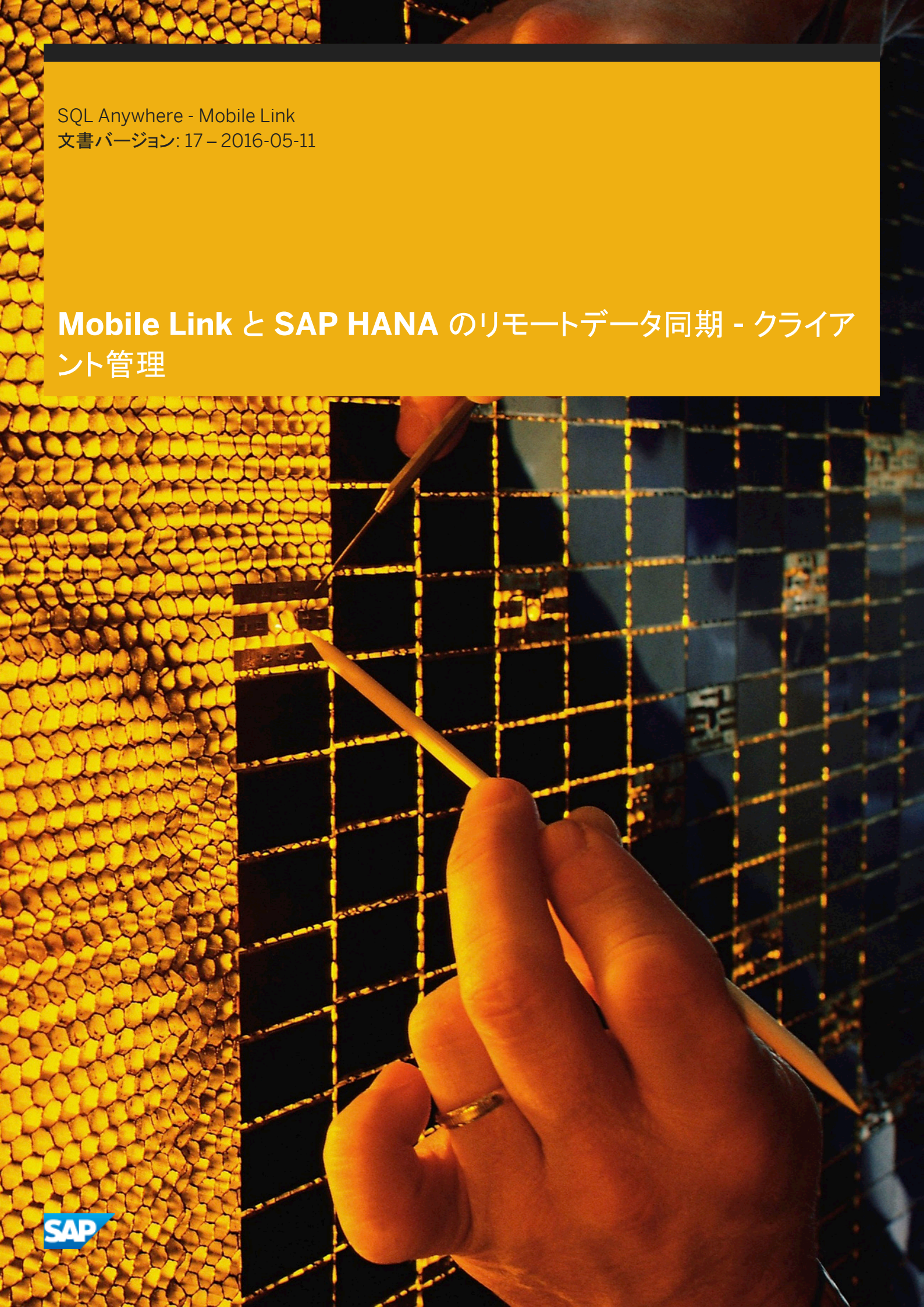
SQL Anywhere - Mobile Link
文書バージョン: 17 – 2016-05-11

Mobile Link と SAP HANA のリモートデータ同期 - クライアント管理



目次

1 Mobile Link と SAP HANA のリモートデータ同期 - クライアント管理

SQL Anywhere 製品に含まれる Mobile Link データ同期テクノロジには、同期サーバ、SQL Anywhere と Ultra Light データベースのクライアントソフトウェア、およびクライアントとサーバ間の通信プロトコルが含まれます。同期サーバは、SAP HANA プラットフォームのオプションとしても提供されます。これらの提供物は SAP HANA リモートデータ同期と呼ばれ、統合モニタリング、設定、ライフサイクル管理、ライセンス管理など数多くの統合機能を、同期サーバに追加します。

同期サーバを SQL Anywhere 製品の一部として取得した場合でも、SAP HANA リモートデータ同期のオプションとして取得した場合でも、Mobile Link クライアントの設定および操作方法は同じです。このマニュアルの Mobile Link クライアントに関する文書は一般的に、いずれかの同期サーバで動作するクライアントに適用されます。

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。

このセクションの内容:

1.1 Mobile Link クライアント

Mobile Link サーバは、現在 SQL Anywhere クライアントおよび Ultra Light クライアントの使用に対応しています。

このセクションの内容:

1.1.1 Mobile Link クライアントとしての SQL Anywhere

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、パブリケーションをユーザに結び付けるサブスクリプションです。

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。

関連情報

同期の開始 [97 ページ]
SQL Anywhere クライアント [74 ページ]
dbmlsync の同期のカスタマイズ [106 ページ]
パブリケーション [81 ページ]
Mobile Link ユーザ [90 ページ]
同期サブスクリプションの作成 [94 ページ]
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 [76 ページ]
Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync) の構文 [110 ページ]

1.1.2 Mobile Link クライアントとしての Ultra Light

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 または HTTPS 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。

1.1.3 クライアントのネットワークプロトコル

Mobile Link サーバでは、-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。

mlsrv17 コマンドラインオプションの構文は次のとおりです。

```
mlsrv17 -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。

```
mlsrv17 -c "DSN=SQL Anywhere 17 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;...)
```

例:

```
mlsrv17 -c "DSN=SQL Anywhere 17 Demo" -x tcpip(
   host=localhost;port=2439)
```

リモートデータ同期では、以下のフォーマットを使用します。

```
protocol=...
protocol_options=...
```

関連情報

Mobile Link クライアントネットワークプロトコルオプション [23 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]
CommunicationType (ctp) 拡張オプション [161 ページ]

1.1.4 Mobile Link のシステムテーブル

Mobile Link で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 サーバに必要な Mobile Link システムテーブルが作成されます。

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。

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。

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

- SYSSYNC システムビュー
- SYSPUBLICATION システムビュー
- SYSSUBSCRIPTION システムビュー
- SYSSYNCSCRIPT システムビュー
- SYSSYNCPROFILE システムビュー
- SYSARTICLE システムビュー
- SYSARTICLECOL システムビュー

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。

1.2 同期システムの Mobile Link ユーザ

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

ユーザ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 ユーザ名を作成してください。
- Mobile Link ユーザ名を Mobile Link サーバに登録してください。

Mobile Link ユーザの名前とパスワードは、データベースユーザの名前とパスワードとは異なります。Mobile Link ユーザ名は、リモートデータベースから Mobile Link サーバ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名と正確に一致する必要があります。Mobile Link ユーザ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 つのユーザ名を同じにすることはできません。たとえば、ユーザ名に **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 **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。

ユーザ名を使用して、Mobile Link サーバ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* パラメータを使用します。

Mobile Link ユーザ名は、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 ユーザ名を割り当てることもできます。

Ultra Light ユーザ認証

Ultra Light と Mobile Link のユーザ認証スキームは異なりますが、Ultra Light ユーザ ID の値を Mobile Link ユーザ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が使用している場合のみです。

このセクションの内容:

Mobile Link のユーザ認証システムは、ユーザ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 サーバに対してユーザ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認証メカニズムを実装することもできます。

認証処理 [18 ページ]
次のリストでは、認証中に発生するイベントの順序を説明します。

カスタムユーザ認証 [20 ページ]
組み込みの Mobile Link メカニズム以外のユーザ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。

関連情報

スクリプトでのリモート ID と Mobile Link ユーザ名 [16 ページ]

1.2.1 Mobile Link ユーザの作成と登録

Mobile Link ユーザ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 ユーザの作成

リモートデータベース側にユーザを追加する場合、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、SQL Central または CREATE SYNCHRONIZATION USER 文を使用します。
- Ultra Light リモートデータベースの場合は、User Name と Password 同期パラメータを設定します。

統合データベースへの Mobile Link ユーザ名の追加

リモートデータベースでユーザ名が作成された後、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名を登録できます。

- mluser ユーティリティを使用します。
- SQL Central を使用します。
- 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します。これらのスクリプトのどちらかを起動すると、Mobile Link サーバによって、認証が正常に行われるユーザが自動的に追加されます。
- mlsrv17 で -zu+ コマンドラインオプションを指定します。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奨めできません。

関連情報

Mobile Link ユーザ [90 ページ]

1.2.2 ユーザの最初の Mobile Link パスワードの設定 (SQL Central の場合)

各ユーザのパスワードは、ユーザ名とともに ml_user テーブルに格納されます。SQL Central は、個々のユーザとパスワードを追加する場合に便利です。

コンテキスト

パスワードがないユーザを作成した場合、そのユーザは 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。

手順

1. Mobile Link プロジェクトをダブルクリックします。
2. 統合データベースをダブルクリックし、統合データベース名をクリックします。
3. ユーザをクリックします。
4. ファイル > 新規 > ユーザ をクリックします。
5. ユーザ作成ウィザードの指示に従います。

結果

指定したパスワードでユーザが作成されます。

1.2.3 最初の Mobile Link パスワードの設定 (mluser ユーティリティ)

各ユーザのパスワードは、ユーザ名とともに ml_user テーブルに格納されます。mluser ユーティリティは、バッチで追加する場合に便利です。

コンテキスト

パスワードがないユーザを作成した場合、そのユーザは 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。

手順

1. 各行に 1 人分ずつ、ユーザ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では、ユーザ名とパスワードが含まれているファイルがある場合は -f オプションを使用し、単一のユーザ名とパスワードを指定する場合は -u および -p オプションを使用して、mluser ユーティリティを実行します。

結果

指定したユーザとパスワードが作成されます。

例

次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、ユーザ名とパスワードを含むファイルを指定します。

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

関連情報

ユーザの最初の Mobile Link パスワードの設定 (SQL Central の場合) [11 ページ]

1.2.4 新しいユーザからの同期

通常、Mobile Link サーバに接続するには、各 Mobile Link クライアントが有効な Mobile Link ユーザ名とパスワードを指定します。Mobile Link サーバの起動時に -zu+ オプションを指定すると、サーバは未登録のユーザ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からの要求を受信すると、要求は処理され、ユーザは 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

ユーザ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名が追加されます。このユーザ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

ユーザがパスワードを入力すると、ユーザ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名が認識されるようになります。これ以降、このユーザ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

新しいユーザは、[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、新しいユーザが Mobile Link システムに追加されます。これ以降、このユーザ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。

未知のユーザによる同期の防止

デフォルトでは、Mobile Link サーバは ml_user テーブルに登録されているユーザのみ認識します。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が間違ったユーザ名やスペルミスのあるユーザ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。

1.2.5 エンドユーザのパスワード

Mobile Link サーバでユーザ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は、毎回 Mobile Link ユーザ名とパスワードを入力しなければなりません。

ユーザ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、エンドユーザに対して Mobile Link ユーザ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。

SQL Anywhere

ユーザは、-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、ユーザ名とパスワードを再入力するように要求されます。

関連情報

-c dbmlsync オプション [122 ページ]
-mp dbmlsync オプション [134 ページ]
MobiLinkPwd (mp) 拡張オプション [174 ページ]

1.2.6 パスワードの変更

Mobile Link では、エンドユーザ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。インタフェースは、Ultra Light クライアントと SQL Anywhere クライアントで異なります。

ユーザ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。

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。

最初のパスワードは、統合データベースサーバで、または最初の同期で設定できます。

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。

関連情報

新しいユーザからの同期 [12 ページ]
ユーザの最初の Mobile Link パスワードの設定 (SQL Central の場合) [11 ページ]
-mp dbmlsync オプション [134 ページ]
-mn dbmlsync オプション [133 ページ]

1.2.7 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。データベースが Mobile Link と同期すると、リモート ID が NULL であることが Mobile Link クライアントによってチェックされます。リモート ID が NULL でることが確認されると、GUID がリモート ID として割り当てられます。リモート ID が設定されると、データベースは、手動で変更されない限り同じリモート ID を保持します (リモート ID を手動で変更することはお奨めしません)。

SQL Anywhere リモートデータベースでは、Mobile Link サーバが、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 によって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 ユーザ名やユニークな認証パラメータ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ます。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、すぐに再同期されます。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を呼び出します。

このセクションの内容:

1.2.7.1 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。

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。

開始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。

