_hook_dbmlsync_ml_connect_failed」232 ページを参照してください。

dbmlsync エラーの処理

dbmlsync エラーメッセージが生成されるたびに、次のフックが呼び出されます。

- まず、エラーのタイプに応じて、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error のいずれかのフックが呼び出されます。これらのフックには、エラーのタイプに固有の情報が含まれています。たとえば、SQL エラーでは sqlcode と sqlstate が返されます。

- 次に、sp_hook_dbmlsync_all_error が呼び出されます。このフックには、発生したすべてのエラーのログを取る場合に役立ちます。

次の項を参照してください。

- 「sp_hook_dbmlsync_communication_error」
- 「sp_hook_dbmlsync_sql_error」
- 「sp_hook_dbmlsync_misc_error」
- 「sp_hook_dbmlsync_all_error」

エラーを受けて同期を再度開始するには、sp_hook_dbmlsync_end 内の user state パラメーターを使用します。

「sp_hook_dbmlsync_end」221 ページを参照してください。

エラーの無視

イベントフックプロシージャー内で未処理のエラーが発生した場合、デフォルトでは同期は停止します。dbmlsync ユーティリティで -eh オプションを指定すると、このようなエラーを無視するように指示できます。

「IgnoreHookErrors (eh) 拡張オプション」152 ページを参照してください。

sp_hook_dbmlsync_abort

このストアドプロシージャーは、同期処理をキャンセルする場合に使用します。

#hook_dict テーブルのロー

名前	値	説明
abort synchronization (in|out)	**true** | **false**	#hook_dict テーブルの abort synchronization ローを **true** に設定すると、同期はイベント終了後すぐに終了します。

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
exit code (in|out)	数値	abort synchronization を TRUE に設定すると、この値を使用して、アボートされた同期の終了コードを設定できます。0 は同期が成功したことを示します。他の数は同期が失敗したことを示します。
script version (in|out)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。同期されるサブスクリプションごとに 1 つの subscription_n エントリがあります。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、そのプロシージャーは dbmlsync の起動時に呼び出され、sp_hook_dbmlsync_delay フックにより各同期が遅延した後に再度呼び出されます。

dbmlsync をコマンドラインから実行する場合に abort synchronization を true に設定すると、残りのすべての同期 (スケジュールされた同期も含む) がキャンセルされます。Dbmlsync API または SQL SYNCHRONIZE 文を使用する場合に abort synchronization を true に設定すると、現在の同期のみが中止されます。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「同期イベントフックの順序」 196 ページ
- 「sp_hook_dbmlsync_process_exit_code」 235 ページ

例

次のプロシージャーは、毎日 19:00 ～ 20:00 にスケジュールされた保守作業時間中に、同期が行われないようにします。

```
CREATE PROCEDURE sp_hook_dbmlsync_abort()
BEGIN
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
  DECLARE down_time_start TIME;
  DECLARE is_down_time VARCHAR(128);
  SET down_time_start='19:00';
  IF datediff( hour,down_time_start,now(*) ) < 1
  THEN
    set is_down_time='true';
  ELSE
    SET is_down_time='false';
  END IF;
  UPDATE #hook_dict
  SET value = is_down_time
  WHERE name = 'abort synchronization'
END;
```

次の 2 つの理由のいずれかにより、同期をアボート可能なアボートフックがあるとします。1 つ目の理由は、同期の正常終了を示すのに、dbmlsync に終了コード 0 を含ませるためです。また、2 つ目の理由は、エラー状態を示すのに、dbmlsync に 0 以外の終了コードを含ませるためです。sp_hook_dbmlsync_abort フックを次のように定義すれば、上記のことが可能です。

```
BEGIN
  IF [condition that defines the normal abort case] THEN
    UPDATE #hook_dict SET value =  '0'
    WHERE name = 'exit code';
    UPDATE #hook_dict SET value =  'TRUE'
    WHERE name = 'abort synchronization';
  ELSEIF [condition that defines the error abort case] THEN
    UPDATE #hook_dict SET value =  '1'
    WHERE name = 'exit code';
    UPDATE #hook_dict SET value =  'TRUE'
    WHERE name = 'abort synchronization';
  END IF;
END;
```

sp_hook_dbmlsync_all_error

このストアドプロシージャーを使用して、すべてのタイプの dbmlsync エラーメッセージを処理します。たとえば、sp_hook_dbmlsync_all_error フックを実装すると、特定のエラーが発生した場合に、ログを取ったり特定のアクションを実行したりすることができます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー

名前	値	説明
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
error message (in)	エラーメッセージテキスト	これは、dbmlsync ログに表示されるテキストと同じです。
error id (in)	整数	メッセージをユニークに識別する ID。このローを使用すると、エラーメッセージテキストが変更されたときに、エラーメッセージを識別できます。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

dbmlsync エラーメッセージが生成されるたびに、次のフックが呼び出されます。

- まず、エラーのタイプに応じて、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error のいずれかのフックが呼び出されます。これらのフックには、エラーのタイプに固有の情報が含まれています。たとえば、SQL エラーでは sqlcode と sqlstate が返されます。

- 次に、sp_hook_dbmlsync_all_error が呼び出されます。このフックには、発生したすべてのエラーのログを取る場合に役立ちます。

同期を開始する前の起動中にエラーが発生した場合、#hook_dict 内の Mobile Link ユーザーとスクリプトバージョンのエントリは空の文字列に設定され、#hook_dict テーブルで設定される publication_*n* ローや subscription_*n* ローはありません。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

error hook user state ローは、エラーの内容を sp_hook_dbmlsync_end フックに渡す場合に役立つメカニズムを提供します。フックでは、この情報を使用して同期をリトライするかどうかを決定できます。

このプロシージャーは別個の接続で実行されるため、同期接続でロールバックが実行されても、このプロシージャーで実行する操作が失われることはありません。dbmlsync が別個の接続を確立できないと、プロシージャーは呼び出されません。

このフックは別の接続で実行されるため、フックプロシージャーで同期されているテーブルにアクセスする場合は注意してください。これらのテーブルは dbmlsync によってロックされている可能性があるためです。これらのロックが原因で、フック内の操作が失敗したり、無期限に待機したりすることがあります。

このプロシージャーのアクションは、フックが完了した直後にコミットされます。

参照

- 「イベントフックプロシージャー内でのエラーと警告の処理」200 ページ
- 「sp_hook_dbmlsync_communication_error」207 ページ
- 「sp_hook_dbmlsync_misc_error」229 ページ
- 「sp_hook_dbmlsync_sql_error」244 ページ

例

次のテーブルを使用して、リモートデータベース内のエラーのログを取ります。

```
CREATE TABLE error_log
(
 pk INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,
 err_id INTEGER,
 err_msg VARCHAR(10240),
);
```

次の例では、sp_hook_dbmlsync_all_error を設定して、エラーログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_all_error()
BEGIN
 DECLARE msg VARCHAR(10240);
 DECLARE id INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error id
 SELECT value INTO id
  FROM #hook_dict
  WHERE name = 'error id';

 // log the error information
 INSERT INTO error_log(err_msg, err_id)
  VALUES (msg, id);
END;
```

エラーの可能性がある ID 値を表示するには、dbmlsync をテスト実行します。たとえば、dbmlsync が「Mobile Link サーバーに接続できません。」というエラーを返すと、sp_hook_dbmlsync_all_error が次のローを error_log に挿入します。

```
  1,14173,
  'Unable to connect to MobiLink server'
```

これで、「Mobile Link サーバーに接続できません。」というエラーとエラー ID 14173 を関連付けることができます。

次の例では、エラー 14173 が発生したときに必ず同期をリトライするようにフックを設定します。

```
CREATE PROCEDURE sp_hook_dbmlsync_all_error()
BEGIN
 IF EXISTS( SELECT value FROM #hook_dict
    WHERE name = 'error id' AND value = '14173' )
 THEN
    UPDATE #hook_dict SET value = '1'
       WHERE name = 'error hook user state';
   END IF;
END;

CREATE PROCEDURE sp_hook_dbmlsync_end()
BEGIN
 IF EXISTS( SELECT value FROM #hook_dict
    WHERE name='error hook user state' AND value='1')
 THEN
    UPDATE #hook_dict SET value = 'sync'
       WHERE name='restart';
   END IF;
END;
```

「sp_hook_dbmlsync_end」221 ページを参照してください。

sp_hook_dbmlsync_begin

このストアドプロシージャーを使用して、同期処理の開始時にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、同期処理の開始時に呼び出されます。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「同期イベントフックの順序」 196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT AUTOINCREMENT ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、テーブル内の各同期の始めにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_begin', subs_list );
END
```

sp_hook_dbmlsync_communication_error

このストアドプロシージャーは、通信エラーを処理する場合に使用します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
error message (in)	エラーメッセージテキスト	これは、dbmlsync ログに表示されるテキストと同じです。
error id (in)	数値	メッセージをユニークに識別する ID。このローを使用すると、エラーメッセージテキストが変更されたときに、エラーメッセージを識別できます。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。
stream error code (in)	整数	ストリームによってレポートされるエラー。
system error code (in)	整数	システム固有のエラーコード。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

同期を開始する前の起動中にエラーが発生した場合、#hook_dict 内の Mobile Link ユーザーとスクリプトバージョンのエントリは空の文字列に設定され、#hook_dict テーブルで設定される publication_*n* ローや subscription_*n* ローはありません。

dbmlsync と Mobile Link サーバー間で通信エラーが発生した場合、このフックを使用すると、ストリーム固有のエラー情報にアクセスすることができます。

stream error code パラメーターは、通信エラーのタイプを示す整数です。

error hook user state ローは、エラーの内容を sp_hook_dbmlsync_end フックに渡す場合に役立つメカニズムを提供します。フックでは、この情報を使用して同期をリトライするかどうかを決定できます。

このプロシージャーは別個の接続で実行されるため、同期接続でロールバックが実行されても、このプロシージャーで実行する操作が失われることはありません。dbmlsync が別個の接続を確立できないと、プロシージャーは呼び出されません。

このフックは別の接続で実行されるため、フックプロシージャーで同期されているテーブルにアクセスする場合は注意してください。これらのテーブルは dbmlsync によってロックされている可能性があるためです。これらのロックが原因で、フック内の操作が失敗したり、無期限に待機したりすることがあります。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「Mobile Link 通信エラーメッセージ」『エラーメッセージ』
- 「イベントフックプロシージャー内でのエラーと警告の処理」200 ページ
- 「sp_hook_dbmlsync_all_error」203 ページ
- 「sp_hook_dbmlsync_misc_error」229 ページ
- 「sp_hook_dbmlsync_sql_error」244 ページ

例

次のテーブルを使用して、リモートデータベース内の通信エラーのログを取ります。

```
CREATE TABLE communication_error_log
(
  error_msg VARCHAR(10240),
  error_code VARCHAR(128)
);
```

次の例では、sp_hook_dbmlsync_communication_error を設定して、通信エラーのログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_communication_error()
BEGIN
 DECLARE msg VARCHAR(255);
 DECLARE code INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';
```

```
// get the error code
SELECT value INTO code
 FROM #hook_dict
 WHERE name = 'stream error code';

// log the error information
INSERT INTO communication_error_log(error_code,error_msg)
 VALUES (code,msg);
END
```

sp_hook_dbmlsync_delay

このストアドプロシージャーを使用して、同期が行われるタイミングを制御します。

#hook_dict テーブルのロー

名前	値	説明
delay duration (in|out)	秒数	プロシージャーが delay duration の値を 0 に設定すると、dbmlsync の同期がすぐに進行します。delay duration が 0 以外の値の場合は、遅延フックが再び呼び出されるまでの秒数を示します。
maximum accumulated delay (in|out)	秒数	最大累積遅延は、各同期前の最大秒数を指定します。dbmlsync は、最後に実行された同期以降の遅延フックへのすべての呼び出しによって生じた合計遅延時間を追跡します。dbmlsync が実行を開始してから同期が行われないと、dbmlsync の起動時間から合計遅延時間が計算されます。合計遅延時間が Maximum Accumulated delay 値を上回ると、遅延フックを呼び出さずに同期が開始されます。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、同期処理の開始時の、**sp_hook_dbmlsync_begin** の前に呼び出されます。

Dbmlsync API または SQL SYNCHRONIZE 文を使用して同期が開始されている場合、このフックは呼び出されません。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「同期イベントフックの順序」196 ページ
- 「イベントフックを使用した同期の開始」97 ページ
- 「sp_hook_dbmlsync_download_end」213 ページ

例

次のテーブルを使用して、リモートデータベース上の注文のログを取ります。

```
CREATE TABLE OrdersTable(
  "id" INTEGER PRIMARY KEY DEFAULT AUTOINCREMENT,
  "priority" VARCHAR(128)
);
```

次に、最大累積遅延 (1 時間) の間、同期を遅らせる例を示します。10 秒ごとにフックが呼び出され、OrdersTable 内に優先度の高いローがないかをチェックします。優先度の高いローが存在する場合、遅延期間は 0 に設定して同期処理を開始します。

```
CREATE PROCEDURE sp_hook_dbmlsync_delay()
BEGIN
     -- Set the maximum delay between synchronizations
    -- or before the first synchronization starts to 1 hour
    UPDATE #hook_dict SET value = '3600'  // 3600 seconds
     WHERE name = 'maximum accumulated delay';

    -- check if a high priority order exists in OrdersTable
    IF EXISTS (SELECT * FROM OrdersTable where priority='high') THEN
     -- start the synchronization to process the high priority row
     UPDATE #hook_dict
      SET value = '0'
      WHERE name='delay duration';
    ELSE
     -- set the delay duration to call this procedure again
     -- following a 10 second delay
     UPDATE #hook_dict
      SET value = '10'
      WHERE name='delay duration';
    END IF;
END;
```

sp_hook_dbmlsync_end フックでは、優先度の高いローを処理済みとしてマークすることができます。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
    BEGIN
        IF EXISTS( SELECT value FROM #hook_dict
     WHERE name = 'Upload status'
     AND value = 'committed' ) THEN
     UPDATE OrderTable SET priority = 'high-processed'
    WHERE priority = 'high';
        END IF;
    END;
```

この例では、同期中にテーブルが確実にロックされるようにするため、LockTables 拡張オプションを使用することを前提としています。テーブルがロックされていないと、アップロードの構築後で、sp_hook_dbmlsync_end フックの実行前に優先度の高いローが挿入される可能性があります。この操作が行われると、ローがアップロードされていなくても優先度が "high-processed" に変更されます。

「sp_hook_dbmlsync_end」221 ページを参照してください。

sp_hook_dbmlsync_download_begin

このストアドプロシージャーを使用して、同期処理のダウンロード処理開始時にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

この名前のプロシージャーが存在する場合、同期処理のダウンロード処理開始時に呼び出されます。

ダウンロードがコミットまたはロールバックされると、このプロシージャーのアクションがコミットまたはロールバックされます。

参照

- 「同期イベントフックの順序」 196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT AUTOINCREMENT ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにダウンロードの始めにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_begin ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_download_begin', subs_list );
END
```

sp_hook_dbmlsync_download_end

このストアドプロシージャーを使用して、同期処理のダウンロード処理終了時にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン
subscription_n (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、同期処理のダウンロード処理終了時に呼び出されます。

ダウンロードがコミットまたはロールバックされると、このプロシージャーのアクションがコミットまたはロールバックされます。

参照

- 「同期イベントフックの順序」196 ページ
- 「イベントフックを使用した同期の開始」97 ページ
- 「sp_hook_dbmlsync_delay」210 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにダウンロードの終わりにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_end ()
BEGIN
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
    DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
    SELECT LIST(value) INTO subs_list
     FROM #hook_dict
     WHERE name LIKE 'subscription_%';

-- log the event
    INSERT INTO SyncLog(event_time, event_name, subs)
    VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_download_end', subs_list );
END
```

sp_hook_dbmlsync_download_log_ri_violation

ダウンロードプロセスの参照整合性違反のログを取ります。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
foreign key table (in)	テーブル名	フック呼び出し対象の外部キーカラムを含むテーブル。
primary key table (in)	テーブル名	フック呼び出し対象の外部キーが参照するテーブル。
role name (in)	ロール名	フック呼び出し対象の外部キーのロール名。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

ダウンロード内のローがリモートデータベース上の外部キー関係に違反すると、ダウンロード参照整合性違反が発生します。このフックを使用すると、発生した参照整合性違反のログを取り、後でその原因を調べることができます。

ダウンロードが完了すると、コミットされる前に、dbmlsync は参照整合性違反があるかどうかをチェックします。参照整合性違反が見つかると、参照整合性違反を含む外部キーを識別して、sp_hook_dbmlsync_download_log_ri_violation を呼び出します (実装されている場合)。次に、sp_hook_dbmlsync_download_ri_violation を呼び出します (実装されている場合)。それでも矛盾がある場合、dbmlsync は外部キー制約に違反するローを削除します。参照整合性違反を含む残りの外部キーに対して、このプロセスが繰り返されます。

このフックは、現在同期中のテーブルに関する参照整合性違反がある場合にのみ呼び出されます。同期中ではないテーブルに関する参照整合性違反がある場合、このフックは呼び出されず、同期が失敗します。

このフックは、dbmlsync がダウンロードに使用する接続とは別の接続で呼び出されます。このフックが使用する接続の独立性レベルは 0 のため、ダウンロードから適用されていてまだコミットされていないローを判別できます。フックのアクションは完了直後にコミットされるので、ダウンロードがコミットされるかロールバックされるかに関係なく、このフックによる変更は保存されます。

このフックは別の接続で実行されるため、フックプロシージャーで同期されているテーブルにアクセスする場合は注意してください。これらのテーブルは dbmlsync によってロックされている可能性があるためです。これらのロックが原因で、フック内の操作が失敗したり、無期限に待機したりすることがあります。

参照整合性違反の問題を解決するために、このフックを使用しないでください。このフックはロギングにのみ使用してください。参照整合性違反を解決するには、sp_hook_dbmlsync_download_ri_violation を使用します。

参照

- 「sp_hook_dbmlsync_download_ri_violation」 217 ページ
- 「同期イベントフックの順序」 196 ページ

例

次のテーブルを使用して、参照整合性違反のログを取ります。

```
CREATE TABLE DBA.LogRIViolationTable
(
   entry_time  TIMESTAMP,
   pk_table  VARCHAR( 255 ),
   fk_table  VARCHAR( 255 ),
   role_name  VARCHAR( 255 )
);
```

次に、リモートデータベース上で参照整合性違反が検出されたときに、外部キーテーブル名、プライマリキーテーブル名、役割名のログを取る例を示します。この情報は、リモートデータベースの LogRIViolationTable に格納されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_log_ri_violation()
BEGIN
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
 INSERT INTO DBA.LogRIViolationTable VALUES(
 CURRENT_TIMESTAMP,
 (SELECT value FROM #hook_dict WHERE name = 'Primary key table'),
 (SELECT value FROM #hook_dict WHERE name = 'Foreign key table'),
 (SELECT value FROM #hook_dict WHERE name = 'Role name' ) );
END;
```

sp_hook_dbmlsync_download_ri_violation

ダウンロードプロセスの参照整合性違反を解決できます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
foreign key table (in)	テーブル名	フック呼び出し対象の外部キーカラムを含むテーブル。
primary key table (in)	テーブル名	フック呼び出し対象の外部キーが参照するテーブル。
role name (in)	ロール名	フック呼び出し対象の外部キーのロール名。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

ダウンロード内のローがリモートデータベース上の外部キー関係に違反すると、ダウンロード参照整合性違反が発生します。このフックを使用すると、dbmlsync が競合の原因であるローを削除する前に、参照整合性違反を解決できます。

ダウンロードが完了すると、コミットされる前に、dbmlsync は参照整合性違反があるかどうかをチェックします。参照整合性違反が見つかると、参照整合性違反を含む外部キーを識別して、sp_hook_dbmlsync_download_log_ri_violation を呼び出します (実装されている場合)。次に、sp_hook_dbmlsync_download_ri_violation を呼び出します (実装されている場合)。それでも競合が解決されない場合は、dbmlsync がそのローを削除します。参照整合性違反を含む残りの外部キーに対して、このプロセスが繰り返されます。

このフックは、現在同期中のテーブルに関する参照整合性違反がある場合にのみ呼び出されます。同期中ではないテーブルに関する参照整合性違反がある場合、このフックは呼び出されず、同期が失敗します。

このフックは、dbmlsync がダウンロードに使用する接続と同じ接続で呼び出されます。データベースでデータの不整合が発生する可能性があるため、このフックに明示的または暗黙的なコミットが含まれないようにしてください。ダウンロードがコミットまたはロールバックされると、このフックのアクションがコミットまたはロールバックされます。

他のフックアクションとは異なり、このフック中に実行される操作は次の同期中にアップロードされません。

参照

- 「sp_hook_dbmlsync_download_log_ri_violation」215 ページ

例

この例は、次に示す Department テーブルと Employee テーブルを使用します。

```
CREATE TABLE Department(
 "department_id"  INTEGER primary key
);

CREATE TABLE Employee(
  "employee_id"      INTEGER PRIMARY KEY,
  "department_id" INTEGER,
  FOREIGN KEY EMPLOYEE_FK1 (department_id) REFERENCES Department
);
```

次の sp_hook_dbmlsync_download_ri_violation の定義は、Department テーブルと Employee テーブル間の参照整合性違反をクリーンアップします。この定義によって、外部キーの役割名が検証され、欠落している department_id の値が Department テーブルに挿入されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_ri_violation()
BEGIN

IF EXISTS (SELECT * FROM #hook_dict WHERE name = 'role name'
 AND value = 'EMPLOYEE_FK1') THEN

 -- update the Department table with missing department_id values
 INSERT INTO Department
  SELECT distinct department_id FROM Employee
  WHERE department_id NOT IN (SELECT department_id FROM Department)

END IF;
END;
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

sp_hook_dbmlsync_download_table_begin

このストアドプロシージャーを使用して、各テーブルがダウンロードされる直前にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
table name (in)	テーブル名	操作が適用される予定のテーブル。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、ダウンロードされた操作がテーブルに適用される直前にテーブルごとに呼び出されます。ダウンロードがコミットまたはロールバックされると、このプロシージャーのアクションがコミットまたはロールバックされます。

参照

- 「同期イベントフックの順序」196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとに各テーブルのダウンロードの始めにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_begin ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_download_table_begin, subs_list );
END
```

sp_hook_dbmlsync_download_table_end

このストアドプロシージャーを使用して、各テーブルがダウンロードされた直後にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
table name (in)	テーブル名	操作が直前に適用されたテーブル。
delete count (in)	ローの数	ダウンロードによってこのテーブルから削除されたローの数。
upsert count (in)	ローの数	ダウンロードによってこのテーブルで更新または挿入されたローの数。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