```
-- 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';
```

関連情報

リモート ID の設定 [79 ページ]

1.2.7.2 スクリプトでのリモート ID と Mobile Link ユーザ名

Mobile Link ユーザ名はユーザを識別し、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 s.remote_id) または Mobile Link ユーザ名 (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 ユーザ間の関係が 1 対 1 である配備の場合、リモート ID は無視できます。この場合、Mobile Link イベントスクリプトでは、username パラメータを参照できます。このパラメータは、認証に使用する Mobile Link ユーザ名です。

Mobile Link ユーザ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 ユーザ名を参照できます。Mobile Link ユーザ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、前のアップロード状態が不明な場合でも、異なるユーザ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。

1.2.8 ユーザ認証メカニズム

ユーザの認証は、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名とパスワードを使用して、認証が実行されます。

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 システムまたは LDAP の代わりにデータベースのユーザ認証を使用できます。

関連情報

カスタムユーザ認証 [20 ページ]

1.2.9 ユーザ認証アーキテクチャ

Mobile Link のユーザ認証システムは、ユーザ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 サーバに対してユーザ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名とパスワードを追加するには、SQL Central または mluser ユーティリティを使用するか、Mobile Link サーバの起動時に -zu+ オプションを指定します。

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

ユーザ名

Mobile Link ユーザ名。必須です。同期を実行するには、ユーザ名を 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 サーバを -zu+ 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。省略可能です。Mobile Link ユーザ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。

関連情報

Mobile Link ユーザの作成と登録 [10 ページ]
カスタムユーザ認証 [20 ページ]

1.2.10 認証処理

次のリストでは、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。
    1. 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。
    2. 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として渡され、このストアドプロシージャによって更新されます。最初のパラメータで返された値は、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。
    1. 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。
    2. 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 サーバは、指定された Mobile Link ユーザ名を ml_user テーブルで問い合わせます。
    1. 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。
    2. カスタムスクリプトが呼び出されず、指定された Mobile Link ユーザ名も ml_user テーブルにない場合は、Mobile Link サーバを -zu+ オプションで起動していれば、指定した Mobile Link ユーザ名が 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。
    3. 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。
    4. 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 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。
    1. パラメータが authenticate_parameters スクリプトに渡されます。
    2. 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。
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 ユーザ名に置き換えられます。
11. Mobile Link サーバは、authentication_status に関係なく、Mobile Link ユーザ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。

1.2.11 カスタムユーザ認証

組み込みの Mobile Link メカニズム以外のユーザ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です。
- 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です。

次の 3 つのカスタム認証ツールがあります。

- mlsrv17 -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

mlsrv17 -zu+ オプションを使用すると、ユーザ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 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。

ユーザ 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。

このセクションの内容:

Java と .NET のユーザ認証 [21 ページ]
Java クラスと .NET クラスではアプリケーションサーバなどのコンピューティング環境で使用されるユーザ名とパスワードの他のソースにアクセスできるため、ユーザ認証は Java と .NET の同期ロジックで本来使用されているものです。

1.2.11.1 Java と .NET のユーザ認証

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

%SQLANYSAMP17%¥MobiLink¥JavaAuthentication.ディレクトリに簡単な例があります。*%SQLANYSAMP17%*¥MobiLink¥JavaAuthentication¥CustEmpScripts.java 内のサンプルコードは、単純なユーザ認証システムを実装します。最初の同期時に、Mobile Link ユーザ名が 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 ID を追加する前のテストは行いません。

1.3 Mobile Link クライアントユーティリティ

Mobile Link ファイル転送ユーティリティ (mlfiletransfer) は、Mobile Link クライアントでは使用できません。

このセクションの内容:

1.3.1 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。
-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、ローカルパスは現行のディレクトリです。
*-p*password	Mobile Link ユーザ名のパスワード。
-q	クワイエットモード。メッセージを表示しません。

オプション	説明
-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。
file	転送するファイルのサーバでのファイル名。Mobile Link は、ダウンロード時に、-ftr ディレクトリの username サブフォルダでファイルを検索します。このサブフォルダ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。 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 mlsrv17 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。

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 ユーザごとにサブフォルダを作成します。

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。

Ultra Light ユーザは、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。

例

次のコマンドは、Mobile Link サーバを SQL Anywhere 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し始めるように Mobile Link サーバに指示します。この例では、Mobile Link サーバはまず 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 フォルダを Mobile Link サ

ーバにモニタ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。

```
mlsrv17 -c "DSN=SQL Anywhere 17 CustDB;uid=ml_server;pwd=sql" -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 17.0" -x tcpip notepad.exe
```

関連情報

Mobile Link クライアントネットワークプロトコルオプション [23 ページ]

1.4 Mobile Link クライアントネットワークプロトコルオプション

Mobile Link クライアントユーティリティは、Mobile Link サーバに接続すると、次の Mobile Link 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します。

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
Ultra Light	Stream Parameters 同期パラメータまたは -x オプション Ultra Light 同期ユーティリティ (ulsync)
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 • StreamHTTPParms インタフェース [Ultra Light J] • StreamHTTPSParms インタフェース [Ultra Light J]
Relay Server	Relay Server 設定ファイル
Mobile Link プロファイラ	Mobile Link プロファイラ (管理ツール) の起動
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)の -x オプション

このセクションの内容:

allow_expired_certs Mobile Link クライアントネットワークプロトコルオプション [29 ページ]
期限切れになったサーバ証明書またはまだ有効でないサーバ証明書を受け入れて、同期を続行します。

buffer_size Mobile Link クライアントネットワークプロトコルオプション [30 ページ]
ネットワークに書き込む前にバッファ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。

certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
このオプションを指定した場合、証明書に記載されている組織フィールドがこの値と一致する場合にのみ、アプリケーションはサーバ証明書を受け入れます。

certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
このオプションを指定した場合、証明書に記載されている通称フィールドがこの値と一致する場合にのみ、アプリケーションはサーバ証明書を受け入れます。

certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
このオプションを指定した場合、証明書に記載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のみ、アプリケーションはサーバ証明書を受け入れます。

client_port Mobile Link クライアントネットワークプロトコルオプション [36 ページ]
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。

compression Mobile Link クライアントネットワークプロトコルオプション [37 ページ]
使用するデータ圧縮の種類を設定します。

custom_header Mobile Link クライアントネットワークプロトコルオプション [39 ページ]
カスタム HTTP ヘッダを指定します。

e2ee_public_key Mobile Link クライアントネットワークプロトコルオプション [40 ページ]
エンドツーエンド暗号化に使用する、サーバ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。

fips Mobile Link クライアントネットワークプロトコルオプション [41 ページ]
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。

host Mobile Link クライアントネットワークプロトコルオプション [42 ページ]
Mobile Link サーバを実行中のコンピュータ、または、Web サーバを介して同期する場合は Web サーバを実行中のコンピュータのホスト名または IP アドレスを指定します。

http_buffer_responses Mobile Link クライアントネットワークプロトコルオプション [43 ページ]
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 からバッファにストリーミングしてから処理します。

http_password Mobile Link クライアントネットワークプロトコルオプション [44 ページ]
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 サーバとゲートウェイに対する認証を行います。

http_proxy_password Mobile Link クライアントネットワークプロトコルオプション [46 ページ]
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。

http_proxy_userid Mobile Link クライアントネットワークプロトコルオプション [47 ページ]
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。

http_userid Mobile Link クライアントネットワークプロトコルオプション [48 ページ]
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 サーバとゲートウェイに対する認証を行います。

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。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]
Mobile Link ファイル転送ユーティリティ (mlfiletransfer) [21 ページ]

1.4.1 プロトコルオプション

同期サーバとして選択するネットワークプロトコルは、クライアントが使用する同期プロトコルと一致する必要があります。Mobile Link サーバの接続オプションは、-x mlsrv17 オプション (またはリモートデータ同期内のプロトコルと protocol_options) を使用して設定されます。

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_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

TLS プロトコルオプション	詳細の参照先
certificate_unit=company_unit	certificate_unit
client_port=nnnnn[-mmmmm]	client_port
compression={*zlib*|*none*}	compression
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
network_adapter_name=name	network_adapter_name
network_leave_open={*off*|*on*}	network_leave_open
network_name=name	network_name
port=portnumber	port
skip_certificate_name_check={*off*|*on*}	skip_certificate_name_check
timeout=seconds	timeout
trusted_certificates=filename	trusted_certificate
trusted_certificate_name=name	trusted_certificate_name
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_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_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

HTTPS プロトコルオプション	詳細の参照先
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
skip_certificate_name_check={*off*|*on*}	skip_certificate_name_check
timeout=seconds	timeout
trusted_certificates=filename	trusted_certificate
trusted_certificate_name=name	trusted_certificate_name
url_suffix=suffix	url_suffix
version=HTTP-version-number	version
zlib_download_window_size=window-size	zlib_download_window_size
zlib_upload_window_size=window-bits	zlib_upload_window_size

1.4.2 allow_expired_certs Mobile Link クライアントネットワークプロトコルオプション

期限切れになったサーバ証明書またはまだ有効でないサーバ証明書を受け入れて、同期を続行します。

構文

allow_expired_certs={ *y* | *n* }

使用可能なプロトコル

HTTPS、TLS

デフォルト

同期が失敗して、エラーメッセージが表示されます。

備考

警告

このオプションを有効にすると、クライアントのセキュリティが弱まり、特定の状況下ではクライアントが攻撃を受けやすくなる可能性があります。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
skip_certificate_name_check Mobile Link プロトコルオプション [59 ページ]

1.4.3 buffer_size Mobile Link クライアントネットワークプロトコルオプション

ネットワークに書き込む前にバッファ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

- Windows Mobile - 16 KB
- Android、iOS、Linux ARM、Windows Phone/Store - 64 KB
- デスクトップ - 256 KB

備考

一般に、HTTP と HTTPS のバッファサイズが大きくなるほど、HTTP 要求応答のサイクル数は減りますが、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。

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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.4 certificate_company Mobile Link クライアントネットワークプロトコルオプション

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

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。

TLS または HTTPS 接続を開始する際、クライアントライブラリは、RFC 2818 に記述された手順を使用して、データベースサーバのホスト名をサーバが提供する証明書と照らし合わせます。この照合が行われるのは、certificate_name、certificate_company、certificate_unit のいずれのオプションも指定されていない場合、および skip_certificate_name_check オプションが有効でない場合です。certificate_name、certificate_company、certificate_unit のいずれかが指定されていれば、それらのオプションだけが確認されます。skip_certificate_name_check オプションを有効にすると、ホスト名のチェックが無効になります。

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。

サーバでは、実装は次のようになります。

```
mlsrv17
   -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   -x https(
     identity=myserver.id;
     identity_password=pwd)
```

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=passwd"
   -e "ctp=https;
       adr='host=myserver=c:¥certs¥rootca.crt;
          certificate_company=My Company;
          certificate_unit=My Division;
          certificate_name=My MobiLink 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 = "https";
    info.stream_parms =
```

```
        "trusted_certificates=c:¥cert¥rootca.crt;"
           "certificate_company=My Company;"
           "certificate_unit=My Division;"
           "certificate_name=My MobiLink Server";
```

関連情報

-e dbmlsync オプション [128 ページ]
trusted_certificates Mobile Link クライアントネットワークプロトコルオプション [62 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]
skip_certificate_name_check Mobile Link プロトコルオプション [59 ページ]

1.4.5 certificate_name Mobile Link クライアントネットワークプロトコルオプション

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

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。

TLS または HTTPS 接続を開始する際、クライアントライブラリは、RFC 2818 に記述された手順を使用して、データベースサーバのホスト名をサーバが提供する証明書と照らし合わせます。この照合が行われるのは、certificate_name、certificate_company、certificate_unit のいずれのオプションも指定されていない場合、および skip_certificate_name_check オプションが有効でない場合です。certificate_name、certificate_company、certificate_unit のいずれかが指定されていれば、それらのオプションだけが確認されます。skip_certificate_name_check オプションを有効にすると、ホスト名のチェックが無効になります。

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。

サーバでは、実装は次のようになります。

```
mlsrv17
   -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   -x https(
     identity=myserver.id;
     identity_password=pwd)
```

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=passwd"
   -e "ctp=https;
       adr='host=MyServer=c:¥certs¥rootca.crt;
          certificate_company=My Company;
          certificate_unit=My Division;
          certificate_name=My MobiLink 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 = "https";
    info.stream_parms =
       "trusted_certificates=c:¥cert¥rootca.crt;"
          "certificate_company=My Company;"
          "certificate_unit=My Division;"
          "certificate_name=My MobiLink Server";
```

関連情報

-e dbmlsync オプション [128 ページ]
trusted_certificates Mobile Link クライアントネットワークプロトコルオプション [62 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]
skip_certificate_name_check Mobile Link プロトコルオプション [59 ページ]

1.4.6 certificate_unit Mobile Link クライアントネットワークプロトコルオプション

このオプションを指定した場合、証明書に記載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のみ、アプリケーションはサーバ証明書を受け入れます。

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。

TLS または HTTPS 接続を開始する際、クライアントライブラリは、RFC 2818 に記述された手順を使用して、データベースサーバのホスト名をサーバが提供する証明書と照らし合わせます。この照合が行われるのは、certificate_name、certificate_company、certificate_unit のいずれのオプションも指定されていない場合、および skip_certificate_name_check オプションが有効でない場合です。certificate_name、certificate_company、certificate_unit のいずれかが指定されていれば、それらのオプションだけが確認されます。skip_certificate_name_check オプションを有効にすると、ホスト名のチェックが無効になります。

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。

サーバでは、実装は次のようになります。

```
mlsrv17
   -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   -x https(
```

```
      identity=myserver.id;
      identity_password=pwd)
```

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=passwd"
   -e "ctp=https;
       adr='host=MyServer=c:¥certs¥rootca.crt;
          certificate_company=My Company;
          certificate_unit=My Division;
          certificate_name=My MobiLink 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 = "https";
    info.stream_parms =
       "trusted_certificates=c:¥cert¥rootca.crt;"
          "certificate_company=My Company;"
          "certificate_unit=My Division;"
          "certificate_name=My MobiLink Server";
```