この名前のプロシージャーが存在する場合、ダウンロードでの操作がすべてテーブルに適用された直後に呼び出されます。

ダウンロードがコミットまたはロールバックされると、このプロシージャーのアクションがコミットまたはロールバックされます。

参照

- 「同期イベントフックの順序」 196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとに各テーブルのダウンロードの終わりにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_end ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_download_table_end, subs_list );
END
```

sp_hook_dbmlsync_end

このストアドプロシージャーを使用して、同期が完了する直前にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
restart (out)	**sync** | **download** | **none**	**sync** に設定すると、dbmlsync は完了した同期のリトライを行います。**true** も同じですが、廃止され、値 **sync** に置き換えられました。 **none** (デフォルト) に設定すると、dbmlsync はコマンドライン引数の指定に従って、停止するか、または再起動します。**false** も同じですが、廃止され、値 **none** に置き換えられました。 **download** に設定した場合、restartable download パラメーターが **true** であると、dbmlsync は失敗したダウンロードを再起動します。
exit code (in)	数値	直前に完了した同期の終了コード。ゼロ以外の値は、同期エラーを表します。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
upload status (in)	**not sent** | **committed** | **failed** | **unknown**	dbmlsync がアップロードの受信確認を行おうとしたときに、Mobile Link サーバーから返されるステータスを指定します。次のいずれかのステータスになります。 ● **not sent** - エラーが原因で、または要求された同期で不要だったので、アップグレードは Mobile Link サーバーに送信されませんでした。これは、ダウンロード専用の同期、再起動したダウンロード、ファイルベースのダウンロードで発生します。 ● **committed** - Mobile Link サーバーがアップロードを受信し、コミットしました。 ● **failed** - Mobile Link サーバーは、アップロードをコミットしませんでした。トランザクション単位のアップロードについては、すべてではないものの、一部のトランザクションが、サーバーによって正しくアップロードされ、受信確認されたときは、アップロードステータスは 'failed' になります。 ● **unknown** - Mobile Link サーバーは、アップロードを確認しませんでした。アップロードがコミットされたかどうかを確認する方法はありません。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン
restartable download (in)	**true**|**false**	**true** の場合、現在の同期のダウンロードが失敗しており、再起動できます。**false** の場合、ダウンロードが正常に行われたか、再起動できません。
restartable download size (in)	整数	restartable download パラメーターが **true** である場合、このパラメーターはダウンロードが失敗する前に受信したバイト数を示します。restartable download が **false** の場合、このパラメーターの値は無効です。

名前	値	説明
error hook user state (in)	整数	この値にはエラーについての情報が含まれ、フック sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、または sp_hook_dbmlsync_sql_error から送信できます
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、各同期の最後に呼び出されます。

再起動パラメーターを常に **sync** に設定するように sp_hook_dbmlsync_end フックが定義されており、ユーザーが dbmlsync のコマンドラインで -s sub1、-s sub2 などの形式で複数のサブスクリプションを指定している場合、dbmlsync は最初のサブスクリプションを繰り返し同期し、2 番目のサブスクリプションを同期しません。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「SQL Anywhere クライアントのイベントフック」 195 ページ
- 「同期イベントフックの順序」 196 ページ
- 「ダウンロードの失敗の再開」『Mobile Link サーバー管理』
- 「イベントフックプロシージャー内でのエラーと警告の処理」 200 ページ

例

次の例では、現在の同期のダウンロードが失敗して再起動が可能な場合、ダウンロードは手動で再起動されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_end()
BEGIN
  -- Restart the download if the download for the current sync
  --  failed and can be restarted
  IF EXISTS (SELECT * FROM #hook_dict
    WHERE name = 'restartable download' AND value='true')
      THEN
   UPDATE #hook_dict SET value ='download' WHERE name='restart';
  END IF;
END;
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

sp_hook_dbmlsync_log_rescan

このストアドプロシージャーを使用して、再スキャンがいつ必要かプログラム的に決定できます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
discarded storage (in)	数値	最後の同期の後で破棄されるメモリのバイト数。
rescan (in|out)	**true** | **false**	フックによって True と設定されている場合、dbmlsync は次の同期の前に完全な再スキャンを行います。エントリ時には、この値は False に設定されます。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

コマンドラインで複数の -n オプションまたは -s オプションを指定すると、メモリ破棄の原因となる断片化が dbmlsync で起きる可能性があります。破棄されたメモリは、データベーストランザクションログの再スキャンによってリカバリできます。このフックにより、dbmlsync を使用してデータベーストランザクションログの再スキャンを行いメモリを回復させるかどうかを決定できます。

再スキャンを強制する他の条件を満たさない場合、このフックは sp_hook_dbmlsync_process_exit_code フックの直後に呼び出されます。

参照

- 「HoverRescanThreshold (hrt) 拡張オプション」 151 ページ

例

次の例では、破棄された記憶領域が 1000 バイトを超える場合にログスキャンが行われます。

```
CREATE PROCEDURE sp_hook_dbmlsync_log_rescan ()
BEGIN
 IF EXISTS(SELECT * FROM #hook_dict
  WHERE name = 'Discarded storage' AND value>1000)
 THEN
  UPDATE #hook_dict SET value ='true' WHERE name='Rescan';
 END IF;
END;
```

sp_hook_dbmlsync_logscan_begin

このストアドプロシージャーを使用して、アップロード用にトランザクションログがスキャンされる直前にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
starting log offset_*n* (in)	数値	同期中の各サブスクリプションの進行値。進行値は、サブスクリプションのすべてのデータがアップロードされた時点までのトランザクションログ内のオフセットです。同期されるサブスクリプションごとに 1 つの値があります。*n* の番号は 0 から始まります。この値は、subscription_*n* と一致します。たとえば、log offset_0 は subscription_0 のオフセットです。
log scan retry (in)	**true** | **false**	この同期でトランザクションログが初めてスキャンされる場合、この値は False、それ以外の場合は True。Mobile Link サーバーと dbmlsync でスキャン開始位置の情報が異なっている場合、ログは 2 回スキャンされます。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、dbmlsync がトランザクションログをスキャンしてアップロードをアセンブルする直前に呼び出されます。

このフックは、アップロードに含める同期中のテーブルに最新の変更を加える場合に適しています。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「同期イベントフックの順序」196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
   "event_id"          integer NOT NULL DEFAULT autoincrement ,
   "event_time"          timestamp NULL,
   "event_name"            varchar(128) NOT NULL ,
   "subs"              varchar(1024) NULL ,
    PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにログスキャンの始めにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_begin ()
BEGIN

    DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
    SELECT LIST(value) INTO subs_list
     FROM #hook_dict
     WHERE name LIKE 'subscription_%';

-- log the event
    INSERT INTO SyncLog(event_time, event_name, subs)
    VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_logscan_begin', subs_list );
END
```

sp_hook_dbmlsync_logscan_end

このストアドプロシージャーを使用して、トランザクションログがスキャンされた直後にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
ending log offset (in)	数値	スキャンの終了位置を示すログオフセット値。
starting log offset_*n* (in)	数値	同期する各サブスクリプションの初期進行値。*n* 値は、publication_*n* の値に対応します。たとえば、Starting log offset_1 は publication_1 のオフセットです。
log scan retry (in)	**true** | **false**	この同期でトランザクションログが初めてスキャンされる場合、この値は False、それ以外の場合は True。Mobile Link サーバーと dbmlsync でスキャン開始位置の情報が異なっている場合、ログは 2 回スキャンされます。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、dbmlsync がトランザクションログをスキャンした直後に呼び出されます。

このプロシージャーのアクションは、実行直後にコミットされます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「同期イベントフックの順序」196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"          timestamp NULL,
  "event_name"            varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにログスキャンの終わりにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_end ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_logscan_end', subs_list );
END
```

sp_hook_dbmlsync_misc_error

このストアドプロシージャーを使用して、データベースエラーまたは通信エラーに分類されない dbmlsync エラーを処理します。たとえば、sp_hook_dbmlsync_misc_error フックを実装すると、特定のエラーが発生した場合に、ログを取ったり特定のアクションを実行したりすることができます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。

名前	値	説明
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
error message (in)	エラーメッセージテキスト	これは、dbmlsync ログに表示されるテキストと同じです。
error id (in)	整数	メッセージをユニークに識別する ID。このローを使用すると、エラーメッセージテキストが変更されたときに、エラーメッセージを識別できます。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

同期を開始する前の起動中にエラーが発生した場合、#hook_dict 内の Mobile Link ユーザーとスクリプトバージョンのエントリは空の文字列に設定され、#hook_dict テーブルで設定される publication_*n* ローや subscription_*n* ローはありません。

error hook user state ローは、エラーの内容を sp_hook_dbmlsync_end フックに渡す場合に役立つメカニズムを提供します。フックでは、この情報を使用して同期をリトライするかどうかを決定できます。

このプロシージャーは別個の接続で実行されるため、同期接続でロールバックが実行されても、このプロシージャーで実行する操作が失われることはありません。dbmlsync が別個の接続を確立できないと、プロシージャーは呼び出されません。

このフックは別の接続で実行されるため、フックプロシージャーで同期されているテーブルにアクセスする場合は注意してください。これらのテーブルは dbmlsync によってロックされている可能性があるためです。これらのロックが原因で、フック内の操作が失敗したり、無期限に待機したりすることがあります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「イベントフックプロシージャー内でのエラーと警告の処理」200 ページ
- 「sp_hook_dbmlsync_communication_error」207 ページ
- 「sp_hook_dbmlsync_all_error」203 ページ
- 「sp_hook_dbmlsync_sql_error」244 ページ

例

次のテーブルを使用して、リモートデータベース内のエラーのログを取ります。

```
CREATE TABLE error_log
(
 pk INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,
 err_id INTEGER,
 err_msg VARCHAR(10240),
);
```

次の例では、sp_hook_dbmlsync_misc_error を設定して、全種類のエラーメッセージのログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_misc_error()
BEGIN
 DECLARE msg VARCHAR(10240);
 DECLARE id INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error id
 SELECT value INTO id
  FROM #hook_dict
  WHERE name = 'error id';

 // log the error information
 INSERT INTO error_log(err_msg,err_id)
  VALUES (msg,id);
END;
```

エラーの可能性がある ID 値を表示するには、dbmlsync をテスト実行します。たとえば、次の dbmlsync コマンドラインは、無効なサブスクリプションを参照します。

```
dbmlsync -c SERVER=custdb;UID=DBA;PWD=sql -s test
```

ここで、error_log テーブルには次のローが含まれ、このエラーはエラー ID 9931 に関連付けられます。

```
1,19912,
 'Subscription "test" not found.'
```

カスタムエラー処理を行うには、sp_hook_dbmlsync_misc_error でエラー ID 19912 を確認します。

```
ALTER PROCEDURE sp_hook_dbmlsync_misc_error()
BEGIN
 DECLARE msg VARCHAR(10240);
```

```
DECLARE id INTEGER;

// get the error message text
SELECT value INTO msg
 FROM #hook_dict
 WHERE name ='error message';

// get the error id
SELECT value INTO id
 FROM #hook_dict
 WHERE name = 'error id';

// log the error information
INSERT INTO error_log(err_msg,err_id)
 VALUES (msg,id);

IF id = 19912 THEN
 // handle invalid subscription
END IF;

END;
```

sp_hook_dbmlsync_ml_connect_failed

このストアドプロシージャーを使用して、Mobile Link サーバーに対する接続が失敗した場合に異なる通信タイプまたはアドレスを使用してリトライします。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
connection address (in|out)	接続アドレス	フックが呼び出されるとき、これは失敗した直前の通信の試行で使用されたアドレスです。この値を新しい接続アドレスに設定して通信を試行できます。retry を true に設定すると、次の接続の試行でこの値が使用されます。 プロトコルオプションのリストについては、「Mobile Link クライアントネットワークプロトコルオプション」 24 ページを参照してください。
connection type (in|out)	ネットワークプロトコル	フックが呼び出されるとき、これは失敗した直前の通信の試行で使用されたネットワークプロトコル (TCPIP など) です。この値を新しいネットワークプロトコルに設定して通信を試行できます。retry を true に設定すると、次の接続の試行でこの値が使用されます。 ネットワークプロトコルのリストについては、「CommunicationType (ctp) 拡張オプション」 144 ページを参照してください。
user data (in|out)	ユーザー定義のデータ	次の通信の試行が失敗した場合に使用されるステータス情報。たとえば、発生したリトライの回数を保存すると便利です。デフォルトは、空の文字列です。
allow remote ahead (in|out)	**true** | **false**	この同期に対して dbmlsync の -ra オプションまたは RemoteProgressGreater=on 同期プロファイルオプションが指定された場合にのみ true になります。このローの値を変更すると、現在の同期のオプションの値のみを変更できます。 「-r dbmlsync オプション」127 ページを参照してください。
allow remote behind (in|out)	**true** | **false**	この同期に対して dbmlsync の -ra オプションまたは RemoteProgressLess=on 同期プロファイルオプションが指定された場合にのみ true になります。このローの値を変更すると、現在の同期のオプションの値のみを変更できます。 「-r dbmlsync オプション」 127 ページを参照してください。

名前	値	説明
retry (in|out)	**true** | **false**	失敗した接続の試行をリトライする場合にこの値を true に設定します。デフォルト値は FALSE です。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、Mobile Link サーバーへの接続で dbmlsync が失敗したときにそのプロシージャーが呼び出されます。

このフックが適用されるのは、データベースへの接続の試行ではなく、Mobile Link サーバーへの接続の試行に対してだけです。

進行オフセット不一致が発生した場合、dbmlsync は Mobile Link サーバーから切断してから再接続します。この種の再接続では、このフックが呼び出されず、再接続が失敗すると同期も失敗します。

このプロシージャーのアクションは、実行直後にコミットされます。

例

この例は、sp_hook_dbmlsync_ml_connect_failed フックを使用して最大 5 回まで接続をリトライします。

```
CREATE PROCEDURE sp_hook_dbmlsync_ml_connect_failed ()
BEGIN
   DECLARE idx integer;

   SELECT IF value = ''then 0 else cast(value as integer)endif
     INTO idx
     FROM #hook_dict
     WHERE name = 'user data';

   IF idx < 5 THEN
      UPDATE #hook_dict
      SET value = idx +1
      WHERE name = 'user data';

      UPDATE #hook_dict
      SET value = 'TRUE'
      WHERE name = 'retry';
  END IF;
END;
```

次に、接続情報が含まれるテーブルを使用する例を示します。接続の試行が失敗すると、フックはリストの次のサーバーを試行します。

```
CREATE TABLE conn_list (
   label   INTEGER PRIMARY KEY,
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
    addr  VARCHAR( 128 ),
    type  VARCHAR( 64 )
);
INSERT INTO conn_list
 VALUES ( 1, 'host=server1;port=91', 'tcpip' );
INSERT INTO conn_list
 VALUES ( 2, 'host=server2;port=92', 'http' );
INSERT INTO conn_list
 VALUES ( 3, 'host=server3;port=93', 'tcpip' );
COMMIT;

CREATE PROCEDURE sp_hook_dbmlsync_ml_connect_failed ()
BEGIN
 DECLARE idx INTEGER;
 DECLARE cnt INTEGER;

 SELECT if value = ''then | else cast(value as integer)endif
  INTO idx
  FROM #hook_dict
  WHERE name = 'user data';

 SELECT COUNT( label ) INTO cnt FROM conn_list;

 IF idx <= cnt THEN
   UPDATE #hook_dict
      SET value = ( SELECT addr FROM conn_list WHERE label = idx )
      WHERE name = 'connection address';
   UPDATE #hook_dict
    SET value = (SELECT type FROM conn_list WHERE label=idx)
    WHERE name = 'connection type';

   UPDATE #hook_dict
    SET value = idx +1
       WHERE name = 'user data';

   UPDATE #hook_dict
    SET value = 'TRUE'
    WHERE name = 'retry';
 END IF;
END;
```

sp_hook_dbmlsync_process_exit_code

このストアドプロシージャーを使用して、終了コードを管理します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。

名前	値	説明
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
fatal error (in)	**true** | **false**	dbmlsync を終了させる原因となるエラーのためにこのフックが呼び出されるときは true。
aborted synchronization (in)	**true** | **false**	sp_hook_dbmlsync_abort フックからのアボート要求のためにこのフックが呼び出されるときは true。
exit code (in)	数値	直近の同期試行からの終了コード。0 は同期が成功したことを示します。他の値は同期が失敗したことを示します。この値は、そのフックを使用して同期をアボートするとき、sp_hook_dbmlsync_abort によって設定できます。
last exit code (in)	数値	最後にこのフックが呼び出されたときに #hook_dict テーブルの **new exit code** ローに格納される値、またはこれがフックへの最初の呼び出しの場合は 0。
new exit code (in|out)	数値	そのプロセスに対して選択した終了コード。dbmlsync が終了するとき、dblmsync の **exit code** はそのフックへの最後の呼び出しによってこのローに格納される値です。この値は -32768 から 32767 になります。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

コマンドラインで -n オプションまたは -s オプションを複数回指定する場合、スケジューリングを使用する場合、および sp_hook_dbmlsync_end で restart パラメーターを使用する場合、dbmlsync セッションは複数の同期を実行できます。これらの状況では、1 つ以上の同期が失敗すると、デフォルトの終了コードによってどれが失敗したのかが示されません。このフックを使用して、同期からの終了コード値に基づいた dbmlsync プロセスの終了コード値を定義できます。また、このフックを使用して終了コード値のログを取ることもできます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

同期を開始する前の起動中にエラーが発生した場合、#hook_dict 内の Mobile Link ユーザーとスクリプトバージョンのエントリは空の文字列に設定され、#hook_dict テーブルで設定される publication_*n* ローや subscription_*n* ローはありません。

例

dbmlsync を実行して 5 つの同期を行い、終了コードで失敗した同期の数を示すとします。たとえば、終了コード 0 は失敗がないことを示し、1 は 1 つの同期が失敗したことを示します。これを実現するには、sp_hook_dbmlsync_process_exit_code フックを次のように定義します。この場合、3 つの同期が失敗すると新しい終了コードは 3 になります。

```
CREATE PROCEDURE sp_hook_dbmlsync_process_exit_code()
BEGIN
  DECLARE  rc INTEGER;

  SELECT value INTO rc FROM #hook_dict WHERE name = 'exit code';
  IF rc <> 0 THEN
    SELECT value INTO rc FROM #hook_dict WHERE name = 'last exit code';
    UPDATE #hook_dict SET value = rc + 1 WHERE name = 'new exit code';
  END IF;
END;
```

参照

- 「同期イベントフックの順序」196 ページ
- 「sp_hook_dbmlsync_abort」201 ページ

sp_hook_dbmlsync_schema_upgrade

このストアドプロシージャーを使用して、スキーマを修正する SQL スクリプトを実行します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
script version (in)	スクリプトバージョンの名前	同期に使用されるスクリプトバージョン。

名前	値	説明
drop hook (out)	**never** | **always** | **on success**	次のいずれかの値を取ります。 **never** - (デフォルト) データベースから sp_hook_dbmlsync_schema_upgrade フックを削除しません。 **always** - フックの実行を試みた後、データベースから sp_hook_dbmlsync_schema_upgrade フックを削除します。 **on success** - フックの実行に成功した場合、データベースから sp_hook_dbmlsync_schema_upgrade フックを削除します。dbmlsync の -eh オプションが使用されている場合、dbmlsync の拡張オプション IgnoreHookErrors が True に設定されている場合、または IgnoreHookErrors 同期プロファイルオプションが on に設定されている場合、on success は always と同じです。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

このフックは、主に下位互換性のために提供されています。ScriptVersion 拡張オプションを使用していない場合は、START SYNCHRONIZATION SCHEMA CHANGE 文を使用することによって、このフックを使用せずにスキーマの変更を安全に実行することができます。

このフックを実装すると、dbmlsync はデフォルトで同期中のテーブルをロックします。

このストアドプロシージャーは、配備されたリモートデータベースでスキーマの変更を行うためのものです。スキーマ更新のためにこのフックを使用すると、スキーマが更新される前にリモートデータベースのすべての変更が同期されます。そうされることによって、データベースの同期が確実に継続されます。このフックが使用中の場合、dbmlsync の拡張オプション LockTables を off に設定しないでください。

同期中に Mobile Link によってアップロードが正常に適用され、受信確認されると、このフックは sp_hook_dbmlsync_download_end フックの後、sp_hook_dbmlsync_end フックの前に呼び出されます。このフックは、ダウンロード専用の同期中、またはファイルベースのダウンロードが作成または適用されるときには呼び出されません。

このフックで実行されるアクションは、フックが完了した直後にコミットされます。このフックでコミットまたはロールバックを行っても安全です。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』
- 「リモート MobiLink クライアントでのスキーマの変更」64 ページ

例

次の例では、sp_hook_dbmlsync_schema_upgrade プロシージャーを使用して、リモートデータベース上の Dealer テーブルにカラムを追加します。アップグレードが成功すると、sp_hook_dbmlsync_schema_upgrade フックは削除されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_schema_upgrade()
BEGIN
 -- Upgrade the schema of the Dealer table. Add a column:
 ALTER TABLE Dealer
  ADD dealer_description VARCHAR(128);

 -- Change the script version used to synchronize
 ALTER SYNCHRONIZATION SUBSCRIPTION sub1
  SET SCRIPT VERSION='v2';


 -- If the schema upgrade is successful, drop this hook:
 UPDATE #hook_dict
  SET value = 'on success'
  WHERE name = 'drop hook';
END;
```

sp_hook_dbmlsync_set_extended_options

同期に適用される拡張オプションを指定して、次の同期の動作をプログラムによってカスタマイズするには、このストアドプロシージャーを使用します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
extended options (out)	*opt*=*val*;...	次の同期のために追加される拡張オプション。

名前	値	説明
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、各同期の前に 1 回以上呼び出されます。

このフックで指定される拡張オプションは、サブスクリプションと Mobile Link ユーザーエントリによって識別される同期にのみ適用され、このフックが同じ同期を対象として次に呼び出されるまで適用されます。

このフックを使用して、Schedule 拡張オプションを指定することはできません。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「同期イベントフックの順序」 196 ページ
- 「Mobile Link SQL Anywhere クライアントの拡張オプション」 137 ページ
- 「優先順位」 137 ページ

例

次の例では、sp_hook_dbmlsync_set_extended_options を使用して、SendColumnNames 拡張オプションを指定します。この拡張オプションは、sub1 が同期される場合にのみ適用されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_set_extended_options ()
BEGIN
 IF exists(SELECT * FROM #hook_dict
  WHERE name LIKE 'subscription_%' AND value='sub1')
 THEN
   -- specify the SendColumnNames=on extended option
   UPDATE #hook_dict
         SET value = 'SendColumnNames=on'
    WHERE name = 'extended options';
 END IF;
END;
```

sp_hook_dbmlsync_set_ml_connect_info

このストアドプロシージャーを使用して、Mobile Link サーバーへの接続に使用されるネットワークプロトコルとネットワークプロトコルオプションを設定します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン
connection type (in/out)	tcpip、tls、http、https	Mobile Link サーバーへの接続に使用するネットワークプロトコル。
connection address (in/out)	プロトコルオプション	Mobile Link サーバーへの接続に使用する通信アドレス。 「Mobile Link クライアントネットワークプロトコルオプション」 24 ページを参照してください。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

このフックを使用して、接続タイプや接続アドレスのローの値を変更することによって、Mobile Link サーバーへの接続に使用されるネットワークプロトコルとネットワークプロトコルオプションを設定できます。

プロトコルとオプションは sp_hook_dbmlsync_set_extended_options でも設定できます。sp_hook_dbmlsync_set_extended_options は、同期の開始時に呼び出されるフックです。sp_hook_dbmlsync_set_ml_connect_info は、dbmlsync が Mobile Link サーバーへの接続を開始する直前に呼び出されます。

このフックは、フック内でオプションを設定したいが、同期プロセスで sp_hook_dbmlsync_set_extended_options よりも後で設定したい場合に便利です。たとえば、使用しているネットワークの信号の強さを取得できるかどうかに基づいてオプションを設定できます。

参照

- 「SQL Anywhere クライアントのイベントフック」 195 ページ
- 「同期イベントフックの順序」 196 ページ
- 「CommunicationType (ctp) 拡張オプション」 144 ページ
- 「Mobile Link クライアントネットワークプロトコルオプション」 24 ページ
- 「sp_hook_dbmlsync_set_extended_options」 239 ページ