関連情報

-e dbmlsync オプション [128 ページ]
trusted_certificates Mobile Link クライアントネットワークプロトコルオプション [62 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]
skip_certificate_name_check Mobile Link プロトコルオプション [59 ページ]

1.4.7 client_port Mobile Link クライアントネットワークプロトコルオプション

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 サーバと通信する場合に役立ちます。

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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.8 compression Mobile Link クライアントネットワークプロトコルオプション

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

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。

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。

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"
```

関連情報

zlib_download_window_size Mobile Link クライアントネットワークプロトコルオプション [67 ページ]
zlib_upload_window_size Mobile Link クライアントネットワークプロトコルオプション [68 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.9 custom_header Mobile Link クライアントネットワークプロトコルオプション

カスタム HTTP ヘッダを指定します。

構文

```
custom_header=header
```

HTTP ヘッダの形式は、`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を含めます。複数のカスタムヘッダを指定するには、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。次に例を示します。*custom_header*=`header1:value1; customer_header`=`header2:value2`

カスタムヘッダは、カスタムヘッダ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。

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を含めることを要求します。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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.10 e2ee_public_key Mobile Link クライアントネットワークプロトコルオプション

エンドツーエンド暗号化に使用する、サーバ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。

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。

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 =
"trusted_certificates¥=rsaroot.crt;e2ee_public_key=rsapublic.pem"
```

関連情報

fips Mobile Link クライアントネットワークプロトコルオプション [41 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.11 fips Mobile Link クライアントネットワークプロトコルオプション

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。

構文

```
fips={ y | n }
```

使用可能なプロトコル

HTTPS、TLS

デフォルト

No

備考

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に接続できます。逆についても同様です。

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 および AES 暗号化の実装を使用します。

FIPS 認定の暗号化には別途ライセンスが必要です。

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。

サーバでは、実装は次のようになります。

```
mlsrv17
  -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
  -x tls(
    fips=y;
```

```
    identity=%SQLANYSAMP17%¥Certificates¥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=
       'fips=y;
        trusted_certificates=%SQLANYSAMP17%¥Certificates¥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 =
       "fips=y;"
       "trusted_certificates=C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥
¥Samples¥¥Certificates¥¥rsaroot.crt;"
       "certificate_name=RSA Server;"
       "certificate_company=SAP;"
       "certificate_unit=SQL Anywhere";
```

関連情報

e2ee_public_key Mobile Link クライアントネットワークプロトコルオプション [40 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.12 host Mobile Link クライアントネットワークプロトコルオプション

Mobile Link サーバを実行中のコンピュータ、または、Web サーバを介して同期する場合は Web サーバを実行中のコンピュータのホスト名または 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 パートナシップが設定されているデスクトップコンピュータの 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 パートナシップが設定されているデスクトップコンピュータ上の Mobile Link サーバに、Windows Mobile デバイスを接続できます。

例

次の例では、クライアントはポート 1234 にあるコンピュータ myhost 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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.13 http_buffer_responses Mobile Link クライアントネットワークプロトコルオプション

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 からバッファにストリーミングしてから処理します。

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 プロキシサーバでは、サーバ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。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.14 http_password Mobile Link クライアントネットワークプロトコルオプション

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 ヘッダに含められますが、HTTPS を使用すると、ヘッダ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。

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 サーバに対する認証のユーザ 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=passwd";
```

関連情報

http_userid Mobile Link クライアントネットワークプロトコルオプション [48 ページ]
http_proxy_password Mobile Link クライアントネットワークプロトコルオプション [46 ページ]
http_proxy_userid Mobile Link クライアントネットワークプロトコルオプション [47 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.15 http_proxy_password Mobile Link クライアントネットワークプロトコルオプション

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_userid と併用する必要があります。

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 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=passwd";
```

関連情報

http_password Mobile Link クライアントネットワークプロトコルオプション [44 ページ]

http_userid Mobile Link クライアントネットワークプロトコルオプション [48 ページ]
http_proxy_userid Mobile Link クライアントネットワークプロトコルオプション [47 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.16 http_proxy_userid Mobile Link クライアントネットワークプロトコルオプション

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。

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 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=passwd";
```

関連情報

http_password Mobile Link クライアントネットワークプロトコルオプション [44 ページ]
http_userid Mobile Link クライアントネットワークプロトコルオプション [48 ページ]
http_proxy_password Mobile Link クライアントネットワークプロトコルオプション [46 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.17 http_userid Mobile Link クライアントネットワークプロトコルオプション

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 ヘッダに含められますが、HTTPS を使用すると、ヘッダ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。

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 サーバに対する認証のユーザ 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=passwd";
```

関連情報

http_password Mobile Link クライアントネットワークプロトコルオプション [44 ページ]
http_proxy_password Mobile Link クライアントネットワークプロトコルオプション [46 ページ]
http_proxy_userid Mobile Link クライアントネットワークプロトコルオプション [47 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.18 identity Mobile Link クライアントネットワークプロトコルオプション

このオプションを使用すると、サードパーティのサーバ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。

クライアント側の証明書を使用すると、クライアント側の証明書認証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とプロキシに対して認証することができます。クライアント側の証明書は、クライアントと Mobile Link サーバ間に位置しています。クライアント側の証明書では、Mobile Link サーバによるクライアント側認証も提供されます。そのために、getCertificateChain メソッドを使用し、証明書の信頼性をテストして、authentication_user connection イベント内部からクライアント側の証明書を取り出します。

関連情報

identity_password Mobile Link クライアントネットワークプロトコルオプション [50 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.19 identity_password Mobile Link クライアントネットワークプロトコルオプション

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。

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。

関連情報

identity Mobile Link クライアントネットワークプロトコルオプション [49 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.20 network_adapter_name Mobile Link クライアントネットワークプロトコルオプション

Mobile Link クライアントが Mobile Link への接続に使用するネットワークアダプタ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 デスクトップでのみ有効です。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.21 network_leave_open Mobile Link クライアントネットワークプロトコルオプション

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。

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";
```

関連情報

network_name Mobile Link クライアントネットワークプロトコルオプション [53 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.22 network_name Mobile Link クライアントネットワークプロトコルオプション

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。

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から接続できます。自動ダイヤルとは、Mobile Link サーバへの接続の 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。

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";
```

関連情報

同期のスケジュール [104 ページ]
network_leave_open Mobile Link クライアントネットワークプロトコルオプション [52 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.23 persistent Mobile Link クライアントネットワークプロトコルオプション

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。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.24 port Mobile Link クライアントネットワークプロトコルオプション

Mobile Link サーバ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。

備考

ポート番号は 10 進数で、Mobile Link サーバ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 サーバポートを指定してください。

例

次の例では、クライアントはコンピュータ 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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.25 proxy_host Mobile Link クライアントネットワークプロトコルオプション

プロキシサーバのホスト名または 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。

例

次の例では、クライアントはコンピュータ myproxyhost 上で実行されているプロキシサーバのポート 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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.26 proxy_port Mobile Link クライアントネットワークプロトコルオプション

プロキシサーバ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。

例

次の例では、クライアントはコンピュータ myproxyhost 上で実行されているプロキシサーバのポート 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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.27 set_cookie Mobile Link クライアントネットワークプロトコルオプション

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、ユーザエージェントが Web サーバ、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。

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;";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.28 skip_certificate_name_check Mobile Link プロトコルオプション

クライアントライブラリがサーバホスト名とデータベースサーバの証明書ホスト名との照合を省略するかどうかを制御します。

使用可能なプロトコル

HTTPS、TLS

デフォルト

なし

備考

データベースサーバのホスト名がサーバの証明書のホスト名のいずれかと一致するかどうか制御するには、このブールオプションを ON または OFF に設定します。ただし、このオプションを有効にすると、クライアントはサーバの認証を完全に行うことができないので、攻撃に対して脆弱なままとなりますので、注意してください。

TLS または HTTPS 接続を開始する際、クライアントライブラリは、データベースサーバのホスト名をサーバが提供する証明書と照らし合わせます。この照合が行われるのは、certificate_name、certificate_company、certificate_unit のいずれの

オプションも指定されていない場合、あるいは、skip_certificate_name_check オプションが有効でない場合です。certificate_name、certificate_company、certificate_unit のいずれかが指定されていれば、それらのオプションだけが確認されます。skip_certificate_name_check オプションを有効にすると、ホスト名のチェックが無効になります。

クライアントは、subjectAltName (サブジェクトの別名、SAN または) 拡張内のホスト名、および SAN が存在しない場合は通称フィールド (CN) 内のホスト名と、サーバのホスト名を照合します。SAN には、ワイルドカードによって複数のホスト名が指定されている場合があります。たとえば、Google 社の証明書には、*.google.com、*.google.ca、*.android.com が指定されているかもしれません。その場合、www.google.ca は有効なホスト名ということになります。

HTTPS は、Web サービスのクライアントプロシージャだけでサポートされています。

例

次の接続文字列フラグメントは、myrootcert.crt がデータベースサーバの証明書署名チェーンのルートであることを確認しますが、myhost.com が証明書で識別されることは確認しません。

```
HOST=myhost.com;SERVER=myserver;ENCRYPTION=TLS(trusted_certificate=myrootcert.crt;
skip_certificate_name_check=ON)
```

関連情報

-e dbmlsync オプション [128 ページ]
trusted_certificates Mobile Link クライアントネットワークプロトコルオプション [62 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.29 timeout Mobile Link クライアントネットワークプロトコルオプション

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 サーバに通知します。また、Mobile Link は活性更新を送り返して、サーバ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。

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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.30 trusted_certificates Mobile Link クライアントネットワークプロトコルオプション

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。

構文

```
trusted_certificates=filename
```

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし

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_certificates パラメータ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_certificates パラメータと 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。

同じストリームオプションセット内で trusted_certificates オプションと trusted_certificate_name オプションを両方とも使用することはできません。

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。

サーバでは、実装は次のようになります。

```
mlsrv17
   -c "DSN=SQL Anywhere 17 Demo;UID=DBA;PWD=sql"
   -x https(
     identity=%SQLANYSAMP17%¥Certificates¥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=passwd"
   -e "ctp=https;
       adr='trusted_certificates=%SQLANYSAMP17%¥Certificates¥rsaroot.crt;
          certificate_name=RSA Server;
          certificate_company=SAP;
          certificate_unit=SQL Anywhere'"
```

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 = "https";
    info.stream_parms =
       "trusted_certificates=C:¥¥Users¥¥Public¥¥Documents¥¥SQL Anywhere 17¥
¥Samples¥¥Certificates¥¥rsaroot.crt;"
       "certificate_name=RSA Server;"
       "certificate_company=SAP;"
       "certificate_unit=SQL Anywhere";
```

関連情報

trusted_certificate_name Mobile Link クライアントネットワークプロトコルオプション [63 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.31 trusted_certificate_name Mobile Link クライアントネットワークプロトコルオプション

リモートデータベースに格納されている証明書のユニークな名前を指定します。このオプションは、SQL Anywhere クライアントでのみ使用できます。

構文

trusted_certificate_name=name

使用可能なプロトコル

- TLS、HTTPS

デフォルト

なし

備考

このオプションを指定すると、指定した証明書が dbmlsync によってデータベースから取得され、サーバの ID を検証するための信頼できる証明書として扱われます。

同じストリームオプションセット内で trusted_certificate_name オプションと trusted_certificates オプションを両方とも使用することはできません。

例

次の例では、信頼できるルート証明書として使用する証明書が `root.crt` にあるとします。データベースに証明書を挿入するには、次の構文を使用します。

```
CREATE CERTIFICATE myroot FROM FILE 'root.crt'
```

次に、以下の dbmlsync オプションを使用して証明書にアクセスします。

```
ALTER SYNCHRONIZATION SUBSCRIPTION mysub
TYPE tls
ADDRESS 'trusted_certificate_name=myroot;...'
```

関連情報

trusted_certificates Mobile Link クライアントネットワークプロトコルオプション [62 ページ]
certificate_company Mobile Link クライアントネットワークプロトコルオプション [31 ページ]
certificate_name Mobile Link クライアントネットワークプロトコルオプション [33 ページ]
certificate_unit Mobile Link クライアントネットワークプロトコルオプション [35 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.32 url_suffix Mobile Link クライアントネットワークプロトコルオプション

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、IIS を使用しているか、Apache Relay Server を使用しているかによって異なります。

リダイレクタ	suffix の構文
IIS	Windows の場合のデフォルトは /rs/client/rs.dll です。
Apache	Linux の場合のデフォルトは /cli/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

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

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 設定ファイルのマニュアルを参照してください。

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.33 version Mobile Link クライアントネットワークプロトコルオプション

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。

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";
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.34 zlib_download_window_size Mobile Link クライアントネットワークプロトコルオプション

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-bits + 3)
```

```
download (decompress): memory = 2(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。

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。

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

関連情報

compression Mobile Link クライアントネットワークプロトコルオプション [37 ページ]
zlib_upload_window_size Mobile Link クライアントネットワークプロトコルオプション [68 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.4.35 zlib_upload_window_size Mobile Link クライアントネットワークプロトコルオプション

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-size + 3)
```

```
download (decompress): memory = 2^(window-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。

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"
```

関連情報

compression Mobile Link クライアントネットワークプロトコルオプション [37 ページ]
zlib_download_window_size Mobile Link クライアントネットワークプロトコルオプション [67 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.5 リモート Mobile Link 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。

このセクションの内容:

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。

1.5.1 スクリプトバージョンとサブスクリプション

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。

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奨めします。既存のサブスクリプションは、以下の例に示す手順でアップグレードできます。

例

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。
   1. -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。
   2. 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。

1.5.2 配備された SQL Anywhere リモートデータベースへのテーブルの追加

SQL Anywhere リモートデータベースを配備した後、それにテーブルを追加することができます。

前提条件

そのパブリケーションの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- パブリケーションでの ALTER 権限
- SYS_REPLICATION_ADMIN_ROLE システムロール

コンテキスト

注記

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。

手順

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。
4. 同期を実行します。

結果

リモートデータベースにテーブルが追加されます。

次のステップ

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。

関連情報

sp_hook_dbmlsync_schema_upgrade [270 ページ]

1.5.3 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

配備された SQL Anywhere リモートデータベース内のテーブル定義は変更できます。

前提条件

そのパブリケーションの所有者であるか、または次のいずれかの権限を持っていることが必要です。

- パブリケーションでの ALTER 権限
- SYS_REPLICATION_ADMIN_ROLE システムロール

コンテキスト

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。

手順

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。
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。

結果

パブリッシュ済みのテーブルが変更されます。

関連情報

sp_hook_dbmlsync_schema_upgrade [270 ページ]

1.5.4 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。

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を実行する場合は、次の点を考慮してください。

- 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 文を参照してください。

1.6 Mobile Link 用 SQL Anywhere クライアント

Mobile Link サーバは、Mobile Link クライアントとして SQL Anywhere の使用に対応しています。

このセクションの内容:

1.6.1 SQL Anywhere クライアント

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。

このセクションの内容:

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 [76 ページ]

SQL Anywhere データベースを Mobile Link システムのリモートデータベースとして使用するには、パブリケーション、Mobile Link ユーザ、同期サブスクリプションを作成して、ユーザを統合データベースに登録する必要があります。

パブリケーション [81 ページ]

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)。

Mobile Link ユーザ [90 ページ]

Mobile Link のユーザ名は、Mobile Link サーバに接続するときの認証に使用されます。Mobile Link ユーザ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。

同期サブスクリプションの作成 [94 ページ]

Mobile Link ユーザ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。

同期の開始 [97 ページ]

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、同期のインタフェースは異なり、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。

同期のスケジュール [104 ページ]

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 の同期のカスタマイズ [106 ページ]

dbmlsync の同期をカスタマイズするには、イベントフック、dbmlsync プログラミングインタフェース、およびスクリプト化されたアップロードを使用します。

Dbmlsync API [107 ページ]

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フェースです。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。

SQL Anywhere クライアントのログ [107 ページ]

SQL Anywhere リモートデータベースを使用して Mobile Link アプリケーションを作成する場合、dbmlsync のメッセージログと SQL Anywhere トランザクションログの 2 種類のクライアントログファイルがあります。

Mobile Link の Mac OS X での実行 [108 ページ]

Mobile Link サーバと 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。

バージョンに関する考慮事項 [110 ページ]

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と一致している必要があります。

1.6.1.1 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

SQL Anywhere データベースを Mobile Link システムのリモートデータベースとして使用するには、パブリケーション、Mobile Link ユーザ、同期サブスクリプションを作成して、ユーザを統合データベースに登録する必要があります。

コンテキスト

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。

> **注記**
>
> 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。

手順

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。
3. リモートデータベースに Mobile Link ユーザを作成します。
4. ユーザを統合データベースに登録します。
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。

結果

SQL Anywhere データベースがリモートデータベースとして設定されます。

このセクションの内容:

進行オフセット [80 ページ]
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。

関連情報

パブリケーション [81 ページ]
Mobile Link ユーザ [90 ページ]
Mobile Link ユーザの作成と登録 [10 ページ]
同期サブスクリプションの作成 [94 ページ]
スクリプト化されたアップロード [295 ページ]
ダウンロード専用のパブリケーション [85 ページ]

1.6.1.1.1 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

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。

コンテキスト

開始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。

手順

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 オプションです。

2. リモートデータベースごとに、次の操作を実行します。
   a. 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。
   b. 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。
   c. 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、トランザクションログファイル名を変更する必要はありません。
   d. 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、パラメータ化されたスクリプトにすることができます。

結果

リモートデータベースが配備されます。

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 *%SQLANYSAMP17%*¥MobiLink¥Contact¥customize.sql にあります。

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

関連情報

リモート ID の設定 [79 ページ]
進行オフセット [80 ページ]

1.6.1.1.2 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。データベースが Mobile Link と同期すると、Mobile Link はリモート ID が NULL かどうかを確認し、NULL である場合は、GUID をリモート ID として割り当てます。リモート ID が設定されると、データベースは、手動で変更されない限り同じリモート 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 SQL Central の SQL Anywhere17 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。

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'
```

関連情報

リモート ID [15 ページ]

1.6.1.1.3 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。

1.6.1.1.4 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 ユーザが同期を行うと、その Mobile Link ユーザ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。

関連情報

トランザクションログファイル [101 ページ]
-r dbmlsync オプション [142 ページ]

1.6.1.2 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にリンクするためのサブスクリプションを作成します。

SQL 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。

SQL Central では、パブリケーションフォルダ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の動作は Mobile Link サーバと同期されません。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が存在するという想定に基づいています。

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、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。
- 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、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。

このセクションの内容:

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。

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 [86 ページ]
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。

アーティクルの追加 [87 ページ]
アーティクルを作成すると、パブリケーションに追加できます。

アーティクルの削除 [88 ページ]
アーティクルが必要でなくなった場合は、削除できます。

既存のパブリケーションの修正 (SQL の場合) [88 ページ]
アーティクルとパブリケーションは、作成した後で修正できます。

パブリケーションの削除 [89 ページ]
パブリケーションが必要でなくなった場合は、削除できます。

関連情報

SendTriggers (st) 拡張オプション [183 ページ]

1.6.1.2.1 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。

前提条件

SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

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。

手順

1. SQL Anywhere17 プラグインを使用して、リモートデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
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。

1.6.1.2.2 テーブル内の一部のカラムだけをパブリッシュする

テーブルのローをすべて含みカラムを一部だけしか含まないパブリケーションを作成できます。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

コンテキスト

注記

- 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。

手順

1. SQL Anywhere17 プラグインを使用して、リモートデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
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. 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

選択したカラムがパブリッシュされます。

1.6.1.2.3 WHERE 句を使用したパブリケーションの作成

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に制限できます。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

コンテキスト

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。

手順

1. SQL Anywhere17 プラグインを使用して、リモートデータベースに接続します。
2. パブリケーションをダブルクリックします。
3. ファイル > 新規 > パブリケーション をクリックします。
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。

結果

新しいパブリケーションが作成されます。

1.6.1.2.4 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。

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。
トランザクションログファイルが必要です。	トランザクションログファイルは必要ありません。

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
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、トランザクション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場合、同期処理は同期トランケーションポイントに影響しません。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。

1.6.1.2.5 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

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。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

DBA 権限を持つユーザか、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。

コンテキスト

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。

手順

1. 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。

結果

パブリケーションまたはアーティクルが変更されます。

関連情報

リモート Mobile Link クライアントでのスキーマの変更 [69 ページ]

1.6.1.2.6 アーティクルの追加

アーティクルを作成すると、パブリケーションに追加できます。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

DBA 権限を持つユーザか、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。

手順

1. SQL Anywhere17 プラグインを使用して、リモートデータベースに接続します。
2. パブリケーションをダブルクリックします。
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。終了をクリックします。

結果

指定したアーティクルがパブリケーションに追加されます。

1.6.1.2.7 アーティクルの削除

アーティクルが必要でなくなった場合は、削除できます。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

DBA 権限を持つユーザか、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。

手順

1. SQL Anywhere17 プラグインを使用して、データベースに接続します。
2. パブリケーションをダブルクリックします。
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。

結果

選択したアーティクルが削除されます。

1.6.1.2.8 既存のパブリケーションの修正 (SQL の場合)

アーティクルとパブリケーションは、作成した後で修正できます。

前提条件

既存のリモートデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っています。

DBA 権限を持つユーザか、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。

手順

1. 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。

結果

パブリケーションまたはアーティクルが変更されます。

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
```

1.6.1.2.9 パブリケーションの削除

パブリケーションが必要でなくなった場合は、削除できます。

前提条件

パブリケーションを削除する場合は、SYS_REPLICATION_ADMIN_ROLE システムロールが設定されたユーザ ID を持っているか、またはパブリケーションの所有者であることが必要です。

手順

1. SQL Anywhere17 プラグインを使用して、リモートデータベースに接続します。
2. パブリケーションをダブルクリックします。
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。

結果

選択したパブリケーションが削除されます。

1.6.1.3 Mobile Link ユーザ

Mobile Link のユーザ名は、Mobile Link サーバに接続するときの認証に使用されます。Mobile Link ユーザ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 ユーザは、データベースユーザとは異なります。データベースユーザ名と一致する Mobile Link ユーザ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。

このセクションの内容:

関連情報

Mobile Link ユーザの作成と登録 [10 ページ]

1.6.1.3.1 Mobile Link ユーザのリモートデータベースへの追加 (SQL Central の場合)

統合データベースの Mobile Link サーバに接続するときに認証に使用されるリモートデータベースで Mobile Link ユーザを作成します。

前提条件

既存のデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザ ID が必要です。

手順

1. SQL Anywhere17 プラグインを使用して、データベースに接続します。
2. *Mobile Link* ユーザをダブルクリックします。
3. ファイル > 新規 > *Mobile Link* ユーザ をクリックします。
4. 新しい *Mobile Link* ユーザの名前を指定してくださいフィールドに、Mobile Link ユーザの名前を入力します。
5. 完了をクリックします。

結果

Mobile Link ユーザが作成されます。

次のステップ

統合データベースに Mobile Link ユーザを登録します。

関連情報

Mobile Link ユーザの作成と登録 [10 ページ]

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

統合データベースの Mobile Link サーバに接続するときに認証に使用されるリモートデータベースで Mobile Link ユーザを作成します。

前提条件

既存のデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザ ID が必要です。

手順

1. 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 ユーザ名は同期システム内でユニークである必要があります。

   Mobile Link ユーザのプロパティは、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。

結果

Mobile Link ユーザが作成されます。

例

次は、Mobile Link ユーザ SSinger を追加する例です。

```
CREATE SYNCHRONIZATION USER SSinger
```

次のステップ

統合データベースに Mobile Link ユーザを登録します。

関連情報

Mobile Link ユーザの作成と登録 [10 ページ]

1.6.1.3.3 Mobile Link ユーザ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。

前提条件

既存のデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザ ID が必要です。

コンテキスト

sp_hook_dbmlsync_set_extended_options ストアドプロシージャを使用すると、次の同期の動作をプログラムによってカスタマイズできます。

手順

1. ユーザをダブルクリックします。
2. Mobile Link ユーザ名をダブルクリックし、プロパティを選択します。
3. 必要に応じてプロパティを変更します。

結果

Mobile Link ユーザに指定したオプションが保存されます。

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
dbmlsync 拡張オプション [100 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]

1.6.1.3.4 Mobile Link ユーザの拡張オプションの格納 (SQL の場合)

リモートデータベースで各 Mobile Link ユーザ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。

前提条件

既存のデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持つユーザ ID が必要です。

コンテキスト

sp_hook_dbmlsync_set_extended_options 拡張オプションを使用すると、次の同期の動作をプログラムによってカスタマイズできます。

手順

1. データベースに接続します。
2. ALTER SYNCHRONIZATION USER 文を実行します。

   Mobile Link ユーザ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。

結果

Mobile Link ユーザに指定したオプションが保存されます。

例

次は、Mobile Link ユーザ 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
```

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
dbmlsync 拡張オプション [100 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]

1.6.1.4 同期サブスクリプションの作成

Mobile Link ユーザ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。

注記

Mobile Link ユーザ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 サーバと同期できます。複数のサーバとの同期を行うには、サーバごとに異なる Mobile Link ユーザを作成します。

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 ユーザを作成します。この場合、Mobile Link ユーザは 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'
   ```

このセクションの内容:

関連情報

パブリケーション [81 ページ]
Mobile Link ユーザ [90 ページ]

1.6.1.4.1 Mobile Link 同期サブスクリプションの変更

既存の同期サブスクリプションを変更できます。

前提条件

既存のデータベースがあり、SYS_REPLICATION_ADMIN_ROLE システムロールを持っていることが必要です。

手順

1. データベースに接続します。
2. ユーザをダブルクリックします。
3. ユーザをダブルクリックします。
4. 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。

結果

同期サブスクリプションが変更されます。

1.6.1.4.2 Mobile Link サブスクリプションの削除

Mobile Link ユーザの同期サブスクリプションが必要でなくなった場合は、削除できます。

前提条件

同期サブスクリプションを削除する場合は、SYS_REPLICATION_ADMIN_ROLE システムロールが必要です。

手順

1. データベースに接続します。
2. ユーザをダブルクリックします。

3. Mobile Link ユーザをダブルクリックします。
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。

結果

選択した同期サブスクリプションが削除されます。

1.6.1.5 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、同期のインタフェースは異なり、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 サーバで使用されるスクリプトが決まります。

dbmlsync の権限

同期を行うには、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザ ID によって 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。

同期のカスタマイズ

dbmlsync の同期をカスタマイズするには、イベントフック、dbmlsync プログラミングインタフェース、およびスクリプト化されたアップロードを使用します。

このセクションの内容:

関連情報

1.6.1.5.1 ロールベースのアクセス制御と同期の場合のセキュリティに関する考慮事項

同期を実行する場合は、ユーザに SYS_RUN_REPLICATION_ROLE が付与されていることが必要です。SYS_RUN_REPLICATION_ROLE はユーザにシステム権限を付与します。しかし、これらの権限を使用できるのは、ユーザが dbmlsync または SQL Remote のような認証ツールを介してログインした場合に限られます。これは、REMOTE DBA の動作方法と似ています。

デフォルトでは、SYS_RUN_REPLICATION_ROLE システムロールに SYS_AUTH_DBA_ROLE 互換ロールが含まれます。ただし、SYS_RUN_REPLICATION_ROLE システムロールから SYS_AUTH_DBA_ROLE 互換ロールを取り消すことができます。SYS_AUTH_DBA_ROLE 互換ロールは、SYS_RUN_REPLICATION_ROLE システムロールから削除できる唯一の権限です。

SYS_AUTH_DBA_ROLE 互換ロールの権限は、通常、同期するのに必要な権限よりも強力です。より安全な同期環境を設定するには、以下のいずれかの方法に従います。

- SYS_AUTH_DBA_ROLE を SYS_RUN_REPLICATION_ROLE システムロールから取り消して、次のシステム権限を付与します。
  - INSERT ANY TABLE
  - UPDATE ANY TABLE
  - DELETE ANY TABLE
  - ALTER ANY TABLE
  - EXECUTE ANY PROCEDURE
  - フックに含まれる文が必要とする任意のシステム権限
  - スクリプト化されたアップロードを定義する場合に使用するストアドプロシージャ内の文が必要とする任意のシステム権限

  この方法の利点は、簡単であること、認証ツール (dbmlsync または SQL REMOTE のような) を介してユーザが接続された場合にのみ上記のシステム権限を使用できることです。この方法の欠点は、同期のために必要以上に強力な権限が SYS_RUN_REPLICATION_ROLE システムロールに付与されることです。いくつかのテーブルまたはプロシージャで必要なだけであっても、すべてのテーブルでの INSERT、UPDATE、DELETE、ALTER 権限と、すべてのプロシージャでの EXECUTE 権限が付与されます
- SYS_RUN_REPLICATION_ROLE システムロールから SYS_AUTH_DBA_ROLE を取り消し、SYS_RUN_REPLICATION_ROLE システムロールを持つユーザ拡張ロールを作成し、データベースの同期が許可されるユーザにユーザ拡張ロールを付与します。ユーザ拡張ロールに次のオブジェクトレベル権限を付与します。
  - 同期させるすべてのテーブルでの INSERT、UPDATE、DELETE、ALTER
  - すべてのフックプロシージャと、スクリプト化されたアップロードを定義する場合に使用するストアドプロシージャでの EXECUTE。
  - テーブル dbo.synchronize_results とテーブル dbo.synchronize_parameters での SELECT、INSERT、UPDATE、DELETE。これは、SQL SYNCHRONIZE 文を使用する場合にのみ必要です。
  - フックに含まれる文が必要とする任意の権限です。
  - スクリプト化されたアップロードを定義する場合に使用するストアドプロシージャ内の文が必要とする任意の権限です。

  この方法の利点は、ユーザに付与する権限を非常に細かく制御できる点にあります。ただし、ユーザは付与された権限を、ログインの方法に関係なく使用できます。ユーザは dbmlsync と SQL Remote によって確立される接続だけに制限されません。

例

次の例は、SYS_RUN_REPLICATION_ROLE システムロールから SYS_AUTH_DBA_ROLE 互換ロールを取り消して、同期に必要な権限を付与する方法を示します。

```
REVOKE ROLE SYS_AUTH_DBA_ROLE FROM SYS_RUN_REPLICATION_ROLE;
GRANT INSERT ANY TABLE TO SYS_RUN_REPLICATION_ROLE;
GRANT UPDATE ANY TABLE TO SYS_RUN_REPLICATION_ROLE;
GRANT DELETE ANY TABLE TO SYS_RUN_REPLICATION_ROLE;
GRANT ALTER ANY TABLE TO SYS_RUN_REPLICATION_ROLE;
GRANT EXECUTE ANY PROCEDURE TO SYS_RUN_REPLICATION_ROLE;
```

次の例は、SYS_RUN_REPLICATION_ROLE システムロールから SYS_AUTH_DBA_ROLE 互換ロールを取り消してから、テーブル **T1** とテーブル **T2** の同期および sp_hook_dbmlsync_begin フックの実行を行うのに十分な権限を持つ **SYNC_USER** と呼ばれるユーザ拡張ロールを作成する方法を示します。次に、ユーザ **user1** に SYNC_USER ロールを付与します。

```
REVOKE ROLE SYS_AUTH_DBA_ROLE FROM SYS_RUN_REPLICATION_ROLE;
// Create user-extended role SYNC_USER
CREATE USER SYNC_USER IDENTIFIED BY 'sql';
GRANT ROLE SYS_RUN_REPLICATION_ROLE TO SYNC_USER;
CREATE ROLE FOR USER SYNC_USER;
// Grant privileges on table T1 to SYNC_USER
GRANT INSERT ON T1 TO SYNC_USER;
GRANT UPDATE ON T1 TO SYNC_USER;
GRANT DELETE ON T1 TO SYNC_USER;
GRANT ALTER ON T1 TO SYNC_USER;
// Grant privileges on table T2 to SYNC_USER
GRANT INSERT ON T2 TO SYNC_USER;
GRANT UPDATE ON T2 TO SYNC_USER;
GRANT DELETE ON T2 TO SYNC_USER;
GRANT ALTER ON T2 TO SYNC_USER;
// Grant privileges on any hooks to SYNC_USER
GRANT EXECUTE ON dba.sp_hook_dbmlsync_begin TO SYNC_USER;
// Grant privileges on the synchronize_results and synchronize_parameters tables
// to SYNC_USER so that it can use the SYNCHRONIZE statement
GRANT SELECT ON dbo.synchronize_results TO SYNC_USER;
GRANT INSERT ON dbo.synchronize_results TO SYNC_USER;
GRANT UPDATE ON dbo.synchronize_results TO SYNC_USER;
GRANT DELETE ON dbo.synchronize_results TO SYNC_USER;
GRANT SELECT ON dbo.synchronize_parameters TO SYNC_USER;
GRANT INSERT ON dbo.synchronize_parameters TO SYNC_USER;
GRANT UPDATE ON dbo.synchronize_parameters TO SYNC_USER;
GRANT DELETE ON dbo.synchronize_parameters TO SYNC_USER;
// Grant SYNC_USER to user1 so that user1 can synchronize the database
GRANT ROLE SYNC_USER to user1;
```

1.6.1.5.2 dbmlsync 拡張オプション

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、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。

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
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。

このセクションの内容:

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]

1.6.1.5.2.1 dbmlsync ネットワークプロトコルオプション

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

関連情報

CommunicationType (ctp) 拡張オプション [161 ページ]
CommunicationAddress (adr) 拡張オプション [160 ページ]

1.6.1.5.3 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、古いトランザクション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、このパラメータを省略できます。

例

古いトランザクション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。

関連情報

進行オフセット [80 ページ]
スクリプト化されたアップロード [295 ページ]

1.6.1.5.4 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には、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。

関連情報

ConflictRetries (cr) 拡張オプション [162 ページ]
-d dbmlsync オプション [124 ページ]
LockTables (lt) 拡張オプション [172 ページ]

1.6.1.5.5 アプリケーションからの同期の開始

リモートユーザ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
- SQL SYNCHRONIZE 文
- 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 インタフェースを使用して 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 Anywhere17 ディレクトリの `SDK¥Include` サブディレクトリにある `dbtools.h` ヘッダファイルを含めることができます。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 インタフェー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奨めします。

関連情報

Dbmlsync API [107 ページ]
dbmlsync の DBTools インタフェース [290 ページ]

1.6.1.6 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、ユーザが停止するまで dbmlsync は実行を続けます。
- 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。

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。

ホバリング

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。

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。

このセクションの内容:

関連情報

PollingPeriod (pp) 拡張オプション [178 ページ]
HoverRescanThreshold (hrt) 拡張オプション [169 ページ]
-p dbmlsync オプション [137 ページ]
DisablePolling (p) 拡張オプション [164 ページ]
sp_hook_dbmlsync_log_rescan [256 ページ]

1.6.1.6.1 dbmlsync オプションを使用して設定されたスケジュール

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、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。例:

```
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。

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。例:

```
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。

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
Schedule (sch) 拡張オプション [179 ページ]
-eu dbmlsync オプション [130 ページ]
-is dbmlsync オプション [131 ページ]

1.6.1.6.2 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。

関連情報

sp_hook_dbmlsync_end [253 ページ]
sp_hook_dbmlsync_delay [240 ページ]

1.6.1.7 dbmlsync の同期のカスタマイズ

dbmlsync の同期をカスタマイズするには、イベントフック、dbmlsync プログラミングインタフェース、およびスクリプト化されたアップロードを使用します。

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フェースで使用できます。

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フェース

次のプログラミングインタフェースを使用して、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。これらのインタフェースは、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フェースです。この新しいプログラミングインタフェー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 インタフェース

dbmlsync 用の DBTools インタフェー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。

関連情報

SQL Anywhere クライアントのイベントフック [222 ページ]
Dbmlsync API [107 ページ]
dbmlsync の DBTools インタフェース [290 ページ]
スクリプト化されたアップロード [295 ページ]

1.6.1.8 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フェースです。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フェー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。

関連情報

Dbmlsync .NET API リファレンス [288 ページ]
Dbmlsync C++ API リファレンス [287 ページ]

1.6.1.9 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して Mobile Link アプリケーションを作成する場合、dbmlsync のメッセージログと SQL Anywhere トランザクションログの 2 種類のクライアントログファイルがあります。

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` という名前のメッセージ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。

メッセージ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。

- メッセージ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。
- メッセージ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、メッセージ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

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。

関連情報

トランザクションログファイル [101 ページ]
-o dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]
-os dbmlsync オプション [136 ページ]
-v dbmlsync オプション [152 ページ]

1.6.1.10 Mobile Link の Mac OS X での実行

Mobile Link サーバと 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。

コンテキスト

Mac OS X 上の Mobile Link 統合データベースを同期するには、ドライバマネージャとして SQL Anywhere ODBC ドライバを使用します。

手順

1. Mac OS X で Mobile Link サーバを開始します。
   a. 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/SQLAnywhere17` にあります。

   b. ファイル ▶ 新規 ▶ *Mobile Link* サーバ をクリックします。
   c. Mobile Link サーバを設定します。

      接続パラメータ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。

      `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 17 Demo;PWD=sql"
      ```

   d. 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 サーバの動作をさまざまな面から制御できます。Mobile Link サーバオプションの完全なリストについては、mlsrv17 構文を参照してください。

   e. 起動をクリックして、Mobile Link サーバを起動します。

      データベースサーバメッセージウィンドウが開き、サーバ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。

2. Mac OS X で dbmlsync を開始します。
   a. 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/SQLAnywhere17` にあります。

   b. ファイル ▶ 新規 ▶ *Mobile Link* クライアント をクリックします。

      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。

      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。

3. Mac OS X でサンプルデータベースを実行します。
   a. `sa_config` 設定スクリプトを準備します。
   b. 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 17 Demo;PWD=passwd"
      -c "UID=DBA;PWD=sql;DBF=/Applications/SQLAnywhere17/System/demo.db"
      ```

c. Mobile Link サーバを実行します。次に例を示します。

```
mlsrv17 -c "DSN=SQL Anywhere 17 Demo;PWD=passwd"
```

結果

Mobile Link サーバと SQL Anywhere Mobile Link クライアントが Mac OS X で実行されます。

関連情報

Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync) の構文 [110 ページ]

1.6.1.11 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 (`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。

1.6.2 Mobile Link SQL Anywhere クライアントユーティリティ (dbmlsync) の構文

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。dbmlsync を実行するには、SYS_RUN_REPLICATION_ROLE システム権限が必要です。

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。
-a	エラー時に再入力のプロンプトを表示しません。

オプション	説明
*-ap*parameter,...	認証パラメータを指定します。
*-ba*filename	ダウンロードファイルを適用します。
*-bc*filename	ダウンロードファイルを作成します。
*-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。
-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。
-bk	バックグラウンド同期を有効にします。
*-bkr*number	バックグラウンド同期が中断された後に dbmlsync で同期をリトライする回数を指定します。
*-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。
*-ci*size	dbmlsync キャッシュの初期サイズを設定します。
*-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。
*-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。
-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。
-dc	以前に失敗したダウンロードを続行します。
-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。
-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。このユーティリティオプションは、-x ユーティリティオプションとは一緒に使用できません。
*-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。
-ds	ダウンロード専用同期を実行します。
-e "keyword=value"...	拡張オプションを指定します。
-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。
-ep "keyword=value"	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。
-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。
-is	スケジュールを無視します。
-k	完了後、ウィンドウを閉じます。
-kpd	ダウンロードが失敗した場合は、必要な情報を保存し、ダウンロードを再度開始します。
-l	使用可能な拡張オプションをリストします。

オプション	説明
-*mn*password	新しい Mobile Link パスワードを指定します。
-*mp*password	Mobile Link パスワードを指定します。
-*n* "name,..."	同期パブリケーション名を指定します。
-*o*filename	このファイルに出力メッセージのログを取ります。
-*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)。
-*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。
-*p*	ログスキャンのポーリングを無効にします。
-*pc*[+ | -]	同期と同期の間で、Mobile Link サーバへのオープン接続を維持します。
-*pi*	Mobile Link に接続できるかどうかをテストします。
-*po*port	dbmlsync が受信するポートを指定します。
-*pp*number	ログスキャンのポーリング周期を設定します。
-*q*	最小化ウィンドウで実行します。
-*qc*	同期が終了したときに dbmlsync を停止します。
-*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。
-*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。
-*s*name	同期サブスクリプション名を指定します。
-*sc*	各同期の前にスキーマ情報を再ロードします。
-*sm*	dbmlsync がサーバモードで起動するようにします。
-*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。
-*ts*session-name(session-option=option-value[;...])	Mobile Link クライアントのトレースセッションを設定します。
-*tu*	トランザクション単位のアップロードを実行します。
-*u*ml_username	同期する Mobile Link ユーザを指定します。
-*ud*	UNIX 専用です。dbmlsync をデーモンとして実行します。
-*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。
-*uo*	アップロード専用同期を実行します。
-*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。
-*ux*	Solaris と Linux で、dbmlsync のメッセージウィンドウを開きます。
-*v*[levels]	冗長オペレーション。
-*wc*classname	ウィンドウクラス名を指定します。

オプション	説明
-x[size]	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。
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、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 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です。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。トランザクションログアーカイブファイルがアーカイブトランザクションログとは別のディレクトリにある場合、このパラメータを指定します。

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もあります。

dbmlsync の使用

dbmlsync の詳細については、起動同期に関するマニュアルを参照してください。

このセクションの内容:

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。

-bk dbmlsync オプション [121 ページ]
バックグラウンド同期を有効にします。

-bkr dbmlsync オプション [121 ページ]
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。

-c dbmlsync オプション [122 ページ]
リモートデータベースの接続パラメータを指定します。

-ci dbmlsync オプション [122 ページ]
dbmlsync キャッシュの初期サイズを設定します。

-cl dbmlsync オプション [123 ページ]
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。

-cm dbmlsync オプション [124 ページ]
dbmlsync キャッシュの最大サイズ上限値を設定します。

-d dbmlsync オプション [124 ページ]
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。

-dc dbmlsync オプション [125 ページ]
以前に失敗したダウンロードを再起動します。

-dl dbmlsync オプション [126 ページ]
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。

-do dbmlsync オプション [126 ページ]
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。

-drs dbmlsync オプション [127 ページ]
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。

-ds dbmlsync オプション [128 ページ]
ダウンロード専用同期を実行します。

-e dbmlsync オプション [128 ページ]
拡張オプションを指定します。

-eh dbmlsync オプション [129 ページ]
フック関数で発生したエラーを無視します。

-ek dbmlsync オプション [130 ページ]
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。

-ep dbmlsync オプション [130 ページ]
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。

-eu dbmlsync オプション [130 ページ]
拡張アップロードオプションを指定します。

-is dbmlsync オプション [131 ページ]
Schedule 拡張オプションを無視します。

-k dbmlsync オプション (旧式) [132 ページ]

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。

-kpd dbmlsync オプション [132 ページ]
失敗したダウンロードの再実行に必要な情報を保存します。

-l dbmlsync オプション [133 ページ]
使用可能な拡張オプションをリストします。

-mn dbmlsync オプション [133 ページ]
同期する Mobile Link ユーザの新しいパスワードを指定します。

-mp dbmlsync オプション [134 ページ]
同期する Mobile Link ユーザのパスワードを指定します。

-n dbmlsync オプション (旧式) [134 ページ]
同期させるパブリケーションを指定します。

-o dbmlsync オプション [136 ページ]
dbmlsync メッセージログファイルの名前を指定します。

-os dbmlsync オプション [136 ページ]
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メッセージログファイルの名前が変更されます)。

-ot dbmlsync オプション [137 ページ]
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。

-p dbmlsync オプション [137 ページ]
ログスキャンのポーリングを無効にします。

-pc dbmlsync オプション [138 ページ]
同期と同期の間で、Mobile Link サーバへの永続的な接続を維持します。

-pi dbmlsync オプション [139 ページ]
Mobile Link サーバを ping します。

-po dbmlsync オプション [140 ページ]
dbmlsync がサーバモードの場合、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。

-pp dbmlsync オプション [140 ページ]
ログスキャンの頻度を指定します。

-q dbmlsync オプション [141 ページ]
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。

-qc dbmlsync オプション [141 ページ]
同期が終了したときに dbmlsync を停止します。

-qi dbmlsync オプション [142 ページ]
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。

-r dbmlsync オプション [142 ページ]
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。

-s dbmlsync オプション [143 ページ]

同期するサブスクリプションを指定します。

-sc dbmlsync オプション [144 ページ]
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。

-sm dbmlsync オプション [144 ページ]
dbmlsync がサーバモードで起動するようにします。

-sp dbmlsync オプション [145 ページ]
-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。

-ts dbmlsync オプション [146 ページ]
Mobile Link クライアントのトレースセッションを設定します。

-tu dbmlsync オプション [147 ページ]
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。

-u dbmlsync オプション (旧式) [148 ページ]
Mobile Link ユーザ名を指定します。

-ud dbmlsync オプション [149 ページ]
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。

-ui dbmlsync オプション [150 ページ]
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。

-uo dbmlsync オプション [150 ページ]
同期がアップロードだけを含むように指定します。

-urc dbmlsync オプション [151 ページ]
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。

-ux dbmlsync オプション [151 ページ]
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。

-v dbmlsync オプション [152 ページ]
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。

-wc dbmlsync オプション [153 ページ]
ウィンドウクラス名を指定します。

-x dbmlsync オプション [154 ページ]
トランザクションログの名前を変更して再起動します。

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
同期の開始 [97 ページ]
トランザクションログファイル [101 ページ]
SQL Anywhere クライアントのイベントフック [222 ページ]

同期の開始 [97 ページ]
SQL Anywhere クライアントのイベントフック [222 ページ]
Dbmlsync API [107 ページ]
dbmlsync の DBTools インタフェース [290 ページ]

1.6.2.1 @data dbmlsync オプション

指定し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

このオプションを使用すると、指定された環境変数または設定ファイルから *dbmlsync* コマンドラインオプションを読み込むことができます。指定された名前の環境変数と設定ファイルが両方存在する場合は、環境変数が使用されます。

設定ファイル内の情報を保護する場合は、ファイル非表示ユーティリティ (dbfhide) を使用して、設定ファイルの内容をエンコードします。

1.6.2.2 -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 ...
```

1.6.2.3 -ap dbmlsync オプション

Mobile Link サーバで authenticate_parameters スクリプトと認証パラメータに渡されるパラメータを指定します。

構文

```
dbmlsync -ap "parameters,..." ...
```

備考

サーバで authenticate_parameters 接続スクリプトや認証パラメータ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"
```

パラメータは Mobile Link サーバに送信され、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。

関連情報

AuthParms 同期プロファイルオプション [200 ページ]

1.6.2.4 -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。任意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。

関連情報

-bc dbmlsync オプション [119 ページ]
-be dbmlsync オプション [119 ページ]
-bg dbmlsync オプション [120 ページ]
ApplyDnldFile 同期プロファイルオプション [200 ページ]

1.6.2.5 -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。

任意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。

関連情報

-ba dbmlsync オプション [118 ページ]
-be dbmlsync オプション [119 ページ]
-bg dbmlsync オプション [120 ページ]
CreateDnldFile 同期プロファイルオプション [204 ページ]

1.6.2.6 -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に渡されます。

関連情報

sp_hook_dbmlsync_validate_download_file [285 ページ]
-bc dbmlsync オプション [119 ページ]
-ba dbmlsync オプション [118 ページ]
DnldFileExtra 同期プロファイルオプション [204 ページ]

1.6.2.7 -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。

関連情報

-ba dbmlsync オプション [118 ページ]
-bc dbmlsync オプション [119 ページ]
UpdateGenNum 同期プロファイルオプション [218 ページ]

1.6.2.8 -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、データベースサーバ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。

関連情報

Mobile Link 同期プロファイル [195 ページ]
-bkr dbmlsync オプション [121 ページ]
Background 同期プロファイルオプション [201 ページ]

1.6.2.9 -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。

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。

関連情報

-bk dbmlsync オプション [121 ページ]
BackgroundRetry 同期プロファイルオプション [202 ページ]

1.6.2.10 -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を指定します。

構文

```
dbmlsync -c "connection-string" ...
```

備考

接続文字列では、SYS_REPLICATION_ADMIN_ROLE システムロールまたはこれと等価な権限を持つユーザとして 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てください。

keyword=value の形式で、複数のパラメータ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。

1.6.2.11 -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。

関連情報

-cm dbmlsync オプション [124 ページ]
-cl dbmlsync オプション [123 ページ]

1.6.2.12 -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。

関連情報

-cm dbmlsync オプション [124 ページ]
-ci dbmlsync オプション [122 ページ]

1.6.2.13 -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。

関連情報

-ci dbmlsync オプション [122 ページ]
-cl dbmlsync オプション [123 ページ]

1.6.2.14 -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。

関連情報

同期中の同時実行性 [102 ページ]
KillConnections 同期プロファイルオプション [210 ページ]

1.6.2.15 -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

このオプションは、失敗した同期において、-kpd dbmlsync オプションまたは KeepPartialDownload synchronization プロファイルオプションが指定されていた場合のみ使用できます。

dbmlsync でサーバ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、失敗したダウンロードにおいて -kpd dbmlsync が指定されていた場合、受信したダウンロードの一部は dbmlsync によって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。-dc オプションを指定すると、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。

-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。

関連情報

ContinueDownload (cd) 拡張オプション [163 ページ]
sp_hook_dbmlsync_end [253 ページ]
DownloadReadSize (drs) 拡張オプション [166 ページ]
ContinueDownload 同期プロファイルオプション [203 ページ]

1.6.2.16 -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メッセージ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。

1.6.2.17 -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 ...
```

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、-do 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。

1.6.2.18 -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。

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
```

関連情報

DownloadReadSize (drs) 拡張オプション [166 ページ]
ContinueDownload (cd) 拡張オプション [163 ページ]
sp_hook_dbmlsync_end [253 ページ]
-dc dbmlsync オプション [125 ページ]

1.6.2.19 -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、同期に必要なスクリプトに関するマニュアルを参照してください。

関連情報

ダウンロード専用のパブリケーション [85 ページ]
DownloadOnly (ds) 拡張オプション [165 ページ]
DownloadOnly 同期プロファイルオプション [205 ページ]

1.6.2.20 -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。

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"...
```

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
-l dbmlsync オプション [133 ページ]
-eu dbmlsync オプション [130 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]
ExtOpt 同期プロファイルオプション [207 ページ]

1.6.2.21 -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 ...
```

関連情報

IgnoreHookErrors 同期プロファイルオプション [208 ページ]

1.6.2.22 -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。

1.6.2.23 -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。

1.6.2.24 -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
```

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に関するマニュアルを参照してください。

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]

1.6.2.25 -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。

関連情報

同期のスケジュール [104 ページ]
IgnoreScheduling 同期プロファイルオプション [208 ページ]

1.6.2.26 -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 ...
```

関連情報

-qc dbmlsync オプション [141 ページ]

1.6.2.27 -kpd dbmlsync オプション

失敗したダウンロードの再実行に必要な情報を保存します。

構文

```
dbmlsync -kpd...
```

備考

このオプションまたは KeepPartialDownload 同期プロファイルオプションのいずれも使用しない場合、dbmlsync は同期が失敗する前に受信したダウンロード部分を保存しないため、失敗したダウンロードを再実行できません。

失敗したダウンロードを再実行するには、次のいずれかのメソッドを使用します。

- -dc dbmlsync オプション
- ContinueDownload 拡張オプション
- ContinueDownload 同期プロファイルオプション
- sp_hook_dbmlsync_end ストアドプロシージャ内の再実行パラメータ

このオプションまたは KeepPartialDownload 同期プロファイルオプションのいずれも使用しない場合、dbmlsync は同期が失敗する前に受信したダウンロード部分を保存しないため、失敗したダウンロードを再実行できません。

関連情報

-dc dbmlsync オプション [125 ページ]
ContinueDownload (cd) 拡張オプション [163 ページ]
ContinueDownload 同期プロファイルオプション [203 ページ]
sp_hook_dbmlsync_end [253 ページ]
KeepPartialDownload 同期プロファイルオプション [209 ページ]

1.6.2.28 -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 ...
```

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]

1.6.2.29 -mn dbmlsync オプション

同期する Mobile Link ユーザの新しいパスワードを指定します。

構文

```
dbmlsync -mn password ...
```

備考

Mobile Link ユーザのパスワードを変更します。

関連情報

同期システムの Mobile Link ユーザ [8 ページ]
MobiLinkPwd (mp) 拡張オプション [174 ページ]
NewMobiLinkPwd (mn) 拡張オプション [175 ページ]
-mp dbmlsync オプション [134 ページ]
NewMobiLinkPwd 同期プロファイルオプション [212 ページ]

1.6.2.30 -mp dbmlsync オプション

同期する Mobile Link ユーザのパスワードを指定します。

構文