例

```
CREATE PROCEDURE sp_hook_dbmlsync_set_ml_connect_info()
begin
   UPDATE #hook_dict
   SET VALUE = 'tcpip'
     WHERE name = 'connection type';

   UPDATE #hook_dict
   SET VALUE = 'host=localhost'
     WHERE name = 'connection address';
end
```

sp_hook_dbmlsync_set_upload_end_progress

このストアドプロシージャーは、スクリプト化されたアップロードサブスクリプションが同期されるときの終了進行状況を定義するために使用されます。このプロシージャーが呼び出されるのは、スクリプト化されたアップロードパブリケーションの同期中だけです。

#hook_dict テーブルのロー

名前	値	説明
generating download exclusion list (in)	TRUE | FALSE	同期中にアップロードが送信されない場合 (ダウンロード専用の同期や、ファイルベースのダウンロードが適用される場合など) は TRUE。この場合でもアップロードスクリプトは呼び出され、生成した操作は、アップロードする必要があるローを変更するダウンロード操作の識別に使用します。このような操作が見つかると、ダウンロードは適用されません。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
start progress as timestamp_*n* (in)	進行状況 (タイムスタンプ)	同期中の各サブスクリプションの開始進行状況がタイムスタンプで示されます。ここで、*n* は、サブスクリプション名の識別に使用する整数と同じです。
start progress as bigint_*n* (in)	進行状況 (bigint)	同期中の各サブスクリプションの開始進行状況が bigint で示されます。ここで、*n* は、サブスクリプション名の識別に使用する整数と同じです。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
end progress is bigint (in|out)	TRUE | FALSE	このローが TRUE に設定されている場合は、終了進行状況の値は、文字列 ('12345' など) として表される符号なし bigint であると見なされます。 このローが FALSE に設定されている場合は、終了進行状況の値は、文字列 ('1900/01/01 12:00:00.000' など) として表される TIMESTAMP であると見なされます。 デフォルト値は FALSE です。
end progress (in|out)	タイムスタンプ	フックはこのローを変更して、アップロードスクリプトに渡される "end progress" と "raw end progress" の値を変更できます。これらの値は、生成中のアップロードにすべての操作が含まれているかどうかの境界となる時点を定義します。 このローの値は、"end progress is bigint" ローの設定に応じて、符号なし bigint またはタイムスタンプのいずれかに設定できます。このローのデフォルト値は、現在のタイムスタンプです。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

スクリプト化されたアップロードの場合は、アップロードプロシージャーが呼び出されるたびに、開始進行状況と終了進行状況の値が渡されます。プロシージャーは、これら 2 つの値で定義された期間に発生した適切な操作をすべて返す必要があります。開始進行状況値は、最後に成功した同期での終了進行状況値と同じです。ただし、今回初めて同期を行っている場合、開始進行状況値は "January 1, 1900, 00:00:00.000" になります。終了進行状況値は、デフォルトで、dbmlsync がアップロードを構築し始めた時間です。

このフックを使用すると、デフォルトの終了進行状況値を上書きできます。アップロードには短い時間を定義したり、タイムスタンプ以外の方法 (世代番号など) に基づいた、進行状況を追跡するスキームを実装することができます。

"end progress is bigint" が true に設定されている場合、終了進行状況は、1900-01-01 00:00:00 から 9999-12-31 23:59:59:9999 の間のミリ秒数 (255,611,203,259,999) 以下の整数でなければなりません。

参照

- 「スクリプト化されたアップロードのカスタム進行状況値」320 ページ
- 「同期イベントフックの順序」196 ページ
- 「スクリプト化されたアップロード」311 ページ

sp_hook_dbmlsync_sql_error

このストアドプロシージャーを使用して、同期中に発生したデータベースエラーを処理します。たとえば、sp_hook_dbmlsync_sql_error フックを実装すると、特定の SQL エラーが発生した場合に、特定のアクションを実行することができます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
error message (in)	エラーメッセージテキスト	これは、dbmlsync ログに表示されるテキストと同じです。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
error id (in)	数値	メッセージをユニークに識別する ID。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。
SQLCODE (in)	SQLCODE	操作が失敗したときにデータベースから返される SQLCODE。 「SQL Anywhere のエラーメッセージ (SQLCODE 順)」『エラーメッセージ』を参照してください。
SQLSTATE (in)	SQLSTATE 値	操作が失敗したときにデータベースから返される SQLSTATE。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

同期を開始する前の起動中にエラーが発生した場合、#hook_dict 内の Mobile Link ユーザーとスクリプトバージョンのエントリは空の文字列に設定され、#hook_dict テーブルで設定される publication_*n* ローや subscription_*n* ローはありません。

SQL エラーは、SQL Anywhere の SQLCODE または ANSI SQL 規格の SQLSTATE を使用して識別できます。 SQLCODE または SQLSTATE 値のリストについては、「SQL Anywhere のエラーメッセージ」『エラーメッセージ』を参照してください。

error hook user state ローは、エラーの内容を sp_hook_dbmlsync_end フックに渡す場合に役立つメカニズムを提供します。フックでは、この情報を使用して同期をリトライするかどうかを決定できます。

このプロシージャーは別個の接続で実行されるため、同期接続でロールバックが実行されても、このプロシージャーで実行する操作が失われることはありません。dbmlsync が別個の接続を確立できないと、プロシージャーは呼び出されません。

このフックは別の接続で実行されるため、フックプロシージャーで同期されているテーブルにアクセスする場合は注意してください。これらのテーブルは dbmlsync によってロックされている可能性があるためです。これらのロックが原因で、フック内の操作が失敗したり、無期限に待機したりすることがあります。

このプロシージャーのアクションは、実行直後にコミットされます。

参照

- 「イベントフックプロシージャー内でのエラーと警告の処理」200 ページ
- 「sp_hook_dbmlsync_all_error」203 ページ
- 「sp_hook_dbmlsync_communication_error」207 ページ
- 「sp_hook_dbmlsync_misc_error」229 ページ
- 「SQL Anywhere のエラーメッセージ」『エラーメッセージ』

sp_hook_dbmlsync_upload_begin

このストアドプロシージャーを使用して、アップロード転送の直前にカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
Publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー
Script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、dbmlsync がアップロードを送信する直前に呼び出されます。

このプロシージャーのアクションは、実行直後にコミットされます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- 「同期イベントフックの順序」196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"          timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにアップロードの始めにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_begin ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_upload_begin', subs_list );
END
```

sp_hook_dbmlsync_upload_end

Mobile Link サーバーによってアップロードが受信されたことを dbmlsync で確認した後に、このストアドプロシージャーを使用してカスタムアクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
failure cause (in)	「備考」の項の値の範囲を参照してください。	アップロード障害の原因。詳細については、「備考」を参照してください。

名前	値	説明
upload status (in)	**retry** | **committed** | **failed** | **unknown**	dbmlsync がアップロードの受信確認を行おうとしたときに、Mobile Link サーバーから返されるステータスを指定します。 **retry** - アップロードの開始位置であるログオフセットの値が、Mobile Link サーバーと dbmlsync で異なっていました。Mobile Link サーバーは、アップロードをコミットしませんでした。dbmlsync ユーティリティは、新しいログオフセットから始まる別のアップロードを送信しようとします。 **committed** - Mobile Link サーバーがアップロードを受信し、コミットしました。 **failed** - Mobile Link サーバーは、アップロードをコミットしませんでした。 **unknown** – Mobile Link サーバーは、アップロードを確認しませんでした。アップロードがコミットされたかどうかを確認する方法はありません。
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー。
script version (in)	スクリプトバージョン名	同期に使用される Mobile Link スクリプトバージョン

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
authentication value (in)	値	この値は、dbmlsync による Mobile Link サーバーに対する認証結果を示します。サーバー上の authenticate_user スクリプト、authenticate_user_hashed スクリプト、または authenticate_parameters スクリプトによって生成されます。upload status が unknown であるとき、またはリモートデータベースと統合データベースに格納されているログオフセット間の競合が原因でアップロードが再送信された後に upload_end フックが呼び出されたとき、値は空の文字列になります。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

この名前のプロシージャーが存在する場合、dbmlsync がアップロードを送信し、Mobile Link サーバーから受信確認を受け取った直後に呼び出されます。

トランザクション単位のアップロードまたはインクリメンタルアップロードを実行する場合、アップロードの各セグメントの送信後にこのフックが呼び出されます。この場合、最後に呼び出される場合を除き、フックが呼び出されるたびにアップロードステータスは「不明」になります。

このプロシージャーのアクションは、実行直後にコミットされます。

#hook_dict テーブルの failure cause ローに有効なパラメーター値の範囲は、次のとおりです。

- **UPLD_ERR_INVALID_USERID_OR_PASSWORD**　ユーザー ID またはパスワードが正しくありません。
- **UPLD_ERR_USERID_OR_PASSWORD_EXPIRED**　ユーザー ID またはパスワードの有効期限が切れています。
- **UPLD_ERR_REMOTE_ID_ALREADY_IN_USE**　このリモート ID はすでに使用されています。
- **UPLD_ERR_SQLCODE_n**　この *n* は整数です。統合データベースに SQL エラーが発生しました。指定された整数は、発生したエラーを表す SQLCODE です。
- **UPLD_ERR_USER_ABORT_REQUEST**　ユーザーの要求によりアップロードが中止されました。

参照

- 「同期イベントフックの順序」 196 ページ

例

次のテーブルを使用して、リモートデータベース上の同期イベントのログを取ります。

```
CREATE TABLE SyncLog
(
  "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_time"         timestamp NULL,
  "event_name"          varchar(128) NOT NULL ,
  "subs"              varchar(1024) NULL ,
   PRIMARY KEY ("event_id")
)
```

次の文は、同期ごとにアップロードの終わりにログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end ()
BEGIN

   DECLARE subs_list VARCHAR(1024);

-- build a list of subscriptions being synchronized
   SELECT LIST(value) INTO subs_list
    FROM #hook_dict
    WHERE name LIKE 'subscription_%';

-- log the event
   INSERT INTO SyncLog(event_time, event_name, subs)
   VALUES( CURRENT TIMESTAMP, 'sp_hook_dbmlsync_upload_end', subs_list );
END
```

sp_hook_dbmlsync_validate_download_file

このフックを使用して、ダウンロードファイルがリモートデータベースに適用できるかどうかを決定するためのカスタム論理を実装します。このフックは、ファイルベースのダウンロードが適用される場合のみ呼び出されます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。同期されるパブリケーションごとに 1 つの publication_*n* エントリがあります。publication_*n* と generation number_*n* の *n* は一致します。*n* の番号は 0 から始まります。
MobiLink user (in)	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	値	説明
file last download time (in)		ダウンロードファイルの最後のダウンロード時刻 (ダウンロードファイルには、最後のダウンロード時刻とその前のダウンロード時刻の間に変更されたすべてのローが含まれます)。
file next last download time (in)		ダウンロードファイルの最後から 2 番目のダウンロード時刻 (ダウンロードファイルには、最後のダウンロード時刻とその前のダウンロード時刻の間に変更されたすべてのローが含まれます)。
file creation time (in)		ダウンロードファイルが作成された時間。
file generation number_*n* (in)	数値	ダウンロードファイルからの世代番号。各 subscription_*n* エントリに対して、1 つのファイル世代番号 number_*n* があります。subscription_*n* と 世代番号 number_*n* の *n* は一致します。*n* の番号は 0 から始まります。
user data (in)	文字列	ダウンロードファイルが作成されたときの dbmlsync -be オプションまたは DnldFileExtra 同期プロファイルオプションで指定した文字列。
apply file (in|out)	**True**|**False**	True (デフォルト) の場合、ダウンロードファイルは dbmlsync の他の検証チェックを通過したときだけ適用されます。False の場合、ダウンロードファイルはリモートデータベースに適用されません。
check generation number (in|out)	**True**|**False**	True (デフォルト) の場合、dbmlsync は世代番号を検証します。ダウンロードファイル内の世代番号がリモートデータベース内の世代番号と一致しない場合、dbmlsync はダウンロードファイルを適用しません。False の場合、dbmlsync は世代番号をチェックしません。

名前	値	説明
setting generation number (in)	**true** | **false**	ダウンロードファイルが作成されたときに -bg オプションまたは UpdateGenNum 同期プロファイルオプションが使用された場合は true。true の場合、リモートデータベースの世代番号はダウンロードファイルから更新され、通常の世代番号チェックは行われません。
subscription_*n* (in)	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。

備考

このストアドプロシージャーを使用して、ダウンロードファイルが適用できるか決定するためのカスタムチェックを実装します。

ダウンロードファイルに含まれる世代番号またはタイムスタンプと、リモートデータベースに格納されている世代番号またはタイムスタンプを比較する場合、それらのデータは SYSSYNC システムビューから問い合わせることができます。

ファイルベースのダウンロードがリモートデータベースに適用される前にこのフックが呼び出されます。

このフックのアクションは、フックが完了した直後にコミットされます。

参照

- 「-be dbmlsync オプション」 109 ページ
- 「-bg dbmlsync オプション」 109 ページ
- 「Mobile Link ファイルベースのダウンロード」『Mobile Link サーバー管理』

例

次の例では、ユーザー文字列「sales manager data」を含まないダウンロードファイルを適用しません。

```
CREATE PROCEDURE sp_hook_dbmlsync_validate_download_file ()
BEGIN
  IF NOT exists(SELECT * FROM #hook_dict
   WHERE name = 'User data' AND value='sales manager data')
  THEN
    UPDATE #hook_dict
          SET value = 'false' WHERE name = 'Apply file';
  END IF;
END;
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Dbmlsync C++ API リファレンス

ヘッダーファイル

dbmlsynccli.hpp

例

この後に示す例は、Dbmlsync API の C++ バージョンを使用して同期を行い、出力イベントを受け取る典型的なアプリケーションを示したものです。この例では、わかりやすいようにエラー処理を省略しています。各 API 呼び出しの戻り値を確認する習慣をつけることをおすすめします。

```
#include <stdio.h>
#include "dbmlsynccli.hpp"


int main( void ) {
    DbmlsyncClient *client;
    DBSC_SyncHdl    syncHdl;
    DBSC_Event     *ev1;

    client = DbmlsyncClient::InstantiateClient();
    if( client == NULL ) return( 1 );
    client->Init();

    // Setting the "server path" is usually required on Windows Mobile.
    // In other environments the server path is usually not required unless
    // your SQL Anywhere install is not in your path or you have multiple
    // versions of the product installed.
    client->SetProperty( "server path", "C:¥¥SQLAnywhere¥¥bin32" );

    client->StartServer( 3426,
            "-c server=remote;dbn=rem1;uid=dba;pwd=sql -v+ -ot c:¥¥dbsync1.txt",
            5000, NULL );
    client->Connect( NULL, 3426, "dba", "sql");
    syncHdl = client->Sync( "my_sync_profile", "" );
    while( client->GetEvent( &ev1, 5000) == DBSC_GETEVENT_OK ) {
        if( ev1->hdl == syncHdl ) {
            //
            // Process events that interest you here
            //

            if( ev1->type == DBSC_EVENTTYPE_SYNC_DONE ) {
                client->FreeEventInfo( ev1 );
                break;
            }
            client->FreeEventInfo( ev1 );
        }
    }
    client->ShutdownServer( DBSC_SHUTDOWN_ON_EMPTY_QUEUE );
    client->WaitForServerShutdown( 10000 );
    client->Disconnect();
    client->Fini();
    delete client;

    return( 0 );
}
```

DbmlsyncClient クラス

TCP/IP を使用して別のプロセス (dbmlsync サーバー) と通信します。このプロセスが、Mobile Link サーバーとリモートデータベースに接続することによって同期を実行します。

構文

```
public class DbmlsyncClient
```

メンバー

継承されたメンバーを含む DbmlsyncClient クラスのすべてのメンバー。

名前	説明
CancelSync メソッド	同期要求をキャンセルします。
Connect メソッド	すでにこのコンピューターで実行されている dbmlsync サーバーとの接続を開きます。
Disconnect メソッド	Connect メソッドを使用して確立された dbmlsync サーバーとの接続を切断します。
Fini メソッド	このクラスインスタンスによって使用されているすべてのリソースを解放します。
FreeEventInfo メソッド	GetEvent メソッドによって返された DBSC_Event 構造体に関連付けられていたメモリを解放します。
GetErrorInfo メソッド	DbmlsyncClient クラスメソッドによって失敗を示すリターンコードが返された後、失敗に関する追加情報を取得します。
GetEvent メソッド	クライアントが要求した同期の次のフィードバックイベントを取得します。
GetProperty メソッド	プロパティの現在の値を取得します。
Init メソッド	DbmlsyncClient クラスインスタンスを初期化します。
InstantiateClient メソッド	同期の制御に使用できる dbmlsync クライアントクラスのインスタンスを作成します。
Ping メソッド	dbmlsync サーバーに ping 要求を送信して、サーバーがアクティブで、要求に応答しているかどうかをチェックします。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

名前	説明
SetProperty メソッド	各種のプロパティを設定して、クラスインスタンスの動作を変更します。
ShutdownServer メソッド	クライアントの接続先である dbmlsync サーバーを停止します。
StartServer メソッド	指定したポートでまだ受信していない場合は、新しい dbmlsync サーバーを起動します。
Sync メソッド	同期を実行するよう、dbmlsync サーバーに要求します。
WaitForServerShutdown メソッド	サーバーが停止したときかタイムアウトになったときのどちらか早い方で戻ります。

備考

複数のクライアントで同じ dbmlsync サーバーを共有できます。ただし、各 dbmlsync サーバーが同期できるのは 1 つのリモートデータベースのみです。各リモートデータベースには、同期する dbmlsync サーバーを 1 つのみ設定できます。

dbmlsync サーバーは、一度に 1 つの同期を実行します。サーバーは同期の実行中に同期要求を受信した場合、その要求をキューイングし、後で履行します。

同期によって生成されたステータス情報は、GetEvent メソッドを通じてクライアントアプリケーションに送り返されます。

参照

- DbmlsyncClient.GetEvent メソッド [Dbmlsync C++]260 ページ

CancelSync メソッド

同期要求をキャンセルします。

オーバーロードリスト

名前	説明
CancelSync(DBSC_SyncHdl) メソッド (旧式)	Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。
CancelSync(DBSC_SyncHdl, bool) メソッド	Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

CancelSync(DBSC_SyncHdl) メソッド (旧式)

Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

構文

```
public virtual bool CancelSync(DBSC_SyncHdl hdl)
```

パラメーター

- **hdl** 同期が要求されたときに Sync メソッドによって返された同期ハンドル。

戻り値

同期要求が正常にキャンセルされた場合は true、正常にキャンセルされなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

キャンセルできるのは、サービスが提供されるのを待っている同期要求のみです。すでに開始された同期を停止するには、CancelSync(UInt32, Boolean) メソッドを使用します。

ShutdownServer メソッドを使用して DBSC_SHUTDOWN_CLEANLY タイプを渡すと、アクティブな同期をキャンセルできます。dbmlsync サーバーは停止する前に同期のキャンセルを試みます。このタスクは、DBTools インターフェイスを使用した同期のキャンセルと同じです。

このメソッドを使用するには、サーバーへの接続が確立されている必要があります。Sync メソッドを呼び出した後、クライアントがサーバーから切断されている場合は、このメソッドを使用することはできません。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ
- DbmlsyncClient.CancelSync メソッド [Dbmlsync C++]255 ページ
- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync C++]264 ページ

CancelSync(DBSC_SyncHdl, bool) メソッド

Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

構文

```
public virtual DBSC_CancelRet CancelSync(
     DBSC_SyncHdl hdl,
    bool cancel_active
)
```

パラメーター

- **hdl** 同期が要求されたときに Sync メソッドによって返された同期ハンドル。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

- **cancel_active**　true に設定されている場合は、同期がすでに開始していても要求がキャンセルされます。false に設定されている場合は、同期が開始していない場合にのみ要求がキャンセルされます。

戻り値

DBSC_CancelRet 列挙の値。DBSC_CANCEL_FAILED が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

ShutdownServer メソッドを使用して DBSC_SHUTDOWN_CLEANLY タイプを渡すと、アクティブな同期をキャンセルできます。dbmlsync サーバーは停止する前に同期のキャンセルを試みます。このタスクは、DBTools インターフェイスを使用した同期のキャンセルと同じです。

このメソッドを使用するには、サーバーへの接続が確立されている必要があります。Sync メソッドを呼び出した後、クライアントがサーバーから切断されている場合は、このメソッドを使用することはできません。

参照

- DBSC_CancelRet 列挙 [Dbmlsync C++]268 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ
- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync C++]264 ページ

Connect メソッド

すでにこのコンピューターで実行されている dbmlsync サーバーとの接続を開きます。

構文

```
public virtual bool Connect(
     const char * host,
    unsigned port,
    const char * uid,
    const char * pwd
)
```

パラメーター

- **host**　この値は予約されています。NULL を使用します。
- **port**　dbmlsync サーバーが受信している TCP ポート。StartServer メソッドを使用して指定した port 値と同じ port 値を使用します。
- **uid**　同期されるリモートデータベースに対する DBA 権限または REMOTE DBA 権限を持つ、有効なデータベースユーザー ID。
- **pwd**　uid で指定されたユーザーのデータベースパスワード。

戻り値

サーバーへの接続が確立された場合は true、確立されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

データベースユーザー ID とパスワードを使用して、このクライアントがデータベースの同期に必要なパーミッションを持っているかどうかが検証されます。同期が実行されるときは、dbmlsync サーバーの起動時に -c オプションで指定したユーザー ID が使用されます。

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync C++]265 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

Disconnect メソッド

Connect メソッドを使用して確立された dbmlsync サーバーとの接続を切断します。

構文

```
public virtual bool Disconnect(void)
```

戻り値

サーバーへの接続が切断された場合は true、切断されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

接続を使い終わったら、必ず Disconnect を呼び出してください。

参照

- DbmlsyncClient.Connect メソッド [Dbmlsync C++]257 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

Fini メソッド

このクラスインスタンスによって使用されているすべてのリソースを解放します。

構文

```
public virtual bool Fini(void)
```

戻り値

クラスインスタンスが正常に終了された場合は true、正常に終了されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

DbmlSyncClient クラスインスタンスを削除するには、このメソッドを呼び出しておく必要があります。

> **注意**
> Disconnect メソッドを使用して接続しているすべてのサーバーを切断してから、クラスインスタンスを終了してください。

参照

- DbmlsyncClient.Disconnect メソッド [Dbmlsync C++]258 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

FreeEventInfo メソッド

GetEvent メソッドによって返された DBSC_Event 構造体に関連付けられていたメモリを解放します。

構文

```
public virtual bool FreeEventInfo(DBSC_Event * event)
```

パラメーター

- **event**　解放する DBSC_Event 構造体へのポインター。

戻り値

メモリが正常に解放された場合は true、正常に解放されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

GetEvent メソッドによって返された DBSC_Event 構造体それぞれに対して、FreeEventInfo を呼び出してください。

参照

- DBSC_Event 構造体 [Dbmlsync C++]276 ページ
- DbmlsyncClient.GetEvent メソッド [Dbmlsync C++]260 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

GetErrorInfo メソッド

DbmlsyncClient クラスメソッドによって失敗を示すリターンコードが返された後、失敗に関する追加情報を取得します。

構文

```
public virtual const DBSC_ErrorInfo * GetErrorInfo(void)
```

戻り値

失敗に関する情報を含む DBSC_ErrorInfo 構造体へのポインター。この構造体の内容は、クラスメソッドが次回呼び出されたときに上書きされることがあります。

参照

- DBSC_ErrorType 列挙 [Dbmlsync C++]268 ページ
- DBSC_ErrorInfo 構造体 [Dbmlsync C++]275 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

GetEvent メソッド

クライアントが要求した同期の次のフィードバックイベントを取得します。

構文