```
dbmlsync -mp password ...
```

備考

Mobile Link ユーザ認証用のパスワードを指定します。

関連情報

同期システムの Mobile Link ユーザ [8 ページ]
MobiLinkPwd (mp) 拡張オプション [174 ページ]
NewMobiLinkPwd (mn) 拡張オプション [175 ページ]
-mn dbmlsync オプション [133 ページ]

1.6.2.31 -n dbmlsync オプション (旧式)

同期させるパブリケーションを指定します。

注記

このオプションは廃止される予定です。今後は -s dbmlsync オプションを使用してください。

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。

<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。

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います。
  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します。
  `-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 分が十分な遅延です。

関連情報

sp_hook_dbmlsync_delay [240 ページ]
パブリケーション同期プロファイルオプション (旧式) [214 ページ]
-s dbmlsync オプション [143 ページ]
-u dbmlsync オプション (旧式) [148 ページ]

1.6.2.32 -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 dbmlsync メッセージ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。

関連情報

-os dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]

1.6.2.33 -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メッセージログファイル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、古いメッセージ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。

関連情報

-o dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]

1.6.2.34 -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。

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。

関連情報

-o dbmlsync オプション [136 ページ]
-os dbmlsync オプション [136 ページ]

1.6.2.35 -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。

関連情報

DisablePolling (p) 拡張オプション [164 ページ]
PollingPeriod (pp) 拡張オプション [178 ページ]
-pp dbmlsync オプション [140 ページ]

1.6.2.36 -pc dbmlsync オプション

同期と同期の間で、Mobile Link サーバへの永続的な接続を維持します。

構文

```
dbmlsync -pc[ + | - ] ...
```

備考

-pc+ を使用すると、dbmlsync は通常どおり Mobile Link サーバに接続しますが、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。
- 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。

-pc+ 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、サーバへの接続確立コストが高くなっている場合です。

-pc- を使用した場合は、各同期が終了すると接続が閉じて、次の同期が開始すると接続が再び開きます。デフォルトでは、永続的な接続は維持されません。

関連情報

timeout Mobile Link クライアントネットワークプロトコルオプション [60 ページ]

1.6.2.37 -pi dbmlsync オプション

Mobile Link サーバを 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 サーバへの接続に必要な情報を取り出し、サーバに接続し、指定した Mobile Link ユーザを認証します。Mobile Link サーバは、ping を受信すると、統合データベースに接続し、ユーザを認証し、ユーザ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 サーバがユーザを 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

関連情報

Ping 同期プロファイルオプション [213 ページ]

1.6.2.38 -po dbmlsync オプション

dbmlsync がサーバモードの場合、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。

関連情報

-sm dbmlsync オプション [144 ページ]

1.6.2.39 -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。

関連情報

PollingPeriod (pp) 拡張オプション [178 ページ]
DisablePolling (p) 拡張オプション [164 ページ]
-p dbmlsync オプション [137 ページ]

1.6.2.40 -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 ...
```

1.6.2.41 -qc dbmlsync オプション

同期が終了したときに dbmlsync を停止します。

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。

関連情報

-o dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]

1.6.2.42 -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。

関連情報

-o dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]

1.6.2.43 -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

-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。

関連情報

進行オフセット [80 ページ]
RemoteProgressGreater 同期プロファイルオプション [215 ページ]
RemoteProgressLess 同期プロファイルオプション [216 ページ]

1.6.2.44 -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います。
  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います。
  `-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 分が十分な遅延です。

関連情報

sp_hook_dbmlsync_delay [240 ページ]
Subscription 同期プロファイルオプション [217 ページ]

1.6.2.45 -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。

1.6.2.46 -sm dbmlsync オプション

dbmlsync がサーバモードで起動するようにします。

構文

```
dbmlsync -sm ...
```

備考

サーバモードの場合、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。

関連情報

-po dbmlsync オプション [140 ページ]

1.6.2.47 -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。

関連情報

Mobile Link 同期プロファイル [195 ページ]

1.6.2.48 -ts dbmlsync オプション

Mobile Link クライアントのトレースセッションを設定します。

構文

```
dbmlsync -ts session-name(session-option=option-value[;...])
```

セッション名は *logging* にする必要があります。

セッションオプション	オプション値
events	システムトレースイベントのカンマで区切られたリストサポートされるイベントは、Info、Warning、および Error です。
targets	`target-type(target-option=value[;...])`。ここで、`target-type` には *file* のみを指定できます。

ターゲットオプションは、名前と値のペアとして指定されます。ターゲットファイルには、次のオプションが用意されていることがあります。

ターゲットオプション名	予期される値	説明
filename_prefix	String	パス付きまたはパスなしの ETD ファイル名プレフィクスすべての ETD ファイルには、`.etd` という拡張子が付きます。このパラメータは必須です。
max_size	Integer	ファイルの最大サイズ (バイト単位)。デフォルトは 0 で、これはファイルサイズに上限がないことを意味し、ディスク領域が許す限り大きくなっていきます。指定したサイズに達すると、新しいファイルが開始されます。
num_files	Integer	イベントトレース情報が書き込まれるファイル数で、max_size が設定されている場合にのみ使用されます。すべてのファイルが指定した最大サイズに達すると、Mobile Link クライアントは最も古いファイルの上書きを開始します。
flush_on_write	yes、true、no、false	記録されるイベントが発生するたびにディスクバッファをフラッシュするかどうかを制御する値。値には、yes、true、no、false を設定できます。デフォルトは false です。このパラメータをオンにすると、多くのトレースイベントが記録される場合、Mobile Link クライアントのパフォーマンスが低下することがあります。
compressed	yes、true、no、false	ディスク領域を節約するための ETD ファイルの圧縮を制御する値。デフォルトは false です。

備考

オプションの *-ts logging* 部分の後で指定するすべての情報は、スペースなしで指定する必要があります。

ETD ファイルに含まれているデータを取得するには、Event Trace Data (ETD) ファイル管理ユーティリティ (dbmanageetd) か、sp_read_etd システムプロシージャのいずれかを使用します。

例

次に、-ts オプションの例を示します。

```
-ts
logging(events=Info,warning,Error;targets=file(filename_prefix=mls_etd;max_size=10
000000;num_files=10;flush_on_write=true))
```

1.6.2.49 -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。

-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。

関連情報

TransactionalUpload 同期プロファイルオプション [217 ページ]

1.6.2.50 -u dbmlsync オプション (旧式)

Mobile Link ユーザ名を指定します。

注記

このオプションは廃止される予定です。今後は -s dbmlsync オプションを使用してください。

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` は、同期する Mobile Link ユーザに置き換えます。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。

`pub_name` は、同期されるパブリケーションの名前に置き換えます。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。

構文

```
dbmlsync -u ml_username ...
```

備考

dbmlsync コマンドラインでユーザを 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。

関連情報

MLUser 同期プロファイルオプション (旧式) [212 ページ]
-n dbmlsync オプション (旧式) [134 ページ]
-s dbmlsync オプション [143 ページ]

1.6.2.51 -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。

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 値を設定します。

関連情報

-o dbmlsync オプション [136 ページ]

-ot dbmlsync オプション [137 ページ]

1.6.2.52 -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。

1.6.2.53 -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、同期に必要なスクリプトに関するマニュアルを参照してください。

関連情報

DownloadOnly (ds) 拡張オプション [165 ページ]
UploadOnly (uo) 拡張オプション [186 ページ]
UploadOnly 同期プロファイルオプション [219 ページ]

1.6.2.54 -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。

関連情報

UploadRowCnt 同期プロファイルオプション [220 ページ]

1.6.2.55 -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。

関連情報

-q dbmlsync オプション [141 ページ]

1.6.2.56 -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します。

c

ログ内の接続文字列を公開します。

p

ログ内の Mobile Link パスワードを公開します。

n

アップロードとダウンロードされた行数のログを取ります。

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ります。

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ります。

s

フックスクリプトに関連するメッセージのログを取ります。

u

アップロードに関連する情報のログを取ります。

-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かかわらず、動作は同じになるはずです。

関連情報

Verbose (v) 拡張オプション [187 ページ]
VerboseHooks (vs) 拡張オプション [188 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
-o dbmlsync オプション [136 ページ]
-ot dbmlsync オプション [137 ページ]
Verbosity 同期プロファイルオプション [220 ページ]

1.6.2.57 -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。

このオプションが適用されるのは、Windows だけです。

例

```
dbmlsync -wc dbmlsync_$message_end...
```

関連情報

同期のスケジュール [104 ページ]
Schedule (sch) 拡張オプション [179 ページ]

1.6.2.58 -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 ] ] ...
```

備考

サイズ値は常に -x オプションと一緒に指定する必要があります。具体的には、*-x 0* を指定することをお奨めします。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を使用する方法があります。

1.6.3 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、SQL 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)

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。
CommunicationAddress=`protocol-option`; ...	Mobile Link サーバ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。
CommunicationType=`network-protocol`; ...	Mobile Link サーバ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。
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。
ContinueDownload={ *ON* | *OFF* }	以前に失敗したダウンロードを再起動します。
DisablePolling={*ON* | *OFF*}; ...	ログスキャンの自動ポーリングを無効にします。
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。
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。
ErrorLogSendLimit=`number`[*K* | *M*]; ...	同期エラーが発生した場合、dbmlsync からサーバ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。
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が起動されるように指定します。
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。
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。
IgnoreScheduling={ *ON* | *OFF* }	Schedule 拡張オプションを無視するように指定します。

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。
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。
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。
MobiLinkPwd=password; ...	Mobile Link パスワードを指定します。
NewMobiLinkPwd=new-password; ...	新しい Mobile Link パスワードを指定します。
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。
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。
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。
Schedule=schedule; ...	同期のスケジュールを指定します。
ScriptVersion=version-name; ...	スクリプトバージョンを指定します。
SendDownloadAck={ *ON* | *OFF* }	クライアントからサーバにダウンロード確認が送信されるように指定します。
SendTriggers={ *ON* | *OFF* }	アップロード時にトリガの動作が送信されるように指定します。
TableOrder=tables; ...	アップロードでのテーブルの順序を指定します。
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。
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。
Verbose={ *ON* | *OFF* }	完全冗長を指定します。
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。
VerboseMin={ *ON* | *OFF* }	少量の情報のログを取るように指定します。
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。
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。
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。
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。

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。

```
dbmlsync -l
```

このセクションの内容:

Schedule 拡張オプションを無視するように指定します。

Increment (inc) 拡張オプション [171 ページ]
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。

LockTables (lt) 拡張オプション [172 ページ]
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。

MirrorLogDirectory (mld) 拡張オプション [173 ページ]
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。

MobiLinkPwd (mp) 拡張オプション [174 ページ]
Mobile Link パスワードを指定します。

NewMobiLinkPwd (mn) 拡張オプション [175 ページ]
新しい Mobile Link パスワードを指定します。

NoSyncOnStartup (nss) 拡張オプション [176 ページ]
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。

OfflineDirectory (dir) 拡張オプション [177 ページ]
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。

PollingPeriod (pp) 拡張オプション [178 ページ]
ログスキャンのポーリング周期を指定します。

Schedule (sch) 拡張オプション [179 ページ]
同期のスケジュールを指定します。

ScriptVersion (sv) 拡張オプション [181 ページ]
スクリプトバージョンを指定します。

SendDownloadAck (sa) 拡張オプション [182 ページ]
クライアントからサーバにダウンロード確認が送信されるように指定します。

SendTriggers (st) 拡張オプション [183 ページ]
アップロード時にトリガの動作が送信されるように指定します。

TableOrder (tor) 拡張オプション [184 ページ]
アップロードでのテーブルの順序を指定します。

TableOrderChecking (toc) 拡張オプション [185 ページ]
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。

UploadOnly (uo) 拡張オプション [186 ページ]
同期がアップロードだけを含むように指定します。

Verbose (v) 拡張オプション [187 ページ]
完全冗長を指定します。

VerboseHooks (vs) 拡張オプション [188 ページ]
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。

VerboseMin (vm) 拡張オプション [189 ページ]
少量の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 [190 ページ]

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。

VerboseRowCounts (vn) 拡張オプション [191 ページ]
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。

VerboseRowValues (vr) 拡張オプション [192 ページ]
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。

VerboseUpload (vu) 拡張オプション [193 ページ]
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。

関連情報

dbmlsync 拡張オプション [100 ページ]
-e dbmlsync オプション [128 ページ]
-eu dbmlsync オプション [130 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]

1.6.3.1 オプション競合の dbmlsync による解決方法

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 SQL 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 SQL Central で Mobile Link ユーザに対して指定されているオプション
7. SQL 文または SQL 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。

> **注記**
>
> 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。

1.6.3.2 BufferDownload (bd) 拡張オプション

Mobile Link サーバ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 サーバへの負荷が低下し、サーバ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。

1.6.3.3 CommunicationAddress (adr) 拡張オプション

Mobile Link サーバ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

Mobile Link ユーザ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。

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'
```

関連情報

Mobile Link クライアントネットワークプロトコルオプション [23 ページ]
CommunicationType (ctp) 拡張オプション [161 ページ]

1.6.3.4 CommunicationType (ctp) 拡張オプション

Mobile Link サーバ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` は、*tcpip*、*tls*、*http*、または *https* のいずれかです。デフォルトは *tcpip* です。

Mobile Link ユーザ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。

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'
```

関連情報

CommunicationAddress (adr) 拡張オプション [160 ページ]

1.6.3.5 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 (リトライは無限に続行されます)。

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 "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';
```

1.6.3.6 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

このオプションは、失敗した同期において、-kpd dbmlsync オプションまたは KeepPartialDownload synchronization プロファイルオプションが指定されていた場合のみ使用できます。

dbmlsync でサーバ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、失敗したダウンロードにおいて -kpd dbmlsync が指定されていた場合、受信したダウンロードの一部は dbmlsync によって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。