```
public virtual DBSC_GetEventRet GetEvent(
     DBSC_Event ** event,
    unsigned timeout
)
```

パラメーター

- **event**　戻り値が DBSC_GETEVENT_OK である場合、event パラメーターに、取得されたイベントに関する情報を含む DBSC_Event 構造体へのポインターが入力されます。event の構造体を使い終わったら、FreeEventInfo メソッドを呼び出して、event の構造体に関連付けられていたメモリを解放してください。

- **timeout**　すぐに返すことができるイベントがない場合に待つ最大のミリ秒数を指定します。DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

DBSC_GetEventRet 列挙の値。DBSC_GETEVENT_FAILED が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

フィードバックイベントには、sync から生成されたメッセージ、進行状況バーを更新するためのデータ、同期サイクル通知などの情報が含まれています。

dbmlsync サーバーは、同期を実行しながら、同期の進行状況に関する情報を含む一連のイベントを生成します。これらのイベントは、サーバーから DbmlsyncClient クラスに送信され、そこでキューイングされます。GetEvent メソッドを呼び出すと、キュー内で待機している次のイベントがある場合、そのイベントが返されます。

キュー内で待機しているイベントがない場合、このメソッドは、イベントが実行可能になるまで、または指定されたタイムアウトになるまで待ってから、戻ります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

同期用に生成されるイベントのタイプは、プロパティを使用して制御できます。

参照

- DBSC_GetEventRet 列挙 [Dbmlsync C++]273 ページ
- DBSC_Event 構造体 [Dbmlsync C++]276 ページ
- DbmlsyncClient.FreeEventInfo メソッド [Dbmlsync C++]259 ページ
- DbmlsyncClient.SetProperty メソッド [Dbmlsync C++]263 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

GetProperty メソッド

プロパティの現在の値を取得します。

構文

```
public virtual bool GetProperty(const char * name, char * value)
```

パラメーター

- **name**　取り出すプロパティの名前。有効なプロパティ名のリストについては、SetProperty を参照してください。
- **value**　プロパティの値を格納する、DBSC_MAX_PROPERTY_LEN バイト以上のバッファー。

戻り値

プロパティが正常に受信された場合は true、正常に受信されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

参照

- DbmlsyncClient.SetProperty メソッド [Dbmlsync C++]263 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

Init メソッド

DbmlsyncClient クラスインスタンスを初期化します。

構文

```
public virtual bool Init(void)
```

戻り値

クラスインスタンスが正常に初期化された場合は true、正常に初期化されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

DbmlSyncClient クラスインスタンスをインスタンス化した後、このメソッドを呼び出してください。インスタンスが正常に初期化されるまで、他の DbmlSyncClient メソッドを呼び出すことはできません。

参照

- DbmlsyncClient.InstantiateClient メソッド [Dbmlsync C++]262 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

InstantiateClient メソッド

同期の制御に使用できる dbmlsync クライアントクラスのインスタンスを作成します。

構文

```
public static DbmlsyncClient * InstantiateClient(void)
```

戻り値

作成された新しいインスタンスへのポインター。エラーが発生した場合は NULL を返します。

備考

このメソッドによって返されるポインターを使用して、クラス内の残りのメソッドを呼び出すことができます。ポインターで標準の削除操作を呼び出すことで、インスタンスを破棄できます。

Ping メソッド

dbmlsync サーバーに ping 要求を送信して、サーバーがアクティブで、要求に応答しているかどうかをチェックします。

構文

```
public virtual bool Ping(unsigned timeout)
```

パラメーター

- **timeout**　サーバーが ping 要求に応答するのを待つ最大のミリ秒数。DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

ping 要求に対する応答をサーバーから受信した場合は true、受信しなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーに接続してから、このメソッドを呼び出してください。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

SetProperty メソッド

各種のプロパティを設定して、クラスインスタンスの動作を変更します。

構文

```
public virtual bool SetProperty(const char * name, const char * value)
```

パラメーター

- **name**　設定するプロパティの名前。有効なプロパティ名のリストについては、表を参照してください。
- **value**　プロパティに設定する値。指定する文字列は、DBCS_MAX_PROPERTY_LEN バイト未満にしてください。

戻り値

プロパティが正常に設定された場合は true、正常に設定されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

プロパティ値への変更は、変更後に行われた同期要求にのみ反映されます。

server path プロパティを設定して、StartServer メソッドが呼び出されたときにクライアントが dbmlsync.exe を起動するディレクトリを指定できます。このプロパティが設定されていない場合、PATH 環境変数を使用して dbmlsync.exe が検索されます。コンピューターに複数のバージョンの SQL Anywhere がインストールされている場合は、**server path** プロパティを使用して dbmlsync.exe のロケーションを指定することをおすすめします。これは、PATH 環境変数によって、インストールされた別のバージョンの SQL Anywhere の dbmlsync 実行プログラムが検出される可能性があるためです。次に例を示します。

```
ret = cli->SetProperty("server path", "c:¥¥sa12¥¥bin32");
```

プロパティは、GetEvent メソッドによって返されるイベントのタイプを制御します。不要なイベントを無効にすることによって、パフォーマンスを向上できることがあります。イベントタイプは、対応するプロパティを 1 に設定すると有効になり、0 に設定すると無効になります。

次の表に、使用可能なプロパティ名と、各プロパティによって制御されるイベントタイプを示します。

プロパティ名	制御されるイベントタイプ	デフォルト値
enable errors	DBSC_EVENTTYPE_ERROR_MSG	1

プロパティ名	制御されるイベントタイプ	デフォルト値
enable warnings	DBSC_EVENTTYPE_WARNING_MSG	1
enable info msgs	DBSC_EVENTTYPE_INFO_MSG	1
enable progress	DBSC_EVENTTYPE_PROGRESS_INDEX	0
enable progress text	DBSC_EVENTTYPE_PROGRESS_TEXT	0
enable title	DBSC_EVENTTYPE_TITLE	0
enable sync start and done	DBSC_EVENTTYPE_SYNC_START DBSC_EVENTTYPE_SYNC_DONE	1
enable status	DBSC_EVENTTYPE_ML_CONNECT DBSC_EVENTTYPE_UPLOAD_COMMITTED DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	1

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync C++]265 ページ
- DbmlsyncClient.GetEvent メソッド [Dbmlsync C++]260 ページ
- DbmlsyncClient.GetProperty メソッド [Dbmlsync C++]261 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

ShutdownServer メソッド

クライアントの接続先である dbmlsync サーバーを停止します。

構文

```
public virtual bool ShutdownServer(DBSC_ShutdownType how)
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

パラメーター

- **how**　サーバー停止の緊急度を示します。サポートされる値は DBSC_ShutdownType 列挙にリストされます。

戻り値

停止要求がサーバーに正常に送信された場合は true、正常に送信されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

Shutdown メソッドはすぐに戻りますが、サーバーが実際に停止するまでに遅延が生じることがあります。

WaitForServerShutdown メソッドを使用すると、サーバーが実際に停止するまで待つことができます。

注意
ShutdownServer を呼び出した後も、Disconnect メソッドを使用してください。

参照

- DBSC_ShutdownType 列挙 [Dbmlsync C++]274 ページ
- DbmlsyncClient.Disconnect メソッド [Dbmlsync C++]258 ページ
- DbmlsyncClient.WaitForServerShutdown メソッド [Dbmlsync C++]267 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

StartServer メソッド

指定したポートでまだ受信していない場合は、新しい dbmlsync サーバーを起動します。

構文

```
public virtual bool StartServer(
     unsigned port,
    const char * cmdline,
    unsigned timeout,
    DBSC_StartType * starttype
)
```

パラメーター

- **port**　既存の dbmlsync サーバーがないかどうかをチェックする TCP ポート。サーバーは、新しく起動される場合、このポートで受信するように設定されます。

- **cmdline**　dbmlsync サーバーを起動するための有効なコマンドライン。コマンドラインには、次のオプションのみを含めることができます。これらのオプションは、dbmlsync ユーティリティに対して持つ意味と同じ意味を持ちます。-a、-c、-dl、-do、-ek、-ep、-k、-l、-o、-os、-ot、-p、-pc+、-pc-、-pd、-pp、-q、-qi、-qc、-sc、-sp、-uc、-ud、-ui、-um、-un、-ux、-v[cnoprsut]、-wc、-wh。-c オプションは必ず指定します。

- **timeout** dbmlsync サーバーが起動された後、要求を受け入れる準備が完了するまでの最大待ち時間 (ミリ秒単位)。DBSC_INFINITY を使用して応答を無期限に待機します。

- **starttype** サーバーが検出または起動されたかどうかを示すために設定される出力パラメーター。starttype がエントリ時に NULL 以外の値であり、かつ StartServer が true を返した場合、終了時に starttype が指している変数は DBSC_StartType 列挙の値に設定されます。

戻り値

サーバーがすでに実行されている場合、または正常に起動された場合は true、それ以外の場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーがある場合、このメソッドは starttype パラメーターを DBSC_SS_ALREADY_RUNNING に設定し、その他の処理を行わずに戻ります。サーバーが見つからない場合、cmdline 引数で指定されたオプションを使用して新しいサーバーを起動し、そのサーバーが要求を受け入れ始めるまで待ってから、戻ります。

Windows Mobile デバイスでは通常、StartServer を正常に呼び出すには、先に **server path** プロパティを設定する必要があります。ただし、次の場合は **server path** プロパティを設定する必要はありません。

●アプリケーションが *dbmlsync.exe* と同じディレクトリにある。

●*dbmlsync.exe* が Windows ディレクトリにある。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

Sync メソッド

同期を実行するよう、dbmlsync サーバーに要求します。

構文

```
public virtual DBSC_SyncHdl Sync(
      const char * profile_name,
     const char * extra_opts
)
```

パラメーター

- **profile_name** 同期のオプションを含む同期プロファイルの名前で、リモートデータベース内に定義されているもの。profile_name が NULL の場合、プロファイルは使用されないため、extra_opts パラメーターに、同期に使用するすべてのオプションが指定されている必要があります。

- **extra_opts** 同期プロファイルのオプション文字列を定義するときと同じルールに従った形式の文字列。<オプション名>=<オプション値> 形式のエレメントのセミコロンで

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

区切ったリストとして指定される文字列です。profile_name が NULL 以外の場合、extra_opts で指定したオプションは、profile_name で指定した同期プロファイルにすでにあるオプションに追加されます。プロファイルに文字列のオプションがすでに存在する場合は、プロファイルにすでに格納済みの値が文字列の値に置き換わります。profile_name が NULL の場合、extra_opts に、同期に使用するすべてのオプションが指定されている必要があります。「CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

戻り値

この同期要求をユニークに識別する DBSC_SyncHdl 値を返します。この値は、クライアントがサーバーから切断するまでの間でのみ有効です。エラーのために同期要求を作成できなかった場合は NULL_SYNCHDL を返します。NULL_SYNCHDL が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーに接続してから、このメソッドを呼び出してください。profile_name と extra_opts のうち少なくとも 1 つを NULL 以外の値にしてください。

戻り値は同期要求を識別し、要求のキャンセルや同期によって返されるイベントの処理に使用できます。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

WaitForServerShutdown メソッド

サーバーが停止したときかタイムアウトになったときのどちらか早い方で戻ります。

構文

```
public virtual bool WaitForServerShutdown(unsigned timeout)
```

パラメーター

- **timeout**　サーバーが停止するまでの最大待ち時間 (ミリ秒) を示します。DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

サーバーのシャットダウンによりメソッドが返された場合は true、それ以外の場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

WaitForServerShutdown を呼び出すには、先に ShutdownServer メソッドを呼び出す必要があります。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync C++]259 ページ

DBSC_CancelRet 列挙

同期キャンセル試行の結果を示します。

構文

```
public enum DBSC_CancelRet
```

メンバー

メンバー名	説明	値
DBSC_CANCEL_OK_QUEUED	待機キュー内の同期をキャンセルしました。	1
DBSC_CANCEL_OK_ACTIVE	アクティブな同期をキャンセルしました。	2
DBSC_CANCEL_FAILED	同期をキャンセルできませんでした。	3

参照

- DbmlsyncClient.CancelSync メソッド [Dbmlsync C++]255 ページ

DBSC_ErrorType 列挙

メソッド呼び出しが失敗した理由を示します。

構文

```
public enum DBSC_ErrorType
```

メンバー

メンバー名	説明	値
DBSC_ERR_OK	エラーは発生しませんでした。	1
DBSC_ERR_NOT_INITIALIZED	クラスが、Init メソッドを呼び出すことによって初期化されていません。	2

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_ERR_ALREADY_INITIALIZED	すでに初期化されたクラスに対して、Init メソッドが呼び出されました。	3
DBSC_ERR_NOT_CONNECTED	dbmlsync サーバーへの接続がありません。	4
DBSC_ERR_CANT_RESOLVE_HOST	ホスト情報を解決できません。	5
DBSC_ERR_CONNECT_FAILED	dbmlsync サーバーへの接続が失敗しました。	6
DBSC_ERR_INITIALIZING_TCP_LAYER	TCP レイヤーの初期化中にエラーが発生しました。	7
DBSC_ERR_ALREADY_CONNECTED	すでに接続が確立されているため、Connect メソッドが失敗しました。	8
DBSC_ERR_PROTOCOL_ERROR	これは内部エラーです。	9
DBSC_ERR_CONNECTION_REJECTED	dbmlsync サーバーによって接続が拒否されました。 str1 は、サーバーによって返された文字列を指します。この文字列には、接続試行が拒否された理由の詳細が含まれることがあります。	10
DBSC_ERR_TIMED_OUT	サーバーからの応答を待っている間にタイムアウトになりました。	11
DBSC_ERR_STILL_CONNECTED	クラスがまだサーバーに接続されているため、そのクラスに対して Fini を実行できませんでした。	12
DBSC_ERR_SYNC_NOT_CANCELED	サーバーが同期要求をキャンセルできませんでした。同期がすでに進行中であることが原因と思われます。	14

メンバー名	説明	値
DBSC_ERR_INVALID_VALUE	SetProperty メソッドに無効なプロパティ値が渡されました。	15
DBSC_ERR_INVALID_PROP_NAME	指定したプロパティ名は無効です。	16
DBSC_ERR_VALUE_TOO_LONG	プロパティ値が長すぎます。プロパティは、DBCS_MAX_PROPERTY_LEN で指定されているバイト数よりも短くしてください。	17
DBSC_ERR_SERVER_SIDE_ERROR	sync のキャンセル中または追加中にサーバー側でエラーが発生しました。 str1 は、サーバーによって返された文字列を指します。この文字列には、エラーの詳細が含まれることがあります。	18
DBSC_ERR_CREATE_PROCESS_FAILED	新しい dbmlsync サーバーを起動できません。	20
DBSC_ERR_READ_FAILED	dbmlsync サーバーからのデータの読み込み中に TCP エラーが発生しました。	21
DBSC_ERR_WRITE_FAILED	dbmlsync サーバーへのデータの送信中に TCP エラーが発生しました。	22
DBSC_ERR_NO_SERVER_RESPONSE	要求されたアクションを完了するために必要な応答をサーバーから受信できませんでした。	23
DBSC_ERR_UID_OR_PWD_TOO_LONG	指定した UID または PWD が長すぎます。	24
DBSC_ERR_UID_OR_PWD_NOT_VALID	指定した UID または PWD が無効です。	25
DBSC_ERR_INVALID_PARAMETER	関数に渡されたパラメーターのいずれかが無効です。	26

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_ERR_WAIT_FAILED	サーバーが停止するのを待っている間にエラーが発生しました。	27
DBSC_ERR_SHUTDOWN_NOT_CALLED	ShutdownServer メソッドを呼び出す前に WaitForServerShutdown メソッドが呼び出されました。	28
DBSC_ERR_NO_SYNC_ACK	同期要求がサーバーに送信されましたが、受信確認が受信されませんでした。サーバーが要求を受信したかどうかを判別する方法はありません。 hdl1 は、送信された同期要求のハンドルです。サーバーが要求を受信した場合は、このハンドルを使用すると、GetEvent を使用して取得された同期用イベントを識別できます。	29
DBSC_ERR_ACTIVE_SYNC_NOT_CANCELED	同期がアクティブであるため、サーバーは同期をキャンセルできませんでした。	30
DBSC_ERR_DEAD_SERVER	dbmlsync サーバーで起動時にエラーが発生しました。 サーバーを停止しています。dbmlsync の -o オプションを使用して、ファイルにエラーメッセージのログを記録してください。	31

DBSC_EventType 列挙

同期によって生成されたイベントのタイプを示します。

構文

```
public enum DBSC_EventType
```

メンバー

メンバー名	説明	値
DBSC_EVENTTYPE_ERROR_MSG	同期によってエラーが生成されました。str1 はエラーのテキストを指します。	1
DBSC_EVENTTYPE_WARNING_MSG	同期によって警告が生成されました。str1 は警告のテキストを指します。	2
DBSC_EVENTTYPE_INFO_MSG	同期によって情報メッセージが生成されました。str1 はメッセージのテキストを指します。	3
DBSC_EVENTTYPE_PROGRESS_INDEX	進行状況バーを更新するための情報を提供します。val1 は、新しい進行状況値を格納します。 完了した割合 (パーセント) を計算するには、val1 を 1000 で割ります。	4
DBSC_EVENTTYPE_PROGRESS_TEXT	進行状況バーに関連付けられているテキストが更新されました。新しい値は、str1 が指します。	6
DBSC_EVENTTYPE_TITLE	同期ウィンドウ／コントロールのタイトルが変更されました。新しいタイトルは、str1 が指します。	7
DBSC_EVENTTYPE_SYNC_START	同期が開始しました。このイベントに関連付けられている追加情報はありません。	8
DBSC_EVENTTYPE_SYNC_DONE	同期が完了しました。val1 は、同期からの終了コードを格納します。 0 という値は、成功を示します。0 以外の値は、同期が失敗したことを示します。	9

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_EVENTTYPE_ML_CONNECT	Mobile Link サーバーへの接続が確立されました。str1 は、使用されている通信プロトコルを示します。str2 は、使用されるネットワークプロトコルオプションを示します。	10
DBSC_EVENTTYPE_UPLOAD_COMMITTED	Mobile Link サーバーは統合データベースへのアップロードを正常にコミットしたことを確認しました。	11
DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	ダウンロードはリモートデータベースで正常にコミットされました。	12

参照

- DBSC_Event 構造体 [Dbmlsync C++]276 ページ
- DbmlsyncClient.GetEvent メソッド [Dbmlsync C++]260 ページ

DBSC_GetEventRet 列挙

イベント取得の試行の結果を示します。

構文

```
public enum DBSC_GetEventRet
```

メンバー

メンバー名	説明	値
DBSC_GETEVENT_OK	イベントが正常に取得されたことを示します。	1
DBSC_GETEVENT_TIMED_OUT	返すことができるイベントがないまま、タイムアウトになったことを示します。	2
DBSC_GETEVENT_FAILED	エラーが発生したためにイベントが返されなかったことを示します。	3

参照

- DbmlsyncClient.GetEvent メソッド [Dbmlsync C++]260 ページ

DBSC_ShutdownType 列挙

サーバーを停止する緊急度を示します。

構文

```
public enum DBSC_ShutdownType
```

メンバー

メンバー名	説明	値
DBSC_SHUTDOWN_ON_EMPTY_QUEUE	サーバーが未処理の同期要求を完了してから停止する必要があることを示します。 サーバーは、停止要求を受信すると、それよりも後の同期要求を受け入れません。	1
DBSC_SHUTDOWN_CLEANLY	サーバーができるだけ早く正常に停止する必要があることを示します。 未処理の同期要求は実行されません。実行中の同期は中断されることがあります。	2

参照

- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync C++]264 ページ

DBSC_StartType 列挙

dbmlsync サーバーを起動しようとしているときに実行されたアクションを示します。

構文

```
public enum DBSC_StartType
```


Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー

メンバー名	説明	値
DBSC_SS_STARTED	新しい dbmlsync サーバーが起動されたことを示します。	1
DBSC_SS_ALREADY_RUNNING	既存の dbmlsync サーバーが見つかったため、新しいサーバーが起動されなかったことを示します。	2

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync C++]265 ページ

DBSC_ErrorInfo 構造体

以前のメソッド呼び出しの失敗に関する情報が含まれています。

構文

```
public typedef struct DBSC_ErrorInfo
```

メンバー

メンバー名	タイプ	説明
hdl1	DBSC_SyncHdl	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
str1	const char *	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
str2	const char *	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

メンバー名	タイプ	説明
type	DBSC_ErrorType	失敗の理由を示す値が含まれています。 サポートされる値は DBSC_ErrorType 列挙にリストされます。
val1	long int	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
val2	long int	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

備考

str1、str2、val1、val2、hdl1 には失敗に関する詳細が含まれ、その意味はエラータイプによって異なります。次のエラータイプはこの構造体のフィールドを使用して詳細を格納します。

- DBSC_ERR_CONNECTION_REJECTED
- DBSC_ERR_SERVER_SIDE_ERROR
- DBSC_ERR_NO_SYNC_ACK

参照

- DBSC_ErrorType 列挙 [Dbmlsync C++]268 ページ

DBSC_Event 構造体

同期によって生成されたイベントに関する情報が含まれています。

構文

```
public typedef struct DBSC_Event
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー

メンバー名	タイプ	説明
data	void *	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
hdl	DBSC_SyncHdl	イベントを生成した同期を示します。 この値は、Sync メソッドによって返される値と一致します。
str1	const char *	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
str2	const char *	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
type	DBSC_EventType	レポートされるイベントのタイプを示します。
val1	long int	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
val2	long int	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

参照

- DBSC_EventType 列挙 [Dbmlsync C++]271 ページ

Dbmlsync .NET API リファレンス

ネームスペース

```
iAnywhere.MobiLink.Client
```

例

この後に示す例は、Dbmlsync API の .NET バージョンを使用して同期を行い、出力イベントを受け取る典型的なアプリケーションを示したものです。この例では、わかりやすいようにエラー処理を省略しています。各 API 呼び出しの戻り値を確認する習慣をつけることをおすすめします。

```
using System;
using System.Collections.Generic;
using System.Text;
using iAnywhere.MobiLink.Client;

namespace ConsoleApplication6
{
    class Program
    {
        static void Main(string[] args)
        {
            DbmlsyncClient cli1;
            DBSC_StartType st1;
            DBSC_Event ev1;
            UInt32 syncHdl;

            cli1 = DbmlsyncClient.InstantiateClient();
            cli1.Init();

            // Setting the "server path" is usually required on Windows
            // Mobile/CE. In other environments the server path is usually
            // not required unless you SA install is not in your path or
            // you have multiple versions of the product installed
            cli1.SetProperty("server path", "d:¥¥sybase¥¥asa1100r¥¥bin32");

            cli1.StartServer(3426,
             "-c server=cons;dbn=rem1;uid=dba;pwd=sql -ve+ -ot c:¥¥dbsync1.txt",
             5000, out st1);
            cli1.Connect(null, 3426, "dba", "sql");
            syncHdl = cli1.Sync("sp1", "");
            while (cli1.GetEvent(out ev1, 5000)
                    == DBSC_GetEventRet.DBSC_GETEVENT_OK)
            {
                if (ev1.hdl == syncHdl)
                {
                    Console.WriteLine("Event Type : {0}", ev1.type);
                    if (ev1.type == DBSC_EventType.DBSC_EVENTTYPE_INFO_MSG)
                    {
                        Console.WriteLine("Info : {0}", ev1.str1);
                    }
                    if (ev1.type == DBSC_EventType.DBSC_EVENTTYPE_SYNC_DONE)
                    {
                        break;
                    }
                }
            }
            cli1.ShutdownServer(DBSC_ShutdownType.DBSC_SHUTDOWN_ON_EMPTY_QUEUE);
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
            cli1.WaitForServerShutdown(10000);
            cli1.Disconnect();
            cli1.Fini();
            Console.ReadLine();
        }
    }
}
```

DbmlsyncClient クラス

TCP/IP を使用して別のプロセス (dbmlsync サーバー) と通信します。このプロセスが、Mobile Link サーバーとリモートデータベースに接続することによって同期を実行します。

Visual Basic の構文

```
Public Class DbmlsyncClient
```

C# の構文

```
public class DbmlsyncClient
```

メンバー

継承されたメンバーを含む DbmlsyncClient クラスのすべてのメンバー。

名前	説明
CancelSync メソッド	同期要求をキャンセルします。
Connect メソッド	すでにこのコンピューターで実行されている dbmlsync サーバーとの接続を開きます。
Disconnect メソッド	Connect メソッドを使用して確立された dbmlsync サーバーとの接続を切断します。
Fini メソッド	このクラスインスタンスによって使用されているすべてのリソースを解放します。
GetErrorInfo メソッド	DbmlsyncClient クラスメソッドによって失敗を示すリターンコードが返された後、失敗に関する追加情報を取得します。
GetEvent メソッド	クライアントが要求した同期の次のフィードバックイベントを取得します。
GetProperty メソッド	プロパティの現在の値を取得します。
Init メソッド	DbmlsyncClient クラスインスタンスを初期化します。

名前	説明
InstantiateClient メソッド	同期の制御に使用できる dbmlsync クライアントクラスのインスタンスを作成します。
Ping メソッド	dbmlsync サーバーに ping 要求を送信して、サーバーがアクティブで、要求に応答しているかどうかをチェックします。
SetProperty メソッド	各種のプロパティを設定して、クラスインスタンスの動作を変更します。
ShutdownServer メソッド	クライアントの接続先である dbmlsync サーバーを停止します。
StartServer メソッド	指定したポートでまだ受信していない場合は、新しい dbmlsync サーバーを起動します。
Sync メソッド	同期を実行するよう、dbmlsync サーバーに要求します。
WaitForServerShutdown メソッド	サーバーが停止したときかタイムアウトになったときのどちらか早い方で戻ります。

備考

複数のクライアントで同じ dbmlsync サーバーを共有できます。ただし、各 dbmlsync サーバーが同期できるのは 1 つのリモートデータベースのみです。各リモートデータベースには、同期する dbmlsync サーバーを 1 つのみ設定できます。

dbmlsync サーバーは、一度に 1 つの同期を実行します。サーバーは同期の実行中に同期要求を受信した場合、その要求をキューイングし、後で履行します。

同期によって生成されたステータス情報は、GetEvent メソッドを通じてクライアントアプリケーションに送り返されます。

参照

- DbmlsyncClient.GetEvent メソッド [Dbmlsync .NET]285 ページ

CancelSync メソッド

同期要求をキャンセルします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

オーバーロードリスト

名前	説明
CancelSync(UInt32) メソッド (旧式)	Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。
CancelSync(UInt32, Boolean) メソッド	Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

CancelSync(UInt32) メソッド (旧式)

Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

Visual Basic 構文

```
Public Function CancelSync(ByVal hdl As UInt32) As Boolean
```

C# 構文

```
public Boolean CancelSync(UInt32 hdl)
```

パラメーター

- **hdl** 同期が要求されたときに Sync メソッドによって返された同期ハンドル。

戻り値

同期要求が正常にキャンセルされた場合は true、正常にキャンセルされなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

キャンセルできるのは、サービスが提供されるのを待っている同期要求のみです。すでに開始された同期を停止するには、CancelSync(UInt32, Boolean) メソッドを使用します。

ShutdownServer メソッドを使用して DBSC_SHUTDOWN_CLEANLY タイプを渡すと、アクティブな同期をキャンセルできます。dbmlsync サーバーは停止する前に同期のキャンセルを試みます。このタスクは、DBTools インターフェイスを使用した同期のキャンセルと同じです。

このメソッドを使用するには、サーバーへの接続が確立されている必要があります。Sync メソッドを呼び出した後、クライアントがサーバーから切断されている場合は、このメソッドを使用することはできません。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ
- DbmlsyncClient.CancelSync メソッド [Dbmlsync .NET]280 ページ
- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync .NET]290 ページ

CancelSync(UInt32, Boolean) メソッド

Sync メソッドを使用して直前に行われた同期要求を、クライアントがキャンセルできるようにします。

Visual Basic 構文

```
Public Function CancelSync(
     ByVal hdl As UInt32,
    ByVal cancel_active As Boolean
) As DBSC_CancelRet
```

C# 構文

```
public DBSC_CancelRet CancelSync(UInt32 hdl, Boolean cancel_active)
```

パラメーター

- **hdl** 同期が要求されたときに Sync メソッドによって返された同期ハンドル。
- **cancel_active** true に設定されている場合は、同期がすでに開始していると要求がキャンセルされます。false に設定されている場合は、同期がまだ開始していない場合にのみ要求がキャンセルされます。

戻り値

DBSC_CancelRet 列挙の値。DBSC_CANCEL_FAILED が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

キャンセルできるのは、サービスが提供されるのを待っている同期要求のみです。

ShutdownServer メソッドを使用して DBSC_SHUTDOWN_CLEANLY タイプを渡すと、アクティブな同期をキャンセルできます。dbmlsync サーバーは停止する前に同期のキャンセルを試みます。このタスクは、DBTools インターフェイスを使用した同期のキャンセルと同じです。

このメソッドを使用するには、サーバーへの接続が確立されている必要があります。Sync メソッドを呼び出した後、クライアントがサーバーから切断されている場合は、このメソッドを使用することはできません。

参照

- DBSC_CancelRet 列挙 [Dbmlsync .NET]294 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ
- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync .NET]290 ページ

Connect メソッド

すでにこのコンピューターで実行されている dbmlsync サーバーとの接続を開きます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Visual Basic 構文

```
Public Function Connect(
     ByVal host As String,
    ByVal port As Int32,
    ByVal uid As String,
    ByVal pwd As String
) As Boolean
```

C# 構文

```
public Boolean Connect(String host, Int32 port, String uid, String pwd)
```

パラメーター

- **host**　この値は予約されています。C# を使用している場合は、NULL を指定します。Visual Basic を使用している場合は、何も指定しないでください。
- **port**　dbmlsync サーバーが受信している TCP ポート。StartServer メソッドを使用して指定した port 値と同じ port 値を使用します。
- **uid**　同期されるリモートデータベースに対する DBA 権限または REMOTE DBA 権限を持つ、有効なデータベースユーザー ID。
- **pwd**　uid で指定されたユーザーのデータベースパスワード。

戻り値

サーバーへの接続が確立された場合は true、確立されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

データベースユーザー ID とパスワードを使用して、このクライアントがデータベースの同期に必要なパーミッションを持っているかどうかが検証されます。同期が実行されるときは、dbmlsync サーバーの起動時に -c オプションで指定したユーザー ID が使用されます。

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync .NET]291 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

Disconnect メソッド

Connect メソッドを使用して確立された dbmlsync サーバーとの接続を切断します。

Visual Basic 構文

```
Public Function Disconnect() As Boolean
```

C# 構文

```
public Boolean Disconnect()
```

戻り値

サーバーへの接続が切断された場合は true、切断されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

接続を使い終わったら、必ず Disconnect を呼び出してください。

参照

- DbmlsyncClient.Connect メソッド [Dbmlsync .NET]282 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

Fini メソッド

このクラスインスタンスによって使用されているすべてのリソースを解放します。

Visual Basic 構文

```
Public Function Fini() As Boolean
```

C# 構文

```
public Boolean Fini()
```

戻り値

クラスインスタンスが正常に終了された場合は true、正常に終了されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

DbmlSyncClient クラスインスタンスを削除するには、このメソッドを呼び出しておく必要があります。

注意
Disconnect メソッドを使用して接続しているすべてのサーバーを切断してから、クラスインスタンスを終了してください。

参照

- DbmlsyncClient.Disconnect メソッド [Dbmlsync .NET]283 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

GetErrorInfo メソッド

DbmlsyncClient クラスメソッドによって失敗を示すリターンコードが返された後、失敗に関する追加情報を取得します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Visual Basic 構文

```
Public Function GetErrorInfo() As DBSC_ErrorInfo
```

C# 構文

```
public DBSC_ErrorInfo GetErrorInfo()
```

戻り値

失敗に関する情報を含む DBSC_ErrorInfo 構造体へのポインター。この構造体の内容は、クラスメソッドが次回呼び出されたときに上書きされることがあります。

参照

- DBSC_ErrorType 列挙 [Dbmlsync .NET]295 ページ
- DBSC_ErrorInfo 構造体 [Dbmlsync .NET]302 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

GetEvent メソッド

クライアントが要求した同期の次のフィードバックイベントを取得します。

Visual Basic 構文

```
Public Function GetEvent(
     ByVal ev As DBSC_Event,
    ByVal timeout As UInt32
) As DBSC_GetEventRet
```

C# 構文

```
public DBSC_GetEventRet GetEvent(out DBSC_Event ev, UInt32 timeout)
```

パラメーター

- **ev**　戻り値が DBSC_GETEVENT_OK である場合、ev パラメーターに、取得されたイベントに関する情報が入力されます。

- **timeout**　すぐに返すことができるイベントがない場合に待つ最大のミリ秒数を指定します。DbmlsyncClient.DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

DBSC_GetEventRet 列挙の値。DBSC_GETEVENT_FAILED が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

フィードバックイベントには、sync から生成されたメッセージ、進行状況バーを更新するためのデータ、同期サイクル通知などの情報が含まれています。

dbmlsync サーバーは、同期を実行しながら、同期の進行状況に関する情報を含む一連のイベントを生成します。これらのイベントは、サーバーから DbmlsyncClient クラスに送信され、そこで

キューイングされます。GetEvent メソッドを呼び出すと、キュー内で待機している次のイベントがある場合、そのイベントが返されます。

キュー内で待機しているイベントがない場合、このメソッドは、イベントが実行可能になるまで、または指定されたタイムアウトになるまで待ってから、戻ります。

同期用に生成されるイベントのタイプは、プロパティを使用して制御できます。

参照

- DBSC_GetEventRet 列挙 [Dbmlsync .NET]300 ページ
- DBSC_Event 構造体 [Dbmlsync .NET]304 ページ
- DbmlsyncClient.SetProperty メソッド [Dbmlsync .NET]288 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

GetProperty メソッド

プロパティの現在の値を取得します。

Visual Basic 構文

```
Public Function GetProperty(
     ByVal name As String,
    ByVal value As String
) As Boolean
```

C# 構文

```
public Boolean GetProperty(String name, out String value)
```

パラメーター

- **name**　取り出すプロパティの名前。有効なプロパティ名のリストについては、SetProperty を参照してください。
- **value**　終了時に、プロパティの値がこの変数に格納されます。

戻り値

プロパティが正常に受信された場合は true、正常に受信されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

参照

- DbmlsyncClient.SetProperty メソッド [Dbmlsync .NET]288 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

Init メソッド

DbmlsyncClient クラスインスタンスを初期化します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

Visual Basic 構文

```
Public Function Init() As Boolean
```

C# 構文

```
public Boolean Init()
```

戻り値

クラスインスタンスが正常に初期化された場合は true、正常に初期化されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

DbmlSyncClient クラスインスタンスをインスタンス化した後、このメソッドを呼び出してください。インスタンスが正常に初期化されるまで、他の DbmlSyncClient メソッドを呼び出すことはできません。

参照

- DbmlsyncClient.InstantiateClient メソッド [Dbmlsync .NET]287 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

InstantiateClient メソッド

同期の制御に使用できる dbmlsync クライアントクラスのインスタンスを作成します。

Visual Basic 構文

```
Public Shared Function InstantiateClient() As DbmlsyncClient
```

C# 構文

```
public static DbmlsyncClient InstantiateClient()
```

戻り値

作成された DbmlsyncClient インスタンス。エラーが発生した場合は NULL を返します。

備考

このメソッドによって返されるオブジェクトを使用して、クラス内の残りのメソッドを呼び出すことができます。

Ping メソッド

dbmlsync サーバーに ping 要求を送信して、サーバーがアクティブで、要求に応答しているかどうかをチェックします。

Visual Basic 構文

```
Public Function Ping(ByVal timeout As UInt32) As Boolean
```

C# 構文

```
public Boolean Ping(UInt32 timeout)
```

パラメーター

- **timeout** サーバーが ping 要求に応答するのを待つ最大のミリ秒数。DbmlsyncClient.DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

ping 要求に対する応答をサーバーから受信した場合は true、受信しなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーに接続してから、このメソッドを呼び出してください。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

SetProperty メソッド

各種のプロパティを設定して、クラスインスタンスの動作を変更します。

Visual Basic 構文

```
Public Function SetProperty(
     ByVal name As String,
    ByVal value As String
) As Boolean
```

C# 構文

```
public Boolean SetProperty(String name, String value)
```

パラメーター

- **name** 設定するプロパティの名前。有効なプロパティ名のリストについては、表を参照してください。
- **value** プロパティに設定する値。

戻り値

プロパティが正常に設定された場合は true、正常に設定されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

プロパティ値への変更は、変更後に行われた同期要求にのみ反映されます。

server path プロパティを設定して、StartServer メソッドが呼び出されたときにクライアントが dbmlsync.exe を起動するディレクトリを指定できます。このプロパティが設定されていない場合、PATH 環境変数を使用して dbmlsync.exe が検索されます。コンピューターに複数のバージョンの SQL Anywhere がインストールされている場合は、**server path** プロパティを使用して dbmlsync.exe のロケーションを指定することをおすすめします。これは、PATH 環境変数によって、インストールされた別のバージョンの SQL Anywhere の dbmlsync 実行プログラムが検出される可能性があるためです。次に例を示します。

```
ret = cli->SetProperty("server path", "c:¥¥sa12¥¥bin32");
```

プロパティは、GetEvent メソッドによって返されるイベントのタイプを制御します。不要なイベントを無効にすることによって、パフォーマンスを向上できることがあります。イベントタイプは、対応するプロパティを 1 に設定すると有効になり、0 に設定すると無効になります。

次の表に、使用可能なプロパティ名と、各プロパティによって制御されるイベントタイプを示します。

プロパティ名	制御されるイベントタイプ	デフォルト値
enable errors	DBSC_EVENTTYPE_ERROR_MSG	1
enable warnings	DBSC_EVENTTYPE_WARNING_MSG	1
enable info msgs	DBSC_EVENTTYPE_INFO_MSG	1
enable progress	DBSC_EVENTTYPE_PROGRESS_INDEX	0
enable progress text	DBSC_EVENTTYPE_PROGRESS_TEXT	0
enable title	DBSC_EVENTTYPE_TITLE	0
enable sync start	DBSC_EVENTTYPE_SYNC_START	1
enable sync done	DBSC_EVENTTYPE_SYNC_DONE	1

プロパティ名	制御されるイベントタイプ	デフォルト値
enable sync start and done	DBSC_EVENTTYPE_SYNC_START DBSC_EVENTTYPE_SYNC_DONE	1
enable status	DBSC_EVENTTYPE_ML_CONNECT DBSC_EVENTTYPE_UPLOAD_COMMITTED DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	1

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync .NET]291 ページ
- DbmlsyncClient.GetEvent メソッド [Dbmlsync .NET]285 ページ
- DbmlsyncClient.GetProperty メソッド [Dbmlsync .NET]286 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

ShutdownServer メソッド

クライアントの接続先である dbmlsync サーバーを停止します。

Visual Basic 構文

```
Public Function ShutdownServer(
     ByVal how As DBSC_ShutdownType
) As Boolean
```

C# 構文

```
public Boolean ShutdownServer(DBSC_ShutdownType how)
```

パラメーター

- **how**　サーバー停止の緊急度を示します。サポートされる値は DBSC_ShutdownType 列挙にリストされます。

戻り値

停止要求がサーバーに正常に送信された場合は true、正常に送信されなかった場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

備考

Shutdown メソッドはすぐに戻りますが、サーバーが実際に停止するまでに遅延が生じることがあります。

WaitForServerShutdown メソッドを使用すると、サーバーが実際に停止するまで待つことができます。

注意
ShutdownServer を呼び出した後も、Disconnect メソッドを使用してください。

参照

- DBSC_ShutdownType 列挙 [Dbmlsync .NET]301 ページ
- DbmlsyncClient.Disconnect メソッド [Dbmlsync .NET]283 ページ
- DbmlsyncClient.WaitForServerShutdown メソッド [Dbmlsync .NET]293 ページ
- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

StartServer メソッド

指定したポートでまだ受信していない場合は、新しい dbmlsync サーバーを起動します。

Visual Basic 構文

```
Public Function StartServer(
     ByVal port As Int32,
    ByVal cmdline As String,
    ByVal timeout As UInt32,
    ByVal starttype As DBSC_StartType
) As Boolean
```

C# 構文

```
public Boolean StartServer(
     Int32 port,
    String cmdline,
    UInt32 timeout,
    out DBSC_StartType starttype
)
```

パラメーター

- **port** 既存の dbmlsync サーバーがないかどうかをチェックする TCP ポート。サーバーは、新しく起動される場合、このポートで受信するように設定されます。

- **cmdline** dbmlsync サーバーを起動するための有効なコマンドライン。コマンドラインには、次のオプションのみを含めることができます。これらのオプションは、dbmlsync ユーティリティに対して持つ意味と同じ意味を持ちます。-a、-c、-dl、-do、-ek、-ep、-k、-l、-o、-os、-ot、-p、-pc+、-pc-、-pd、-pp、-q、-qi、-qc、-sc、-sp、-uc、-ud、-ui、-um、-un、-ux、-v[cnoprsut]、-wc、-wh。-c オプションは必ず指定します。

- **timeout**　dbmlsync サーバーが起動された後、要求を受け入れる準備が完了するまでの最大待ち時間 (ミリ秒単位)。DbmlsyncClient.DBSC_INFINITY を使用して応答を無期限に待機します。

- **starttype**　サーバーが検出または起動されたかどうかを示すために設定される出力パラメーター。starttype がエントリ時に NULL 以外の値であり、かつ StartServer が true を返した場合、終了時に starttype が指している変数は DBSC_StartType 列挙の値に設定されます。

戻り値

サーバーがすでに実行されている場合、または正常に起動された場合は true、それ以外の場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーがある場合、このメソッドは starttype パラメーターを DBSC_SS_ALREADY_RUNNING に設定し、その他の処理を行わずに戻ります。サーバーが見つからない場合、cmdline 引数で指定されたオプションを使用して新しいサーバーを起動し、そのサーバーが要求を受け入れ始めるまで待ってから、戻ります。

Windows Mobile デバイスでは通常、StartServer を正常に呼び出すには、先に **server path** プロパティを設定する必要があります。ただし、次の場合は **server path** プロパティを設定する必要はありません。

●アプリケーションが dbmlsync.exe と同じディレクトリにある。

●dbmlsync.exe が Windows ディレクトリにある。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

Sync メソッド

同期を実行するよう、dbmlsync サーバーに要求します。

Visual Basic 構文

```
Public Function Sync(
     ByVal syncName As String,
    ByVal opts As String
) As UInt32
```

C# 構文

```
public UInt32 Sync(String syncName, String opts)
```

パラメーター

- **syncName**　同期のオプションを含む同期プロファイルの名前で、リモートデータベース内に定義されているもの。syncName が NULL の場合、プロファイルは使用されないため、opts パラメーターに、同期に使用するすべてのオプションが指定されている必要があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

- **opts** 同期プロファイルのオプション文字列を定義するときと同じルールに従った形式の文字列。<オプション名>=<オプション値> 形式のエレメントのセミコロンで区切ったリストとして指定される文字列です。syncName が NULL でない場合、syncName で指定された同期プロファイルにすでに存在するオプションに、opts で指定されたオプションが追加されます。プロファイルに文字列のオプションがすでに存在する場合は、プロファイルにすでに格納済みの値が文字列の値に置き換わります。syncName が NULL の場合、opts に、同期に使用するすべてのオプションが指定されている必要があります。「CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]」『SQL Anywhere サーバー SQL リファレンス』 を参照してください。

戻り値

この同期要求をユニークに識別する整数値を返します。この値は、クライアントがサーバーから切断するまでの間でのみ有効です。エラーのために同期要求を作成できなかった場合は NULL_SYNCHDL を返します。NULL_SYNCHDL が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

サーバーに接続してから、このメソッドを呼び出してください。syncName と opts のうち少なくとも 1 つを NULL 以外の値にしてください。

戻り値は同期要求を識別し、要求のキャンセルや同期によって返されるイベントの処理に使用できます。

次の C# の例は、Sync メソッドを呼び出した後でエラーコードを表示する方法を示しています。

```
// Insert code to initialize the syncronization client.

UInt32 request = syncClient.Sync("syncName", null);
if (request == DbmlsyncClient.NULL_SYNCHDL) {
    string error_code = syncClient.GetErrorInfo().type.ToString();
    MessageBox.Show(error_code, "Sync Error");
}
```

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

WaitForServerShutdown メソッド

サーバーが停止したときかタイムアウトになったときのどちらか早い方で戻ります。

Visual Basic 構文

```
Public Function WaitForServerShutdown(
     ByVal timeout As UInt32
) As Boolean
```

C# 構文

```
public Boolean WaitForServerShutdown(UInt32 timeout)
```

パラメーター

- **timeout**　サーバーが停止するまでの最大待ち時間 (ミリ秒) を示します。DbmlsyncClient.DBSC_INFINITY を使用して応答を無期限に待機します。

戻り値

サーバーのシャットダウンによりメソッドが返された場合は true、それ以外の場合は false を返します。false が返されたときは、GetErrorInfo メソッドを呼び出して、失敗に関する詳細な情報を取得できます。

備考

WaitForServerShutdown を呼び出すには、先に ShutdownServer メソッドを呼び出す必要があります。

参照

- DbmlsyncClient.GetErrorInfo メソッド [Dbmlsync .NET]284 ページ

DBSC_CancelRet 列挙

同期キャンセル試行の結果を示します。

Visual Basic 構文

```
Public Enum DBSC_CancelRet
```

C# 構文

```
public enum DBSC_CancelRet
```

メンバー

メンバー名	説明	値
DBSC_CANCEL_OK_QUEUED	待機キュー内の同期をキャンセルしました。	1
DBSC_CANCEL_OK_ACTIVE	アクティブな同期をキャンセルしました。	2
DBSC_CANCEL_FAILED	同期をキャンセルできませんでした。	3

参照

- DbmlsyncClient.CancelSync メソッド [Dbmlsync .NET]280 ページ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

DBSC_ErrorType 列挙

メソッド呼び出しが失敗した理由を示します。

Visual Basic 構文

```
Public Enum DBSC_ErrorType
```

C# 構文

```
public enum DBSC_ErrorType
```

メンバー

メンバー名	説明	値
DBSC_ERR_OK	エラーは発生しませんでした。	1
DBSC_ERR_NOT_INITIALIZED	クラスが、Init メソッドを呼び出すことによって初期化されていません。	2
DBSC_ERR_ALREADY_INITIALIZED	すでに初期化されたクラスに対して、Init メソッドが呼び出されました。	3
DBSC_ERR_NOT_CONNECTED	dbmlsync サーバーへの接続がありません。	4
DBSC_ERR_CANT_RESOLVE_HOST	ホスト情報を解決できません。	5
DBSC_ERR_CONNECT_FAILED	dbmlsync サーバーへの接続が失敗しました。	6
DBSC_ERR_INITIALIZING_TCP_LAYER	TCP レイヤーの初期化中にエラーが発生しました。	7
DBSC_ERR_ALREADY_CONNECTED	すでに接続が確立されているため、Connect メソッドが失敗しました。	8
DBSC_ERR_PROTOCOL_ERROR	これは内部エラーです。	9

メンバー名	説明	値
DBSC_ERR_CONNECTION_REJECTED	dbmlsync サーバーによって接続が拒否されました。 str1 は、サーバーによって返された文字列を指します。この文字列には、接続試行が拒否された理由の詳細が含まれることがあります。	10
DBSC_ERR_TIMED_OUT	サーバーからの応答を待っている間にタイムアウトになりました。	11
DBSC_ERR_STILL_CONNECTED	クラスがまだサーバーに接続されているため、そのクラスに対して Fini を実行できませんでした。	12
DBSC_ERR_SYNC_NOT_CANCELED	サーバーが同期要求をキャンセルできませんでした。同期がすでに進行中であることが原因と思われます。	14
DBSC_ERR_INVALID_VALUE	SetProperty メソッドに無効なプロパティ値が渡されました。	15
DBSC_ERR_INVALID_PROP_NAME	指定したプロパティ名は無効です。	16
DBSC_ERR_VALUE_TOO_LONG	プロパティ値が長すぎます。プロパティは、DBCS_MAX_PROPERTY_LEN で指定されているバイト数よりも短くしてください。	17
DBSC_ERR_SERVER_SIDE_ERROR	sync のキャンセル中または追加中にサーバー側でエラーが発生しました。 str1 は、サーバーによって返された文字列を指します。この文字列には、エラーの詳細が含まれることがあります。	18

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_ERR_CREATE_PROCESS_FAILED	新しい dbmlsync サーバーを起動できません。	20
DBSC_ERR_READ_FAILED	dbmlsync サーバーからのデータの読み込み中に TCP エラーが発生しました。	21
DBSC_ERR_WRITE_FAILED	dbmlsync サーバーへのデータの送信中に TCP エラーが発生しました。	22
DBSC_ERR_NO_SERVER_RESPONSE	要求されたアクションを完了するために必要な応答をサーバーから受信できませんでした。	23
DBSC_ERR_UID_OR_PWD_TOO_LONG	指定した UID または PWD が長すぎます。	24
DBSC_ERR_UID_OR_PWD_NOT_VALID	指定した UID または PWD が無効です。	25
DBSC_ERR_INVALID_PARAMETER	関数に渡されたパラメーターのいずれかが無効です。	26
DBSC_ERR_WAIT_FAILED	サーバーが停止するのを待っている間にエラーが発生しました。	27
DBSC_ERR_SHUTDOWN_NOT_CALLED	ShutdownServer メソッドを呼び出す前に WaitForServerShutdown メソッドが呼び出されました。	28

メンバー名	説明	値
DBSC_ERR_NO_SYNC_ACK	同期要求がサーバーに送信されましたが、受信確認が受信されませんでした。サーバーが要求を受信したかどうかを判別する方法はありません。 hdl1 は、送信された同期要求のハンドルです。サーバーが要求を受信した場合は、このハンドルを使用すると、GetEvent を使用して取得された同期用イベントを識別できます。	29
DBSC_ERR_ACTIVE_SYNC_NOT_CANCELED	同期がアクティブであるため、サーバーは同期をキャンセルできませんでした。	30
DBSC_ERR_DEAD_SERVER	dbmlsync サーバーで起動時にエラーが発生しました。 サーバーを停止しています。dbmlsync の -o オプションを使用して、ファイルにエラーメッセージのログを記録してください。	31

DBSC_EventType 列挙

同期によって生成されたイベントのタイプを示します。

Visual Basic 構文

```
Public Enum DBSC_EventType
```

C# 構文

```
public enum DBSC_EventType
```

メンバー

メンバー名	説明	値
DBSC_EVENTTYPE_ERROR_MSG	同期によってエラーが生成されました。str1 はエラーのテキストを格納します。	1

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_EVENTTYPE_WARNING_MSG	同期によって警告が生成されました。str1 は警告のテキストを格納します。	2
DBSC_EVENTTYPE_INFO_MSG	同期によって情報メッセージが生成されました。str1 はメッセージのテキストを格納します。	3
DBSC_EVENTTYPE_PROGRESS_INDEX	進行状況バーを更新するための情報を提供します。val1 は、新しい進行状況値を格納します。 完了した割合 (パーセント) を計算するには、val1 を 1000 で割ります。	4
DBSC_EVENTTYPE_PROGRESS_TEXT	進行状況バーに関連付けられているテキストが更新されました。str1 は、新しい値を格納します。	5
DBSC_EVENTTYPE_TITLE	同期ウィンドウ／コントロールのタイトルが変更されました。str1 は新しいタイトルを格納します。	6
DBSC_EVENTTYPE_SYNC_START	同期が開始しました。このイベントに関連付けられている追加情報はありません。	7
DBSC_EVENTTYPE_SYNC_DONE	同期が完了しました。val1 は、同期からの終了コードを格納します。 0 という値は、成功を示します。0 以外の値は、同期が失敗したことを示します。	8

メンバー名	説明	値
DBSC_EVENTTYPE_ML_CONNECT	Mobile Link サーバーへの接続が確立されました。str1 は、使用されている通信プロトコルを示します。str2 は、使用されるネットワークプロトコルオプションを示します。	10
DBSC_EVENTTYPE_UPLOAD_COMMITTED	Mobile Link サーバーは統合データベースへのアップロードを正常にコミットしたことを確認しました。	11
DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	ダウンロードはリモートデータベースで正常にコミットされました。	12

参照

- DBSC_Event 構造体 [Dbmlsync .NET]304 ページ
- DbmlsyncClient.GetEvent メソッド [Dbmlsync .NET]285 ページ

DBSC_GetEventRet 列挙

イベント取得の試行の結果を示します。

Visual Basic 構文

```
Public Enum DBSC_GetEventRet
```

C# 構文

```
public enum DBSC_GetEventRet
```

メンバー

メンバー名	説明	値
DBSC_GETEVENT_OK	イベントが正常に取得されたことを示します。	1
DBSC_GETEVENT_TIMED_OUT	返すことができるイベントがないまま、タイムアウトになったことを示します。	2

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	説明	値
DBSC_GETEVENT_FAILED	エラーが発生したためにイベントが返されなかったことを示します。	3

参照

- DbmlsyncClient.GetEvent メソッド [Dbmlsync .NET]285 ページ

DBSC_ShutdownType 列挙

サーバーを停止する緊急度を示します。

Visual Basic 構文

```
Public Enum DBSC_ShutdownType
```

C# 構文

```
public enum DBSC_ShutdownType
```

メンバー

メンバー名	説明	値
DBSC_SHUTDOWN_ON_EMPTY_QUEUE	サーバーが未処理の同期要求を完了してから停止する必要があることを示します。 サーバーは、停止要求を受信すると、それよりも後の同期要求を受け入れません。	1
DBSC_SHUTDOWN_CLEANLY	サーバーができるだけ早く正常に停止する必要があることを示します。 未処理の同期要求は実行されません。実行中の同期は中断されることがあります。	2

参照

- DbmlsyncClient.ShutdownServer メソッド [Dbmlsync .NET]290 ページ

DBSC_StartType 列挙

dbmlsync サーバーを起動しようとしているときに実行されたアクションを示します。

Visual Basic 構文

```
Public Enum DBSC_StartType
```

C# 構文

```
public enum DBSC_StartType
```

メンバー

メンバー名	説明	値
DBSC_SS_STARTED	新しい dbmlsync サーバーが起動されたことを示します。	1
DBSC_SS_ALREADY_RUNNING	既存の dbmlsync サーバーが見つかったため、新しいサーバーが起動されなかったことを示します。	2

参照

- DbmlsyncClient.StartServer メソッド [Dbmlsync .NET]291 ページ

DBSC_ErrorInfo 構造体

以前のメソッド呼び出しの失敗に関する情報が含まれています。

Visual Basic 構文

```
Structure DBSC_ErrorInfo
```

C# 構文

```
public typedef struct DBSC_ErrorInfo
```

メンバー

メンバー名	タイプ	説明
hdl1	UInt32	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

メンバー名	タイプ	説明
str1	String	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
str2	String	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
type	DBSC_ErrorType	失敗の理由を示す値が含まれています。 サポートされる値は DBSC_ErrorType 列挙にリストされます。
val1	Int32	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
val2	Int32	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

備考

str1、str2、val1、val2、hdl1 には失敗に関する詳細が含まれ、その意味はエラータイプによって異なります。次のエラータイプはこの構造体のフィールドを使用して詳細を格納します。

- DBSC_ERR_CONNECTION_REJECTED
- DBSC_ERR_SERVER_SIDE_ERROR
- DBSC_ERR_NO_SYNC_ACK

参照

- DBSC_ErrorType 列挙 [Dbmlsync .NET]295 ページ

DBSC_Event 構造体

同期によって生成されたイベントに関する情報が含まれています。

Visual Basic 構文

```
Structure DBSC_Event
```

C# 構文

```
public typedef struct DBSC_Event
```

メンバー

メンバー名	タイプ	説明
hdl	UInt32	イベントを生成した同期を示します。 この値は、Sync メソッドによって返される値と一致します。
str1	String	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
str2	String	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
type	DBSC_EventType	レポートされるイベントのタイプを示します。
val1	Int32	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。
val2	Int32	失敗に関する詳細が含まれています。 この情報の意味は type 変数の値によって異なります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

参照

- DBSC_EventType 列挙 [Dbmlsync .NET]298 ページ

dbmlsync の DBTools インターフェイス

データベースツール (DBTools) は、同期を含むデータベース管理をアプリケーションに統合するために使用できるライブラリです。データベース管理ユーティリティはすべて、DBTools によって構築されます。

「データベースツールインターフェイス (DBTools)」『SQL Anywhere サーバー プログラミング』を参照してください。

注意
アプリケーションに同期を統合する場合、Dbmlsync API インターフェイスを使用することをおすすめします。DBTools インターフェイスと非常に良く似た機能があり、使い方も簡単です。

「Dbmlsync API」98 ページを参照してください。

dbmlsync 用の DBTools インターフェイスを使用することで、Mobile Link 同期クライアントアプリケーションに同期機能を統合できます。たとえば、このインターフェイスを使用して、同期を起動し、カスタムユーザーインターフェイスに dbmlsync の出力メッセージを表示できます。

dbmlsync 用の DBTools インターフェイスは、Mobile Link 同期クライアントを設定および実行する次の要素から構成されます。

- **a_sync_db 構造体** この構造体は、dbmlsync コマンドラインオプションに対応する設定を保持します。これらの設定によって同期をカスタマイズできます。この構造体には、同期と進行状況情報を受け取るコールバック関数へのポインターも含まれます。

  a_sync_db 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』を参照してください。

- **a_syncpub 構造体** この構造体は、パブリケーションまたはサブスクリプションの情報を保持します。同期用のパブリケーションまたはサブスクリプションのリンクリストを指定できます。

  a_syncpub 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』を参照してください。

- **DBSynchronizeLog 関数** この関数は同期処理を開始します。この関数のパラメーターは、a_sync_db インスタンスへのポインターだけです。

  DBSynchronizeLog メソッド [データベースツール]『SQL Anywhere サーバー プログラミング』を参照してください。

dbmlsync の DBTools インターフェイスの設定

この項では、dbmlsync の DBTools インターフェイスの基本的な使用手順を示します。

DBTools ライブラリの詳細については、「データベースツールインターフェイス (DBTools)」『SQL Anywhere サーバー プログラミング』を参照してください。

ご使用の開発環境でのインポートライブラリの使用の詳細については、「DBTools インポートライブラリ」『SQL Anywhere サーバー プログラミング』を参照してください。

♦ C または C++ で作成された DBTools インターフェイスを使用した dbmlsync の設定と起動

1. DBTools ヘッダーファイルをインクルードします。

   DBTools ヘッダーファイル *dbtools.h* は、DBTools ライブラリのエントリポイントをリストし、必要なデータ型を定義します。

   ```
   #include "dbtools.h"
   ```

2. DBTools インターフェイスを起動します。

   ●a_dbtools_info 構造体の宣言と初期化を行います。

   ```
   a_dbtools_info   info;
   short ret;
   ...
   // clear a_dbtools_info fields
   memset( &info, 0, sizeof( info ) );
   info.errorrtn = dbsyncErrorCallBack;
   ```

   dbsyncErrorCallBack はエラーメッセージを処理する関数であり、この作業の手順 4 で定義します。

   ●DBToolsInit 関数を使用して DBTools を初期化します。

   ```
   ret = DBToolsInit( &info );
   if( ret != 0 ) {
    printf("dbtools initialization failure ¥n");
   }
   ```

   DBTools の初期化の詳細については、次の項を参照してください。

   ○ 「DBTools ライブラリの初期化とファイナライズ」『SQL Anywhere サーバー プログラミング』
   ○ a_dbtools_info 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』
   ○ DBToolsInit メソッド [データベースツール]『SQL Anywhere サーバー プログラミング』

3. a_sync_db 構造体を初期化します。

   ●a_sync_db インスタンスを宣言します。たとえば、次のように dbsync_info というインスタンスを宣言します。

   ```
   a_sync_db dbsync_info;
   ```

   ●a_sync_db 構造体のフィールドをクリアします。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
memset( &dbsync_info, 0, sizeof( dbsync_info ) );
```

●必須の a_sync_db のフィールドを設定します。

```
dbsync_info.version = DB_TOOLS_VERSION_NUMBER;
dbsync_info.output_to_mobile_link = 1;
dbsync_info.default_window_title
  = "dbmlsync dbtools sample";
```

●データベース接続文字列を設定します。

```
dbsync_info.connectparms = "UID=DBA;PWD=sql";
```

データベース接続パラメーターの詳細については、「-c dbmlsync オプション」111 ページを参照してください。

●同期をカスタマイズするための他の a_sync_db のフィールドを設定します。

ほとんどのフィールドは、dbmlsync コマンドラインオプションに対応しています。この対応の詳細については、*dbtools.h* を参照してください。

次の例では、冗長オペレーションが指定されています。

```
dbsync_info.verbose_upload = 1;
dbsync_info.verbose_option_info = 1;
dbsync_info.verbose_row_data = 1;
dbsync_info.verbose_row_cnts = 1;
```

a_sync_db のフィールドの詳細については、a_sync_db 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』を参照してください。

4. 同期中にフィードバックを受け取るコールバック関数を作成し、これらの関数を適切な a_sync_db のフィールドに割り当てます。

次の関数は、標準出力ストリームを使用して dbmlsync のエラー、ログ、進行状況情報を表示します。

DBTools のコールバック関数の詳細については、「コールバック関数」『SQL Anywhere サーバー プログラミング』を参照してください。

●たとえば、生成されたエラーメッセージを処理する dbsyncErrorCallBack という関数を作成します。

```
extern short _callback dbsyncErrorCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Error Msg    %s¥n", str );
    }
    return 0;
}
```

●たとえば、生成された警告メッセージを処理する dbsyncWarningCallBack という関数を作成します。

```
extern short _callback dbsyncWarningCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Warning Msg  %s¥n", str );
    }
```

```
    return 0;
}
```

●たとえば、冗長情報メッセージを受け取る dbsyncLogCallBack という関数を作成します。この情報メッセージは、ウィンドウに表示する代わりにファイルに記録できます。

```
extern short _callback dbsyncLogCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Log Msg      %s¥n", str );
    }
    return 0;
}
```

●たとえば、同期中に生成された情報メッセージを受け取る dbsyncMsgCallBack という関数を作成します。

```
extern short _callback dbsyncMsgCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Display Msg  %s¥n", str );
    }
    return 0;
}
```

●たとえば、進行状況テキストを受け取る dbsyncProgressMessageCallBack という関数を作成します。dbmlsync ユーティリティでは、このテキストは進行状況バーの真上に表示されます。

```
extern short _callback dbsyncProgressMessageCallBack(
 char *str )
{
    if( str != NULL ) {
        printf( "ProgressText %s¥n", str );
    }
    return 0;
}
```

●たとえば、進行状況インジケーターまたは進行状況バーを更新するために情報を受け取る dbsyncProgressIndexCallBack という関数を作成します。この関数は、次の 2 つのパラメーターを受け取ります。

○ **index**　同期の現在の進行状況を表す整数。

○ **max**　進行状況の最大値。この値が 0 である場合、最大値はこのイベントが最後に呼び出されてから変更されていません。

```
extern short _callback dbsyncProgressIndexCallBack
(a_sql_uint32 index, a_sql_uint32 max )
{
    printf( "ProgressIndex    Index %d Max: %d¥n",
        index, max );
    return 0;
}
```

次に、このコールバックへの呼び出しの一般的な順序を示します。

```
// example calling sequence
dbsyncProgressIndexCallBack( 0, 100 );
dbsyncProgressIndexCallBack( 25, 0 );
dbsyncProgressIndexCallBack( 50, 0 );
dbsyncProgressIndexCallBack( 75, 0 );
dbsyncProgressIndexCallBack( 100, 0 );
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

この順序では、0% 完了、25% 完了、50% 完了、75% 完了、100% 完了に設定された進行状況バーが表示されます。

●たとえば、ステータス情報を受け取る dbsyncWindowTitleCallBack という関数を作成します。dbmlsync ユーティリティでは、この情報はタイトルバーに表示されます。

```
extern short _callback dbsyncWindowTitleCallBack(
 char *title )
{
   printf( "Window Title     %s¥n", title );
   return 0;
}
```

●dbsyncMsgQueueCallBack 関数は、遅延またはスリープが必要な場合に呼び出します。この関数は、*dllapi.h* に定義されている次の値のいずれかを返す必要があります。

○ **MSGQ_SLEEP_THROUGH**　要求したミリ秒の間ルーチンがスリープしたことを示します。

○ **MSGQ_SHUTDOWN_REQUESTED**　できるだけ早く同期を終了したいことを示します。

○ **MSGQ_SYNC_REQUESTED**　要求したミリ秒に達しないうちにルーチンがスリープ状態を終了したことと、同期が現在進行中でない場合はただちに次の同期をとり始めることを示します。

```
extern short _callback dbsyncMsgQueueCallBack(
  a_sql_uint32 sleep_period_in_milliseconds )
{

 printf( "Sleep %d ms¥n", sleep_period_in_milliseconds );
 Sleep( sleep_period_in_milliseconds );
 return MSGQ_SLEEP_THROUGH;
}
```

●適切な a_sync_db 同期構造体のフィールドにコールバック関数のポインターを割り当てます。

```
// set call back functions
dbsync_info.errorrtn    = dbsyncErrorCallBack;
dbsync_info.warningrtn  = dbsyncWarningCallBack;
dbsync_info.logrtn      = dbsyncLogCallBack;
dbsync_info.msgrtn      = dbsyncMsgCallBack;
dbsync_info.msgqueuertn = dbsyncMsgQueueCallBack;
dbsync_info.progress_index_rtn
    = dbsyncProgressIndexCallBack;
dbsync_info.progress_msg_rtn
    = dbsyncProgressMessageCallBack;
dbsync_info.set_window_title_rtn
    = dbsyncWindowTitleCallBack;
```

5. どのサブスクリプションを同期するかを指定する a_syncpub 構造体のリンクリストを作成します。

   リンクリスト内の各ノードは、dbmlsync コマンドライン上の -s オプションの 1 つのインスタンスに対応します。

   ●a_syncpub インスタンスを宣言します。たとえば、これを publication_info とします。

   ```
   a_syncpub publication_info;
   ```

- a_syncpub フィールドを初期化し、同期するサブスクリプションを指定します。

  たとえば、単一の同期セッションで template_p1 サブスクリプションと template_p2 サブスクリプションをまとめて同期するには、次のように指定します。

  ```
  publication_info.next = NULL; // linked list terminates
  publication_info.subscription = "template_p1,template_p2";
  publication_info.ext_opt  = "dir=c:¥¥logs";
  publication_info.alloced_by_dbsync = 0;
  publication_info.pub_name = NULL;
  ```

  これは、dbmlsync コマンドラインで -s template_p1,template_p2 と指定するのと同じです。

  ext_opt フィールドを使用して拡張オプションを指定すると、dbmlsync -eu オプションと同じ機能が提供されます。

- a_sync_db インスタンスの upload_defs フィールドにパブリケーション構造体を割り当てます。

  ```
  dbsync_info.upload_defs   = &publication_info;
  ```

  a_syncpub 構造体のリンクリストを作成できます。リンクリスト内の各 a_syncpub インスタンスは、dbmlsync コマンドライン上の -n または -s オプションの 1 つの定義に相当します。

6. DBSynchronizeLog 関数を使用して dbmlsync を実行します。

   次に示すコード内の sync_ret_val には、成功した場合は戻り値 0、失敗した場合は 0 以外の値が格納されます。

   ```
   short sync_ret_val;
   printf("Running dbmlsync using dbtools interface...¥n");
   sync_ret_val = DBSynchronizeLog(&dbsync_info);
   printf("¥n Done... synchronization return value is: %I ¥n", sync_ret_val);
   ```

   手順 6 は、同じパラメーター値または異なるパラメーター値を指定して複数回繰り返すことができます。

7. DBTools インターフェイスをシャットダウンします。

   DBToolsFini 関数は、DBTools リソースを解放します。

   ```
   DBToolsFini( &info );
   ```

参照

- 「-eu dbmlsync オプション」 118 ページ
- 「-s dbmlsync オプション」 128 ページ
- 「-n dbmlsync オプション (旧式)」 121 ページ
- a_syncpub 構造体 [データベースツール]『SQL Anywhere サーバー プログラミング』
- DBToolsFini メソッド [データベースツール]『SQL Anywhere サーバー プログラミング』

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

スクリプト化されたアップロード

スクリプト化されたアップロードは、SQL Anywhere リモートデータベースを使用する Mobile Link アプリケーションだけに適用されます。

警告
スクリプト化されたアップロードを実装した場合、dbmlsync では、アップロードするデータの判別にトランザクションログは使用されません。その結果、スクリプトがすべての変更を取得しなかった場合は、リモートデータベース上のデータが失われることがあります。このため、ほとんどのアプリケーションの同期方法としては、ログベースの同期をおすすめします。

ほとんどの Mobile Link アプリケーションでは、データベースのトランザクションログによってアップロードが判別されるので、最後のアップロード以降にリモートデータベースに加えられた変更が同期されます。これは、ほとんどのアプリケーションでは適切な設計であり、リモートデータベース上のデータが失われないようにしています。

しかし、トランザクションログを無視して、アップロードを定義する必要がある場合があります。スクリプト化されたアップロードを使用すると、アップロードするデータを正確に定義できます。スクリプト化されたアップロードを使用する場合は、リモートデータベースのトランザクションログを保持する必要はありません。トランザクションログはかなりの領域を占有するため、小型デバイスでは考慮する必要があります。ただし、トランザクションログは、データベースのバックアップとリカバリには非常に重要であり、データベースのパフォーマンスの向上に役立ちます。

スクリプト化されたアップロードを実装するには、特殊なパブリケーションを作成し、そのパブリケーションで、作成したストアドプロシージャーの名前を指定します。ストアドプロシージャーは、統合データベースで挿入、更新、または削除するローが含まれる結果セットを返すことによって、アップロードを定義します。

「注意：」スクリプト化されたアップロードとアップロードスクリプトを混同しないように注意してください。アップロードスクリプトは、アップロードによりどのような処理を行うかを Mobile Link サーバーに指示するための、統合データベース上の Mobile Link イベントスクリプトです。スクリプト化されたアップロードを使用する場合は、統合データベースにアップロードを適用するためにアップロードスクリプトを作成し、ダウンロードするデータを指定するためにダウンロードスクリプトを作成する必要があります。

シナリオ

スクリプト化されたアップロードが役立つシナリオを以下に示します。

- リモートデータベースは記憶領域が限定されているデバイスで実行中で、トランザクションログを格納するのに十分な領域がない場合
- すべてのリモートデータベースからすべてのデータをアップロードして、新しい統合データベースを作成する場合
- 統合データベースにアップロードされる変更を特定するカスタム論理を作成する場合

警告

スクリプト化されたアップロードを実装する前に、この項全体をお読みください。次の点に特に注意してください。

- スクリプト化されたアップロードを正しく設定しないと、データが失われます。
- スクリプト化されたアップロードを実装する場合は、dbmlsync が通常処理するデータを維持または参照する必要があります。これには、データの更新前と更新後のイメージや、同期の進行状況が含まれます。
- スクリプト化されたアップロードを介して同期している間、リモートデータベース上のテーブルをロックする必要があります。ログベースの同期を行う場合は、ロックする必要はありません。
- スクリプト化されたアップロードを使用してトランザクション単位のアップロードを実装するのは非常に困難です。

スクリプト化されたアップロードの設定

次の手順では、Mobile Link 同期が設定済みであることを前提に、スクリプト化されたアップロードの設定に必要なタスクの概要について説明します。

警告
スクリプト化されたアップロードを使用するときは、dbmlsync の LockTables 拡張オプションにデフォルト設定を使用することをおすすめします。

スクリプト化されたアップロードの多くの問題は、LockTables にデフォルト設定を使用することで防ぐことができます。この設定により、dbmlsync は、すべての同期テーブルのロックを取得してから、アップロードを構築できます。これにより、スクリプトがアップロードを構築中に、他の接続が同期テーブルを変更するのを防ぐことができます。また、この設定を行うと、スクリプトがアップロードを構築中に、同期テーブルに影響するコミットされていないトランザクションをなくすことができます。

♦ スクリプト化されたアップロードの設定の概要

1. アップロードするローを識別するストアドプロシージャーを作成します。テーブルごとに 3 つのストアドプロシージャー (アップロード、挿入、削除用に 1 つずつ) を定義できます。

   「スクリプト化されたアップロードのストアドプロシージャー」319 ページを参照してください。

2. WITH SCRIPTED UPLOAD というキーワードが含まれ、ストアドプロシージャーの名前を指定するパブリケーションを作成します。

   「スクリプト化されたアップロードのパブリケーション」323 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

クイックスタートのためのその他の資料

- 「チュートリアル：スクリプト化されたアップロードの使用」

スクリプト化されたアップロードの設計上の考慮事項

1 ローあたり 1 つの操作

アップロードには、1 つのローに対して複数の操作 (挿入、更新、または削除) を含めることはできません。ただし、複数の操作を 1 つのアップロード操作にまとめることはできます。たとえば、ローを挿入してから更新する場合は、2 つの操作を、最後の値を 1 回挿入する操作に置き換えることができます。

操作の順序

アップロードを統合データベースに適用すると、挿入と更新操作の後に削除操作が適用されます。特定のテーブル内での操作順序について他の想定をすることはできません。

競合の解決

同期と同期の間に複数のデータベースでローが更新されると、競合が発生します。アップロード内の各更新操作には、更新中のローの更新前イメージが含まれているので、Mobile Link サーバーは競合を検出することができます。更新前イメージは、最後にアップロードまたはダウンロードに成功したローの全カラムの値です。アップロードが提供されたときに、更新前イメージが統合データベースの値と一致しないと、Mobile Link サーバーは競合を検出します。

アプリケーションで競合を検出する必要があり、スクリプト化されたアップロードを使用している場合は、リモートデータベースで、最後にアップロードまたはダウンロードに成功した各ローの値を追跡する必要があります。これにより、正しい更新前イメージをアップロードできます。

更新前イメージのデータを維持する 1 つの方法は、同期テーブルと同一の更新前イメージテーブルを作成することです。次に、更新を実行するたびに更新前イメージテーブルを移植するトリガーを同期テーブルに作成します。アップロードに成功したら、更新前イメージテーブルのローを削除できます。

競合解決の実装の例については、「チュートリアル：スクリプト化されたアップロードの使用」 324 ページを参照してください。

競合を処理しない

競合の検出を処理する必要がない場合は、更新前イメージを追跡しなければ、アプリケーションを大幅に簡素化できます。代わりに、挿入操作として更新をアップロードします。次に、ローが存在しない場合は挿入し、存在する場合はローを更新する upload_insert スクリプトを統合データベースに作成します。SQL Anywhere 統合データベースを使用している場合、この操作は、upload_insert スクリプトの INSERT 文にある ON EXISTING 句を使用して実行できます。

「INSERT 文」『SQL Anywhere サーバー SQL リファレンス』を参照してください。

競合を処理せず、複数のリモートデータベースで同じローが変更されると、最後に同期したリモートデータベースによって、それまでの変更が上書きされます。

強制的な競合解決の処理

削除操作では、アップロードされたローのプライマリキーが正しいことが重要です。しかし、非プライマリキーカラムの値が、統合データベースの値と一致しているかどうかは問題ではありません。非プライマリキーカラムが重要なのは、Mobile Link サーバーで強制的な競合モードが使用されるときだけです。この場合、カラムのすべての値が、統合データベースの upload_old_row_insert スクリプトに渡されます。このスクリプトを実装した方法によっては、非プライマリキーカラムを正しい値にする必要があります。

「強制的な競合解決 (非推奨)」『Mobile Link サーバー管理』を参照してください。

ロッキング

スクリプト化されたアップロードの多くの問題は、dbmlsync 拡張オプション LockTables にデフォルト設定を使用することで防ぐことができます。この設定により、dbmlsync は、すべての同期テーブルの排他ロックを取得してから、アップロードを構築できます。これにより、スクリプトがアップロードを構築中に、他の接続が同期テーブルを変更するのを防ぐことができます。また、この設定を行うと、スクリプトがアップロードを構築中に、同期テーブルに影響するコミットされていないトランザクションをなくすことができます。

テーブルのロックをオフにする必要がある場合は、「テーブルをロックしない場合のスクリプト化されたアップロード」316 ページを参照してください。

冗長なアップロード

通常、リモートデータベースで各操作をアップロードするのは、一度だけです。この操作を支援するため、Mobile Link は各サブスクリプションの進行状況値を維持します。進行状況値は、デフォルトで、dbmlsync が正常な最終アップロードを構築し始めた時間です。この進行状況値は、sp_hook_dbmlsync_set_upload_end_progress フックを使用して異なる値で上書きできます。

「sp_hook_dbmlsync_set_upload_end_progress」242 ページを参照してください。

アップロードプロシージャーの 1 つが呼び出されるたびに、#hook_dict テーブルを介して値が渡されます。渡される値としては、'start progress' や 'end progress' があります。これらの値は、リモートデータベースへの変更が構築中のアップロードに含まれるようにする時間を定義します。'start progress' 前に発生した操作は、すでにアップロードされています。'end progress' 後に発生する操作は、次の同期中にアップロードされます。

不明なアップロードステータス

スクリプト化されたアップロードの実装でよくある間違いは、sp_hook_dbmlsync_upload_end または sp_hook_dbmlsync_end フックで、アップロードが正常に統合データベースに適用されたかどうかを判断することによってストアドプロシージャーを作成してしまうことです。この方法は信頼性に欠けます。

たとえば、次の例では、各ローの 1 ビットを使用して挿入を処理し、ローをアップロードする必要があるかどうかを追跡しようとしています。ビットは、ローが挿入されると設定され、アップロードが正常にコミットされると sp_hook_dbmlsync_upload_end フックで解除されます。

```
//
// DO NOT DO THIS!
//
CREATE TABLE t1 (
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
  pk          integer primary key,
  val         varchar( 256 ),
  to_upload   bit DEFAULT 1
);

CREATE PROCEDURE t1_ins()
RESULT( pk integer, val varchar(256) )
BEGIN
   SELECT pk, val
   FROM t1
   WHERE to_upload = 1;
END;

CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
BEGIN
   DECLARE     upload_status   varchar(256);

   SELECT value
   INTO upload_status
   FROM #hook_dict
   WHERE name = 'upload status';

   if upload_status = 'committed' THEN
      UPDATE t1 SET to_upload = 0;
   END IF
END;

CREATE PUBLICATION p1 WITH SCRIPTED UPLOAD (
    TABLE t1 USING ( PROCEDURE t1_ins FOR UPLOAD INSERT )
);
```

この方法は、ほとんどの場合に機能します。ハードウェアまたはソフトウェアの障害が発生し、アップロードが送信された後、サーバーで受信確認される前に、dbmlsync が停止されると、失敗します。この場合、アップロードは統合データベースに適用されますが、sp_hook_dbmlsync_upload_end フックは呼び出されず、to_upload ビットは解除されません。その結果、次回の同期では、アップロード済みのローに挿入がアップロードされます。この場合、通常は統合データベースで重複プライマリキーエラーが発生し、同期が失敗します。

そのほかには、Mobile Link サーバーとの通信が、アップロードが送信された後、受信確認される前に失われた場合に、問題が発生します。この場合、dbmlsync は、アップロードが正常に適用されたかどうかを確認できません。dbmlsync は sp_hook_dbmlsync_upload_end フックを呼び出して、アップロードステータスを不明に設定します。フックが作成されると、to_upload ビットが解除されるのを防ぐことができます。サーバーによってアップロードが適用されなかった場合、問題は発生しません。ただし、アップロードが適用されると、前述と同じ問題が発生します。いずれの場合も、問題を手動で解決するまで、影響を受けたリモートデータベースを再び同期することはできません。

ダウンロード時のデータ損失の防止

スクリプト化されたアップロードを使用すると、リモートデータベースでアップロードが必要なデータが、統合データベースからダウンロードされたデータで上書きされる可能性があります。この場合、リモートデータベースでの変更内容が失われます。このデータ損失を防止するには、構築するアップロードごとに sp_hook_dbmlsync_set_upload_end_progress フックを呼び出す前に、リモートデータベースでコミットされた変更内容がすべてアップロードプロシージャーに含まれている必要があります。

この規則に従わなかった場合にデータがどのように失われるかを次の例に示します。

時刻	
1:05:00	統合データベースとリモートデータベースの両方にあるロー R が、リモートデータベースで新しい値 R1 に更新され、変更がコミットされます。
1:06:00	統合データベースでロー R が新しい値 R2 に更新され、変更がコミットされます。
1:07:00	同期が発生します。アップロードスクリプトは、1:00:00 より前にコミットされた操作だけが含まれるように記述されています。アップロードの構築前に発生したすべての操作がアップロードされないので、これは規則に従っていません。ロー R への変更は、1:00:00 を過ぎてから発生したのでアップロードに含まれません。サーバーから受信されるダウンロードにはロー R2 が含まれます。ダウンロードが適用されると、リモートデータベースのロー R1 がロー R2 に置き換わります。リモートデータベースでの更新内容は失われます。

テーブルをロックしない場合のスクリプト化されたアップロード

デフォルトでは、dbmlsync によって同期対象のテーブルがロックされてからアップロードスクリプトが呼び出され、このロックはダウンロードがコミットされるまで保持されます。拡張オプション LockTables をオフに設定すると、テーブルのロックを無効にできます。

可能な場合は、テーブルをロックするデフォルトの動作を使用することをおすすめします。テーブルをロックしないでスクリプト化されたアップロードを行うと、考慮する必要がある問題が増え、実行可能な正しい解決法を作成するのが難しくなります。これは、データベースの同時実行性と同期の概念をよく理解している上級ユーザーだけが行ってください。

テーブルをロックしない場合の独立性レベルの使用

テーブルのロックがオフのときは、アップロードのストアドプロシージャーを実行する独立性レベルが非常に重要です。独立性レベルによって、コミットされていないトランザクションの処理方法が決まるからです。テーブルのロックがオンのときは、テーブルのロックによって、アップロードの構築時に、同期対象のテーブルに対するコミットされていない変更が防止されるので、独立性レベルは問題ではありません。

アップロードのストアドプロシージャーで独立性レベルを明示的に変更しなかった場合、ストアドプロシージャーは、dbmlsync のコマンドラインで指定したデータベースユーザーのデフォルトの独立性レベルで実行されます。

データベースのデフォルトの独立性レベルは 0 ですが、テーブルをロックしないでスクリプト化されたアップロードを行うときは、アップロードのプロシージャーを独立性レベル 0 で実行しないことをおすすめします。テーブルをロックしないでスクリプト化されたアップロードを実行するときに独立性レベル 0 を使用すると、コミットされていない変更がアップロードされ、次の問題が発生する可能性があります。

- コミットされていない変更がロールバックされると、間違ったデータが統合データベースに送信されます。
- コミットされていないトランザクションが完了していない場合は、トランザクションの一部だけがアップロードされ、統合データベースの整合性が失われる可能性があります。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

使用できる独立性レベルは 1、2、3、またはスナップショットです。これらの独立性レベルでは、コミットされていないトランザクションはアップロードされません。

独立性レベル 1、2、または 3 を使用した場合、テーブルに対してコミットされていない変更があると、アップロードのストアドプロシージャーがブロックする可能性があります。アップロードのストアドプロシージャーは、dbmlsync が Mobile Link サーバーに接続している間に呼び出されるので、サーバー接続が占有される可能性があります。独立性レベル 1 を使用した場合、SELECT 文で READPAST テーブルヒント句を使用することでブロックを防止できる場合があります。

スナップショットアイソレーションを使用すると、ブロックと、コミットされていない変更の読み込みの両方を防止できます。

コミットされていない変更の損失

テーブルをロックしない場合は、同期の発生時にコミットされていない操作を処理するメカニズムが必要です。なぜこれが問題であるかを理解するために、次に例を示します。

スクリプト化されたアップロードでテーブルを同期するとします。わかりやすくするために、挿入だけをアップロードするとします。テーブルには insert_time というカラムがあります。このカラムは、各ローが挿入された時刻を示すタイムスタンプです。

各アップロードは、テーブル内で insert_time が最後の正常なアップロードと、現在のアップロードの構築を開始した時刻 (sp_hook_dbmlsync_set_upload_end_progress フックが呼び出された時刻) の間にあるすべてのコミットされたローを選択して構築します。次の処理が行われるとします。

時刻	
1:00:00	正常な同期が発生します。
1:04:00	ロー R がテーブルに挿入されますが、コミットされません。R の insert_time カラムが 1:04:00 に設定されます。
1:05:00	同期が発生します。insert_time が 1:00:00 と 1:05:00 の間にあるローがアップロードされます。ロー R はコミットされていないのでアップロードされません。同期の進行状況が 1:05:00 に設定されます。
1:07:00	1:04:00 に挿入されたローがコミットされます。R の insert_time カラムは 1:04:00 のままです。
1:10:00	同期が発生します。insert_time が 1:05:00 と 1:10:00 の間にあるローがアップロードされます。ロー R は、insert_time がこの範囲内にないのでアップロードされません。つまり、ロー R はアップロードされることはありません。

一般に、同期の前に発生し、同期の後にコミットされた操作は、このように失われる可能性があります。

コミットされていないトランザクションの処理

コミットされていないトランザクションを処理する方法として最も簡単なのは、sp_hook_dbmlsync_set_upload_end_progress フックを使用して、各同期の終了進行状況を、フックの呼び出し時にコミットされていない最も古いトランザクションの開始時刻に設定する方法です。この時刻は、次のように sa_transactions システムプロシージャーを使用して確認できます。

```
SELECT min( start_time )
FROM sa_transactions()
```

この場合、アップロードのストアドプロシージャーでは、sa_transactions を使用して sp_hook_dbmlsync_set_upload_end_progress フックで計算され、#hook_dict テーブルを使用して渡される終了進行状況を無視する必要があります。ストアドプロシージャーでは、単に開始進行状況後に発生したコミットされた操作をすべてアップロードします。このようにすると、変更内容をアップロードする必要があるローがダウンロード内容で上書きされません。また、コミットされていないトランザクションがあっても、操作が適時にアップロードされます。

この解決法では、操作は失われませんが、一部の操作が複数回アップロードされる可能性があります。サーバー側のスクリプトは、複数回アップロードされる操作を処理するように記述する必要があります。この設定でローが複数回アップロードされる例を次に示します。

時刻	
1:00:00	正常な同期が発生します。
2:00:00	ロー R1 が挿入されますが、コミットされません。
2:10:00	ロー R2 が挿入され、コミットされます。
3:00:00	同期が発生します。1:00 と 3:00 の間に発生した操作がアップロードされます。ロー R2 はアップロードされます。コミットされていない最も古いトランザクションの開始時刻が 2:00 なので、進行状況は 2:00 に設定されます。
4:00:00	ロー R1 がコミットされます。
5:00:00	同期が発生します。2:00 と 5:00 の間に発生した操作がアップロードされ、進行状況が 5:00 に設定されます。アップロードには R1 と R2 の両方のローが含まれます。いずれもタイムスタンプがアップロード範囲内にあるからです。したがって、R2 は 2 回アップロードされます。

統合データベースが SQL Anywhere の場合は、統合データベースの upload_insert スクリプトで INSERT...ON EXISTING UPDATE 文を使用することで、複数回アップロードされる挿入操作を処理できます。

その他の統合データベースについては、upload_insert スクリプトから呼び出すストアドプロシージャーで似たような論理を実装できます。挿入するローとプライマリキーが同じであるローが統合データベースにあるかどうかを確認するだけです。ローがある場合は更新し、ない場合は新しいローを挿入します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

サーバー側に競合を検出または解決する論理がある場合は、削除操作と更新操作が複数回アップロードされることが問題になります。サーバー側で競合の検出と解決のスクリプトを記述する場合は、スクリプトで複数回のアップロードを処理できる必要があります。

統合データベースでプライマリキーの値を再利用できる場合は、削除操作が複数回アップロードされることが重大な問題になります。次の一連のイベントを考えてみます。

1. プライマリキーが 100 のロー R がリモートデータベースに挿入され、統合データベースにアップロードされます。
2. ロー R がリモートデータベースで削除され、削除操作がアップロードされます。
3. プライマリキーが 100 の新しいロー R' が統合データベースに挿入されます。
4. 手順 2 のロー R の削除操作がリモートデータベースからもう一度アップロードされます。これにより、ロー R' が統合データベースから間違って削除される可能性があります。

参照

- 「sa_transactions システムプロシージャー」『SQL Anywhere サーバー SQL リファレンス』
- 「独立性レベルの設定」『SQL Anywhere サーバー SQL の使用法』

スクリプト化されたアップロードのストアドプロシージャー

スクリプト化されたアップロードを実装するには、統合データベースで挿入、更新、または削除するローが含まれる結果セットを返すことによってアップロードを定義する、ストアドプロシージャーを作成します。

ストアドプロシージャーが呼び出されると、#hook_dict というテンポラリテーブルが作成されます。このテーブルには、name と value という 2 つのカラムがあります。このテーブルを使用すると、名前と値の組み合わせをストアドプロシージャーに渡すことができます。ストアドプロシージャーは、このテーブルから有用な情報を取得することができます。

次の名前と値の組み合わせが定義されています。

名前	値	説明
start progress	タイムスタンプ文字列	リモートデータベースに対するすべての変更がアップロードされるまでの時間。アップロードが反映されるのは、この時間以降に発生した操作だけです。
raw start progress	64 ビット符号なし整数	符号なし整数で示された開始進行状況
end progress	タイムスタンプ文字列	アップロード期間の終了。アップロードが反映されるのは、この時間の前に発生した操作だけです。

名前	値	説明
raw end progress	64 ビット符号なし整数	符号なし整数で示された終了進行状況
generating download exclusion list	true|false	同期がダウンロード専用またはファイルベースの場合は true。この場合、アップロードは送信されず、ダウンロードがスクリプト化されたアップロードのストアドプロシージャーで選択したローに影響しない場合、ダウンロードは適用されません (これにより、アップロードの必要がある、リモートデータベースで追加された変更が、ダウンロードによって上書きされないようにすることができます)。
subscription_*n*	サブスクリプション名	同期されているサブスクリプションの名前 (*n* は整数)。これは、同期される各サブスクリプションの subscription_*n* エントリです。*n* の番号は 0 から始まります。
publication_*n*	パブリケーション名	廃止予定。代わりに subscription_*n* を使用します。同期されているパブリケーション (*n* は整数)。*n* の番号は 0 から始まります。
script version	バージョン名	同期に使用される Mobile Link スクリプトバージョン
MobiLink user	Mobile Link ユーザー名	同期対象となる Mobile Link ユーザー

参照

- 「#hook_dict テーブル」198 ページ

スクリプト化されたアップロードのカスタム進行状況値

スクリプト化されたアップロードプロシージャーに渡された開始進行状況値と終了進行状況値は、デフォルトで、タイムスタンプを表します。終了進行状況値は、デフォルトで、dbmlsync がアップロードを構築し始めた時間です。同期の開始進行状況値は、その同期のサブスクリプションを最後に正常にアップロードしたときに使用した終了進行状況値です。ほとんどの実装には、このデフォルトの動作が適しています。

まれに、別の動作が必要な場合は、sp_hook_dbmlsync_set_upload_end_progress フックが使用されます。このフックを使用すると、アップロードに使用する終了進行状況値を設定できます。終了

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

進行状況値には、開始進行状況値より大きい値を設定する必要があります。開始進行状況値を変更することはできません。

sp_hook_dbmlsync_set_upload_end_progress フックでは、終了進行状況値を TIMESTAMP または UNSIGNED INTEGER として指定できます。アップロードストアドプロシージャーに対しては、いずれの形式の値も使用できます。便宜上、sa_convert_ml_progress_to_timestamp と sa_convert_timestamp_to_ml_progress 関数を使用して、2 形式間で進行状況値を変換することができます。

参照

- 「sp_hook_dbmlsync_set_upload_end_progress」242 ページ
- 「sa_convert_ml_progress_to_timestamp システムプロシージャー」『SQL Anywhere サーバー SQL リファレンス』
- 「sa_convert_timestamp_to_ml_progress システムプロシージャー」『SQL Anywhere サーバー SQL リファレンス』

挿入用のストアドプロシージャー

挿入用のストアドプロシージャーは、CREATE PUBLICATION 文で定義したように、CREATE TABLE 文で宣言されたカラムと同じ順序で、アップロードするすべてのカラムを含む結果セットを返す必要があります。

カラムの順序

t1 と呼ばれるテーブル内にあるカラムの作成順序を確認するには、次のクエリを使用します。

```
SELECT column_name
FROM SYSTAB JOIN SYSTABCOL
  WHERE table_name = 't1'
ORDER BY column_id
```

例

挿入用のストアドプロシージャーの定義方法の詳細については、「チュートリアル：スクリプト化されたアップロードの使用」324 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアドプロシージャー t1_insert を挿入プロシージャーとして登録します。t1_insert ストアドプロシージャーの定義の結果セットには、CREATE PUBLICATION 文にリストされたすべてのカラムが含まれますが、順序は、CREATE TABLE 文でカラムが宣言された順です。

```
CREATE TABLE t1(
  //The column ordering is taken from here
  pk integer primary key,
  c1 char( 30),
  c2 float,
  c3 double );

CREATE PROCEDURE t1_insert ()
RESULT( pk integer, c1 char(30), c3 double )
begin
```

```
  ...
end

CREATE PUBLICATION WITH SCRIPTED UPLOAD p1(
  // Order of columns here is ignored
  TABLE t1( c3, pk, c1 ) USING (
    PROCEDURE t1_insert FOR UPLOAD INSERT
  )
)
```

削除用のストアドプロシージャー

削除用のストアドプロシージャーは、CREATE PUBLICATION 文で定義したように、CREATE TABLE 文で宣言されたカラムと同じ順序で、アップロードするすべてのカラムを含む結果セットを返す必要があります。

カラムの順序

t1 と呼ばれるテーブル内にあるカラムの作成順序を確認するには、次のクエリを使用します。

```
SELECT column_name
FROM SYSTAB JOIN SYSTABCOL
  WHERE table_name = 't1'
ORDER BY column_id
```

例

削除用のストアドプロシージャーの定義方法の詳細については、「チュートリアル：スクリプト化されたアップロードの使用」324 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアドプロシージャー t1_delete を削除プロシージャーとして登録します。t1_delete ストアドプロシージャーの定義の結果セットには、CREATE PUBLICATION 文にリストされたすべてのカラムが含まれますが、順序は、CREATE TABLE 文でカラムが宣言された順です。

```
CREATE TABLE t1(
  //The column ordering is taken from here
  pk integer primary key,
  c1 char( 30),
  c2,   float,
  c3 double );

CREATE PROCEDURE t1_delete ()
RESULT( pk integer, c1 char(30), c3 double )
begin
  ...
end

CREATE PUBLICATION p1 WITH SCRIPTED UPLOAD(
  // Order of columns here is ignored
  TABLE t1( c3, pk, c1 ) USING (
    PROCEDURE t1_delete FOR UPLOAD DELETE
  )
)
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

更新用のストアドプロシージャー

更新用のストアドプロシージャーは、次に示す 2 つの値セットを含む結果セットを返す必要があります。

- 最初の値セットは更新前イメージを指定します (Mobile Link サーバーから最後に受信または正常にアップロードされたローの中の値)。
- 2 つ目の値セットは更新後イメージを指定します (統合データベースで更新される必要があるローの中の値)。

つまり、更新用のストアドプロシージャーは、挿入または削除用のストアドプロシージャーの 2 倍のカラムを持つ結果セットを返す必要があります。

例

更新用のストアドプロシージャーの定義方法の詳細については、「チュートリアル：スクリプト化されたアップロードの使用」324 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアドプロシージャー t1_update を更新プロシージャーとして登録します。パブリケーションは、同期させる 3 つのカラム (pk、c1、c3) を指定します。更新プロシージャーは、6 つのカラムを持つ結果セットを返します。最初の 3 つのカラムには、pk,、c1、c3 のカラムの更新前イメージが含まれています。2 つ目の 3 つのカラムには、同じカラムの更新後イメージが含まれています。どちらの場合も、カラムの順序は、CREATE PUBLICATION 文の順序ではなく、テーブルが作成されたときの順序です。

```
CREATE TABLE t1(
  //Column ordering is taken from here
  pk integer primary key,
  c1 char( 30),
  c2 float,
  c3 double );

CREATE PROCEDURE t1_update ()
RESULT( preimage_pk integer, preimage_c1 char(30), preimage_c3 double,
postimage_pk integer, postimage_c1 char(30), postimage_c3 double  )
BEGIN
  ...
END

CREATE PUBLICATION WITH SCRIPTED UPLOAD p1 (
       // Order of columns here is ignored
       TABLE t1( c3, pk, c1 ) USING (
  PROCEDURE t1_update FOR UPLOAD UPDATE
  )
)
```

スクリプト化されたアップロードのパブリケーション

スクリプト化されたアップロードパブリケーションを作成するには、キーワード WITH SCRIPTED UPLOAD を使用して、USING 句でストアドプロシージャーを指定します。

スクリプト化されたアップロードパブリケーションのテーブルに対してストアドプロシージャーを定義しないと、テーブルには操作はアップロードされません。ALTER PUBLICATION を使用して、通常のパブリケーションをスクリプト化されたアップロードパブリケーションに変更することはできません。

例

次のパブリケーションはストアドプロシージャーを使用して、t1 と t2 という 2 つのテーブルのデータをアップロードします。テーブル t1 には、挿入、削除、更新がアップロードされます。テーブル t2 には、挿入のみがアップロードされます。

```
CREATE PUBLICATION pub WITH SCRIPTED UPLOAD (
    TABLE t1 (col1, col2, col3) USING (
      PROCEDURE my.t1_ui FOR UPLOAD INSERT,
      PROCEDURE my.t1_ud FOR UPLOAD DELETE,
      PROCEDURE my.t1_uu FOR UPLOAD UPDATE
    ),
    TABLE t2 USING (
      PROCEDURE my.t2_ui FOR UPLOAD INSERT
    )
  )
```

参照

- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』
- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバー SQL リファレンス』

チュートリアル：スクリプト化されたアップロードの使用

このチュートリアルは、競合を検出するスクリプト化されたアップロードの設定方法を示しています。チュートリアルでは、スクリプト化されたアップロードで必要な統合データベースとリモートデータベース、ストアドプロシージャー、パブリケーション、サブスクリプションを作成します。このチュートリアルは、参考にするだけでもかまいませんし、テキストをコピーアンドペーストしてサンプルを実行することもできます。

レッスン 1：統合データベースの作成

このチュートリアルでは、ファイル名を指定し、ファイルが現在のディレクトリ (*scripteduploa*d) にあるものと想定しています。実際のアプリケーションでは、ファイルのフルパスを指定してください。

♦ 統合データベースの作成

1. 次のコマンドを実行して、チュートリアルのファイルを保存するディレクトリを作成し、そのディレクトリに移動します。

   ```
   md c:¥scripteduploa
   cd c:¥scripteduploa
   ```

2. 次のコマンドを実行して、統合データベースを作成します。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
dbinit consol.db
```

3. 次のコマンドを実行して、統合データベースの ODBC データソースを定義します。

```
dbdsn -w dsn_consol -y -c "UID=DBA;PWD=sql;DBF=consol.db;SERVER=consol"
```

4. データベースを Mobile Link 統合データベースとして使用するには、Mobile Link で使用するシステムテーブル、ビュー、ストアドプロシージャーを追加する設定スクリプトを実行する必要があります。次のコマンドを実行して、統合データベースとして *consol.db* を設定します。

```
dbisql -c "DSN=dsn_consol" "%SQLANY12%¥MobiLink¥setup¥syncsa.sql"
```

5. Interactive SQL を開き、dsn_consol DSN を使用して *consol.db* に接続するには、次のコマンドを実行します。

```
dbisql -c "DSN=dsn_consol"
```

6. 次の SQL 文を実行します。実行すると、統合データベースで employee テーブルが作成され、値をテーブルが挿入され、必要な同期スクリプトが作成されます。

```
CREATE TABLE employee (
  id      unsigned integer primary key,
  name    varchar( 256),
  salary  numeric( 9, 2 )
);

INSERT INTO employee VALUES( 100, 'smith', 225000 );
COMMIT;

CALL ml_add_table_script( 'default', 'employee', 'upload_insert',
     'INSERT INTO employee ( id, name, salary ) VALUES ( {ml r.id}, {ml r.name}, {ml r.salary} )' );

CALL ml_add_table_script( 'default', 'employee', 'upload_update',
     'UPDATE employee SET name = {ml r.name}, salary = {ml r.salary} WHERE id = {ml r.id}' );

CALL ml_add_table_script( 'default', 'employee', 'upload_delete',
     'DELETE FROM employee WHERE id = {ml r.id}' );

CALL ml_add_table_script( 'default', 'employee', 'download_cursor',
     'SELECT * from employee' );
```

チュートリアルに従って作業する間に、統合データベースに対してさらに SQL を実行するため、この SQL の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

7. 「レッスン 2：リモートデータベースの作成」325 ページに進みます。

レッスン 2：リモートデータベースの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」324 ページを参照してください。

♦ **リモートデータベースの作成**

1. サンプルディレクトリのコマンドプロンプトで、次のコマンドを実行して、リモートデータベースを作成します。

   ```
   dbinit remote.db
   ```

2. 次のコマンドを実行して、ODBC データソースを定義します。

   ```
   dbdsn -w dsn_remote -y -c "UID=DBA;PWD=sql;DBF=remote.db;SERVER=remote"
   ```

3. Interactive SQL を開き、dsn_remote を使用して *remote.db* に接続するには、次のコマンドを実行します。

   ```
   dbisql -c "DSN=dsn_remote"
   ```

4. 次の文のセットを実行して、リモートデータベースでオブジェクトを作成します。

   まず、同期させるテーブルを作成します。insert_time と delete_time カラムは同期されませんが、アップロードするローを指定するためにアップロードストアドプロシージャーで使用される情報が含まれます。

   ```
   CREATE TABLE employee (
       id          unsigned integer primary key,
       name        varchar( 256),
       salary      numeric( 9, 2 ),
       insert_time   timestamp default '1900-01-01'
   );
   ```

   チュートリアルに従って作業する間に、リモートデータベースに対してさらに SQL を実行するため、この SQL の実行完了後も、引き続き Interactive SQL を実行し、データベースに接続した状態にします。

5. 「レッスン 3：挿入の処理」326 ページに進みます。

次に、ストアドプロシージャーと、アップロードを処理するその他の処理を定義する必要があります。更新、挿入、削除ごとに別々に定義します。

レッスン 3：挿入の処理

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」324 ページを参照してください。

♦ **挿入の処理**

1. リモートデータベースに接続された Interactive SQL のインスタンスを使用し、次の SQL を使用して、ローを挿入するときに各ローに insert_time を設定するトリガーを作成します。

   ```
   CREATE TRIGGER emp_ins AFTER INSERT ON employee
   REFERENCING NEW AS newrow
   FOR EACH ROW
   BEGIN
      UPDATE employee SET insert_time = CURRENT TIMESTAMP
      WHERE id = newrow.id
   END;
   ```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

このタイムスタンプは、最後の同期以降にローが挿入されたかどうかを調べるのに使用されます。統合データベースからダウンロードされた挿入を dbmlsync が適用するときは、このトリガーは起動しません。これは、この例の後半で、FireTriggers 拡張オプションがオフに設定されるからです。ダウンロードによって挿入されたローは、employee テーブルが作成されたときに定義されたデフォルト値 1900-01-01 を insert_time として取得します。ローが新しい挿入として処理され、次の同期中にアップロードされるのを防ぐために、この値は、開始進行状況値よりも前に設定する必要があります。

2. さらにリモートデータベースで、アップロード用に挿入されたすべてのローを結果セットとして返すプロシージャーを作成します。

```
CREATE PROCEDURE employee_insert()
RESULT( id  unsigned integer,
        name varchar( 256 ),
        salary numeric( 9,2 )
      )
BEGIN
   DECLARE start_time timestamp;

   SELECT value
   INTO start_time
   FROM #hook_dict
   WHERE name = 'start progress as timestamp';

   // Upload as inserts all rows inserted after the start_time
   // that were not subsequently deleted
   SELECT id, name, salary
   FROM employee e
   WHERE insert_time > start_time AND
     NOT EXISTS( SELECT id FROM employee_delete ed  WHERE ed.id = e.id );

END;
```

このプロシージャーは、最後に正常にアップロードされてから (insert_time に基づいて) 挿入された後、続いて削除されなかったたすべてのローを返します。最後の正常なアップロードの時間は、#hook_dict テーブルの開始進行状況値から判別します。この例では、dbmlsync 拡張オプション LockTables にデフォルト設定を使用して、同期対象のテーブルをロックします。したがって、終了進行状況後に挿入されたローを除外する必要がありません。テーブルのロックによって、アップロードの構築中に、操作が終了進行状況後に発生することが防止されます。

3. 「レッスン 4：更新の処理」327 ページに進みます。

レッスン 4：更新の処理

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」324 ページを参照してください。

アップロードを処理するには、アップロード構築中、開始進行状況値に基づいて正しい更新前イメージを使用する必要があります。

♦ 更新の処理

1. リモートデータベースに接続された Interactive SQL のインスタンスを使用して、更新されたローの更新前イメージを保持するテーブルを作成します。スクリプト化されたアップロードを生成するときには、更新前イメージが使用されます。

```
CREATE TABLE employee_preimages (
  id          unsigned integer NOT NULL,
  name        varchar( 256),
  salary      numeric( 9, 2 ),
  img_time    timestamp default CURRENT TIMESTAMP,
  primary key( id, img_time )
);
```

2. 次に、各ローが更新されるときに、更新前イメージを保存するトリガーを作成します。挿入トリガーと同様、このトリガーもダウンロードでは起動しません。

```
CREATE TRIGGER emp_upd AFTER UPDATE OF name,salary ON employee
  REFERENCING OLD AS oldrow
  FOR EACH ROW
BEGIN
  INSERT INTO employee_preimages ON EXISTING SKIP VALUES(
    oldrow.id, oldrow.name, oldrow.salary, CURRENT TIMESTAMP );
END;
```

このトリガーは、ローが更新されるたびに更新前イメージを保存します (ただし、2 つの更新が非常に近く、タイムスタンプが同じ場合を除く)。これは一見、無駄に見えるので、ローの更新前イメージがテーブルにない場合のみ保存し、sp_hook_dbmlsync_upload_end フックに依存して、更新前イメージがアップロードされた後に削除しがちです。

しかし、sp_hook_dbmlsync_upload_end フックは、この目的では確実ではありません。アップロードを送信した後で受信確認される前に、ハードウェアまたはソフトウェアの障害により dbmlsync が停止した場合、フックが呼び出されない可能性があります。その結果、ローが正常にアップロードされても、ローは更新前イメージ テーブルから削除されません。また、通信障害が発生すると、dbmlsync はサーバーからアップロードの受信確認を受信できない場合があります。この場合、フックに渡されるアップロードステータスは「不明」になります。このステータスでは、フックは、更新前イメージテーブルがクリーンアップされたのかそのままなのかを判別できません。複数の更新前イメージを保存すると、アップロードが構築されるときに、開始進行状況値に基づいて正しい更新前イメージが常に選択されます。

3. 次に、更新を処理するアップロードプロシージャーを作成します。

```
CREATE PROCEDURE employee_update()
RESULT(
      preimage_id  unsigned integer,
      preimage_name varchar( 256),
      preimage_salary numeric( 9,2 ),
      postimage_id  unsigned integer,
      postimage_name varchar( 256),
      postimage_salary numeric( 9,2 )
    )
BEGIN
   DECLARE start_time timestamp;

   SELECT value
   INTO start_time
   FROM #hook_dict
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
    WHERE name = 'start progress as timestamp';

    // Upload as an update all rows that have been updated since
    // start_time that were not newly inserted or deleted.
    SELECT ep.id, ep.name, ep.salary, e.id, e.name, e.salary
    FROM employee e JOIN employee_preimages ep
        ON ( e.id = ep.id )
    // Do not select rows inserted since the start time. These should be
    // uploaded as inserts.
    WHERE insert_time <= start_time
      // Do not upload deleted rows.
      AND NOT EXISTS( SELECT id FROM employee_delete ed  WHERE ed.id = e.id )
      // Select the earliest pre-image after the start time.
      AND ep.img_time = ( SELECT MIN( img_time )
            FROM employee_preimages
            WHERE id = ep.id
            AND img_time > start_time );
END;
```

このストアドプロシージャーは、他のスクリプトの 2 倍のカラムを持つ結果セットを 1 つ返します。これには、更新前イメージ (Mobile Link サーバーから最後に受信したローと、正常にアップロードされたローの値) と更新後イメージ (統合データベースに入力される値) が含まれます。

更新前イメージは、start_progress 後に記録された employee_preimages の一番最初の値セットです。この例では、削除されてから再度挿入された既存のローは、正しく処理されません。より完全なソリューションでは、これらのローは更新としてアップロードされます。

4. 「レッスン 5：削除の処理」329 ページに進みます。

レッスン 5：削除の処理

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」324 ページを参照してください。

♦ 削除の処理

1. リモートデータベースに接続された Interactive SQL のインスタンスを使用して、削除されたローのリストを維持するテーブルを作成します。

```
CREATE TABLE employee_delete (
    id          unsigned integer  primary key NOT NULL,
    name        varchar( 256 ),
    salary      numeric( 9, 2 ),
    delete_time  timestamp
);
```

2. 次に、employee テーブルからローが削除されるときに employee_delete テーブルを移植するトリガーを作成します。

```
CREATE TRIGGER emp_del AFTER DELETE ON employee
REFERENCING OLD AS delrow
FOR EACH ROW
BEGIN
    INSERT INTO employee_delete
VALUES( delrow.id, delrow.name, delrow.salary, CURRENT TIMESTAMP );
END;
```

後で dbmlsync 拡張オプション FireTriggers を false に設定するので、このトリガーは、ダウンロード中は呼び出されません。このトリガーは、削除されたローは再度挿入されることはないことを前提としています。したがって、複数回削除されるローは処理されません。

3. 次の SQL 文は、削除を処理するアップロードプロシージャーを作成します。

```
CREATE PROCEDURE employee_delete()
RESULT( id  unsigned integer,
        name varchar( 256),
        salary numeric( 9,2 )
      )
BEGIN
   DECLARE start_time timestamp;

   SELECT value
   INTO start_time
   FROM #hook_dict
   WHERE name = 'start progress as timestamp';

  // Upload as a delete all rows that were deleted after the
  // start_time that were not inserted after the start_time.
  // If a row was updated before it was deleted, then the row
  // to be deleted is the pre-image of the update.
   SELECT IF ep.id IS NULL THEN ed.id ELSE ep.id ENDIF,
         IF ep.id IS NULL THEN ed.name ELSE ep.name ENDIF,
         IF ep.id IS NULL THEN ed.salary ELSE ep.salary ENDIF
   FROM employee_delete ed LEFT OUTER JOIN employee_preimages ep
        ON( ed.id = ep.id AND ep.img_time > start_time )
   WHERE
     // Only upload deletes that occurred since the last sync.
     ed.delete_time > start_time
     // Don't upload a delete for rows that were inserted since
     // the last upload and then deleted.
   AND NOT EXISTS (
     SELECT id
       FROM employee e
       WHERE e.id = ep.id AND e.insert_time > start_time )
   // Select the earliest preimage after the start time.
   AND ( ep.id IS NULL OR ep.img_time = (SELECT MIN( img_time )
                                         FROM employee_preimages
                                         WHERE id = ep.id
                                          AND img_time > start_time ) );
END;
```

ストアドプロシージャーは、統合データベースで削除するローが含まれる結果セットを返します。ストアドプロシージャーは employee_preimages テーブルを使用するので、ローが更新された後に削除されると、削除用にアップロードされるイメージは、最後に正常にダウンロードまたはアップロードされたイメージになります。

4. 「レッスン 6：更新前イメージテーブルのクリーンアップ」330 ページに進みます。

レッスン 6：更新前イメージテーブルのクリーンアップ

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」324 ページを参照してください。

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

♦ 更新前イメージテーブルのクリーンアップ

1. リモートデータベースに接続された Interactive SQL のインスタンスを使用して、アップロードが成功したときに employee_preimage と employee_delete テーブルをクリーンアップする upload_end フックを作成します。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end()
BEGIN
   DECLARE val   varchar(256);

   SELECT value
   INTO val
   FROM #hook_dict
   WHERE name = 'upload status';

   IF val = 'committed' THEN
     DELETE FROM employee_delete;
     DELETE FROM employee_preimages;
   END IF;
END;
```

   このチュートリアルでは、同期中にテーブルがロックされるように、dbmlsync 拡張オプション LockTables にデフォルト設定を使用します。したがって、テーブルのローに対して、end_progress 後に発生した操作が実行される心配がありません。ロックにより、このような操作が発生するのを防ぐことができます。

2. 「レッスン 7：パブリケーション、Mobile Link ユーザー、サブスクリプションの作成」331 ページに進みます。

レッスン 7：パブリケーション、Mobile Link ユーザー、サブスクリプションの作成

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。「レッスン 1：統合データベースの作成」324 ページを参照してください。

♦ パブリケーション、Mobile Link ユーザー、サブスクリプションの作成

1. 統合データベースに接続された Interactive SQL のインスタンスを使用して、次の SQL 文を実行します。pub1 と呼ばれるパブリケーションでは、スクリプト化されたアップロードの構文 (WITH SCRIPTED UPLOAD) が使用されます。このパブリケーションによって、employee テーブルのアーティクルが作成され、スクリプト化されたアップロード用に作成したばかりの 3 つのストアドプロシージャーが登録されます。u1 という Mobile Link ユーザーと、v1 と pub1 の間のサブスクリプションが作成されます。拡張オプション FireTriggers はオフに設定されるので、ダウンロードが適用されているときはリモートデータベースでトリガーが起動されません。これにより、次の同期中に、ダウンロードされた変更がアップロードされるのを防ぐことができます。

```
CREATE PUBLICATION pub1 WITH SCRIPTED UPLOAD (
TABLE employee( id, name, salary ) USING (
  PROCEDURE employee_insert FOR UPLOAD INSERT,
  PROCEDURE employee_update FOR UPLOAD UPDATE,
  PROCEDURE employee_delete FOR UPLOAD DELETE
    )
```

```
);

CREATE SYNCHRONIZATION USER u1;

CREATE SYNCHRONIZATION SUBSCRIPTION TO pub1 FOR u1
TYPE 'tcpip'
ADDRESS 'host=localhost'
OPTION FireTriggers='off'
SCRIPT VERSION 'default';
```

2. 「レッスン 8：スクリプト化されたアップロードの実行」332 ページに進みます。

レッスン 8：スクリプト化されたアップロードの実行

このレッスンは、受講者がこれまでのすべてのレッスンを終了していることを前提としています。 「レッスン 1：統合データベースの作成」324 ページを参照してください。

♦ スクリプト化されたアップロードの実行

1. リモートデータベースに接続された Interactive SQL のインスタンスを使用し、スクリプト化されたアップロードを使用して、同期するデータを挿入します。リモートデータベースで次の SQL 文を実行します。

```
INSERT INTO employee(id, name, salary) VALUES( 7, 'black', 700 );
INSERT INTO employee(id, name, salary) VALUES( 8, 'anderson', 800 );
INSERT INTO employee(id, name, salary) VALUES( 9, 'dilon', 900 );
INSERT INTO employee(id, name, salary) VALUES( 10, 'dwit', 1000 );
INSERT INTO employee(id, name, salary) VALUES( 11, 'dwit', 1100 );
COMMIT;
```

2. コマンドプロンプトで、Mobile Link サーバーを起動します。

```
mlsrv12 -c "DSN=dsn_consol" -o mlserver.mls -v+ -dl -zu+
```

3. dbmlsync を使用して同期を開始します。

```
dbmlsync -c "DSN=dsn_remote" -k -uo -o remote.mlc -v+
```

4. 次の SQL 文を実行して、挿入がアップロードされたことを確認します。

```
select * from employee
```

このレッスンの最初に挿入された値が表示されます。

5. 「クリーンアップ」332 ページに進みます。

クリーンアップ

チュートリアルを完了した後にコンピューターをクリーンアップするには、次のコマンドを実行します。

```
mlstop -h -w
dbstop -y -c server=consol
dbstop -y -c server=remote
```

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

```
dberase -y consol.db
dberase -y remote.db

del remote.mlc mlserver.mls
del remote.mlc mlserver.mls remote.rid
```

Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

索引

記号

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

A

B

C

D

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

E

F

G

H

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

I

J

L

M

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

N

O

P

R

S

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

T

U

V

W

Z

あ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

き

く

こ

さ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

し

す

せ

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

た

ち

つ

て

と

な

に

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1

へ

ほ

み

め

も

 Copyright © 2012, iAnywhere Solutions, Inc. - SQL Anywhere 12.0.1