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';
```

関連情報

-kpd dbmlsync オプション [132 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]
-dc dbmlsync オプション [125 ページ]
sp_hook_dbmlsync_end [253 ページ]

1.6.3.7 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。

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。

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';
```

関連情報

PollingPeriod (pp) 拡張オプション [178 ページ]
-p dbmlsync オプション [137 ページ]
-pp dbmlsync オプション [140 ページ]

1.6.3.8 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。

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)。

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';
```

関連情報

ダウンロード専用のパブリケーション [85 ページ]
-ds dbmlsync オプション [128 ページ]

1.6.3.9 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。

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';
```

関連情報

-drs dbmlsync オプション [127 ページ]
ContinueDownload (cd) 拡張オプション [163 ページ]
sp_hook_dbmlsync_end [253 ページ]
-dc dbmlsync オプション [125 ページ]

1.6.3.10 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
-e dbmlsync オプション [128 ページ]
-eu dbmlsync オプション [130 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseHooks (vs) 拡張オプション [188 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.11 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';
```

1.6.3.12 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を実行することです。

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 "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';
```

関連情報

sp_hook_dbmlsync_log_rescan [256 ページ]

1.6.3.13 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';
```

1.6.3.14 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。

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。

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';
```

関連情報

同期のスケジュール [104 ページ]
Schedule (sch) 拡張オプション [179 ページ]

1.6.3.15 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';
```

1.6.3.16 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。すべてのプラットフォームにおいて、ON は SHARE と同じです。

デフォルトは OFF です。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。

同期テーブルが排他モードでロックされているとき、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。

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';
```

関連情報

同期中の同時実行性 [102 ページ]
SQL Anywhere クライアントのイベントフック [222 ページ]

1.6.3.17 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とは異なるコンピュータで実行されている

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。

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';
```

1.6.3.18 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 *-emn* で変更できます。

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 "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。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION mp='password';
```

関連情報

NewMobiLinkPwd (mn) 拡張オプション [175 ページ]
-mn dbmlsync オプション [133 ページ]
-mp dbmlsync オプション [134 ページ]

1.6.3.19 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。

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'
```

関連情報

MobiLinkPwd (mp) 拡張オプション [174 ページ]
-mn dbmlsync オプション [133 ページ]
-mp dbmlsync オプション [134 ページ]

1.6.3.20 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。

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';
```

関連情報

同期のスケジュール [104 ページ]
Schedule (sch) 拡張オプション [179 ページ]

1.6.3.21 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指定できます。

名前の変更された (オフライン) トランザクションログが、オンライントランザクションログと同じディレクトリにある場合、このオプションを使用しないでください。この設定は、データベースやトランザクションログファイルを、OfflineDirectory 設定を変更することなく、クラウド内で、または dblog -t を使用して移動することを可能にするなど、より高い柔軟性を提供します。しかしながら、データベースサーバはページを取得するためにより多くの動作をする必要があるため、パフォーマンスが多少低下します。したがってパフォーマンスが重要な場合は、OfflineDirectory の使用が配備にとって最適であることもあります。

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 "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';
```

1.6.3.22 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。

このオプションは dbmlsync *-pp* 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 "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';
```

関連情報

DisablePolling (p) 拡張オプション [164 ページ]
-pp dbmlsync オプション [140 ページ]
-p dbmlsync オプション [137 ページ]
sp_hook_dbmlsync_delay [240 ページ]
Schedule (sch) 拡張オプション [179 ページ]

1.6.3.23 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。

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。

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';
```

関連情報

同期のスケジュール [104 ページ]
NoSyncOnStartup (nss) 拡張オプション [176 ページ]
-wc dbmlsync オプション [153 ページ]
IgnoreScheduling (isc) 拡張オプション [171 ページ]
-is dbmlsync オプション [131 ページ]

1.6.3.24 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奨めします。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。

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';
```

1.6.3.25 SendDownloadAck (sa) 拡張オプション

クライアントからサーバ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。

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';
```

1.6.3.26 SendTriggers (st) 拡張オプション

アップロード時にトリガ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。

注記

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での操作が同期されるようにするには、そのトリガでの操作が、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。

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';
```

1.6.3.27 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。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。

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;
```

関連情報

TableOrderChecking (toc) 拡張オプション [185 ページ]

1.6.3.28 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 }; ...

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。

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';
```

関連情報

TableOrder (tor) 拡張オプション [184 ページ]

1.6.3.29 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、同期に必要なスクリプトに関するマニュアルを参照してください。

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 "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';
```

関連情報

DownloadOnly (ds) 拡張オプション [165 ページ]

1.6.3.30 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
VerboseHooks (vs) 拡張オプション [188 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.31 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。

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';
```

関連情報

SQL Anywhere クライアントのイベントフック [222 ページ]
-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.32 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.33 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.34 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。

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。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION vn='on';
```

関連情報

-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.35 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseUpload (vu) 拡張オプション [193 ページ]

1.6.3.36 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。

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';
```

関連情報

-v dbmlsync オプション [152 ページ]
Verbose (v) 拡張オプション [187 ページ]
Verbose (v) 拡張オプション [187 ページ]
VerboseMin (vm) 拡張オプション [189 ページ]
VerboseOptions (vo) 拡張オプション [190 ページ]
VerboseRowCounts (vn) 拡張オプション [191 ページ]
VerboseRowValues (vr) 拡張オプション [192 ページ]

1.6.4 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]
- ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]
- ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]
- ALTER SYNCHRONIZATION USER 文 [Mobile Link]
- CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]
- CREATE SYNCHRONIZATION PROFILE 文 [Mobile Link]
- CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]
- CREATE SYNCHRONIZATION USER 文 [Mobile Link]
- DROP PUBLICATION 文 [Mobile Link] [SQL Remote]
- DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]
- DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]

- DROP SYNCHRONIZATION USER 文 [Mobile Link]
- GRANT ROLE 文
- REVOKE ROLE 文
- START SYNCHRONIZATION DELETE 文 [Mobile Link]
- START SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]
- STOP SYNCHRONIZATION DELETE 文 [Mobile Link]
- STOP SYNCHRONIZATION SCHEMA CHANGE 文 [Mobile Link]
- SYNCHRONIZE 文 [Mobile Link]

Ultra Light クライアント

Ultra Light SQL 文の詳細については、Ultra Light マニュアルを参照してください。

1.6.5 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]
- ALTER SYNCHRONIZATION PROFILE 文 [Mobile Link]
- DROP SYNCHRONIZATION PROFILE 文 [Mobile Link]

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。他のインタフェースには、追加オプションを指定するための特定のパラメータまたは句があります。

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にパラメータを入力します。
ApplyDnldFile	*ba*	文字列	ダウンロードファイルを適用します。
バックグラウンド	*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。

オプション名	省略名	指定可能な値	説明
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。
ContinueDownload	*dc*	*Boolean*	以前に失敗したダウンロードを再起動します。
CreateDnldFile	*bc*	文字列	ダウンロードファイルを作成します。
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。
DownloadOnly	*ds*	*Boolean*	ダウンロード専用同期を実行します。
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。
ExtOpt	*e*	文字列	拡張オプションを指定します。
IgnoreHookErrors	*eh*	*Boolean*	フック関数で発生したエラーを無視します。
IgnoreScheduling	*is*	*Boolean*	スケジュールの指示を無視して、直ちに同期を行います。
KeepPartialDownload	*kpd*	*Boolean*	ダウンロードが失敗した場合は、必要な情報を保存し、失敗したダウンロードを再度開始します。
KillConnections	*d*	*Boolean*	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。
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。
MobiLinkPwd	*mp*	文字列	Mobile Link ユーザのパスワードを指定します。
MLUser	*u*	文字列	Mobile Link ユーザ名を指定します。
NewMobiLinkPwd	*mn*	文字列	Mobile Link ユーザ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。
Ping	*pi*	*Boolean*	Mobile Link サーバに対して ping を実行し、クライアントと Mobile Link 間の通信を確認します。

オプション名	省略名	指定可能な値	説明
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。
RemoteProgressGreater	*ra*	*Boolean*	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。
RemoteProgressLess	*rb*	*Boolean*	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。
サブスクリプション	*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。
TransactionalUpload	*tu*	*Boolean*	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。
UpdateGenNum	*bg*	*Boolean*	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。
UploadOnly	*uo*	*Boolean*	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。
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。

オプション名	省略名	指定可能な値	説明
冗長性		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。冗長性のある MobiLink SQL Anywhere クライアント拡張オプションと同様。 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。

このセクションの内容:

ExtOpt 同期プロファイルオプション [207 ページ]
拡張オプションを指定します。

IgnoreHookErrors 同期プロファイルオプション [208 ページ]
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。

IgnoreScheduling 同期プロファイルオプション [208 ページ]
Schedule 拡張オプションを無視して、直ちに同期を行います。

KeepPartialDownload 同期プロファイルオプション [209 ページ]
失敗したダウンロードの再実行に必要な情報を保存します。

KillConnections 同期プロファイルオプション [210 ページ]
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。

LogRenameSize 同期プロファイルオプション [210 ページ]
トランザクションログの名前を変更して再起動します。

MobiLinkPwd 同期プロファイルオプション [211 ページ]
Mobile Link ユーザのパスワードを指定します。

MLUser 同期プロファイルオプション (旧式) [212 ページ]
Mobile Link ユーザ名を指定します。

NewMobiLinkPwd 同期プロファイルオプション [212 ページ]
Mobile Link ユーザ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。

Ping 同期プロファイルオプション [213 ページ]
Mobile Link サーバを ping します。

パブリケーション同期プロファイルオプション (旧式) [214 ページ]
同期させるパブリケーションを指定します。

RemoteProgressGreater 同期プロファイルオプション [215 ページ]
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。

RemoteProgressLess 同期プロファイルオプション [216 ページ]
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。

Subscription 同期プロファイルオプション [217 ページ]
同期させるサブスクリプションを指定します。

TransactionalUpload 同期プロファイルオプション [217 ページ]
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。

UpdateGenNum 同期プロファイルオプション [218 ページ]
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。

UploadOnly 同期プロファイルオプション [219 ページ]
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。

UploadRowCnt 同期プロファイルオプション [220 ページ]

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。

Verbosity 同期プロファイルオプション [220 ページ]
dbmlsync の冗長性を制御します。

1.6.5.1 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にパラメータを入力します。

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を使用するときに使用します。

パラメータは Mobile Link サーバに送信され、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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'AuthParms=p1,p2,p3'
```

関連情報

-ap dbmlsync オプション [117 ページ]

1.6.5.2 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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'ApplyDnldFile=filename'
```

関連情報

-ba dbmlsync オプション [118 ページ]

1.6.5.3 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、データベースサーバ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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'Background=on'
```

関連情報

-bk dbmlsync オプション [121 ページ]

1.6.5.4 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。

この値を -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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'BackgroundRetry=4'
```

関連情報

-bkr dbmlsync オプション [121 ページ]

1.6.5.5 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

このオプションは、失敗した同期において、-kpd dbmlsync オプションまたは KeepPartialDownload synchronization プロファイルオプションが指定されていた場合のみ使用できます。

dbmlsync でサーバ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、失敗したダウンロードにおいて -kpd dbmlsync が指定されていた場合、受信したダウンロードの一部は dbmlsync によって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。ContinueDownload=on オプションを指定すると、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。

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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'ContinueDownload=on'
```

関連情報

-kpd dbmlsync オプション [132 ページ]
-dc dbmlsync オプション [125 ページ]
sp_hook_dbmlsync_end [253 ページ]
ContinueDownload (cd) 拡張オプション [163 ページ]

1.6.5.6 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。

任意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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'CreateDnldFile=dnldl.df'
```

関連情報

-bc dbmlsync オプション [119 ページ]

1.6.5.7 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'DnldFileExtra=val1,val2,val3'
```

関連情報

-be dbmlsync オプション [119 ページ]
sp_hook_dbmlsync_validate_download_file [285 ページ]

1.6.5.8 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'DownloadOnly=on'
```

関連情報

-ds dbmlsync オプション [128 ページ]
DownloadOnly (ds) 拡張オプション [165 ページ]

1.6.5.9 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。

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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'DownloadReadSize=100'
```

関連情報

-drs dbmlsync オプション [127 ページ]
DownloadReadSize (drs) 拡張オプション [166 ページ]

1.6.5.10 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。

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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile
'ExtOpt={lt=exclusive;tableorder=t1,t2,t3}'
```

関連情報

Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
-e dbmlsync オプション [128 ページ]

1.6.5.11 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*}

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'IgnoreHookErrors=on'
```

関連情報

-eh dbmlsync オプション [129 ページ]

1.6.5.12 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'IgnoreScheduling=on'
```

関連情報

同期のスケジュール [104 ページ]
-is dbmlsync オプション [131 ページ]

1.6.5.13 KeepPartialDownload 同期プロファイルオプション

失敗したダウンロードの再実行に必要な情報を保存します。

構文

kpd={ *TRUE* | *FALSE* }

KeepPartialDownload={ *TRUE* | *FALSE* }

備考

このオプションには省略形と長形式があり、kpd または KeepPartialDownload を使用できます。

このオプションまたは -kpd dbmlsync オプションのいずれも使用しない場合、dbmlsync は同期が失敗する前に受信したダウンロード部分を保存しないため、失敗したダウンロードを再実行できません。

失敗したダウンロードを再実行するには、次のいずれかのメソッドを使用します。

- -dc dbmlsync オプション
- ContinueDownload 拡張オプション
- ContinueDownload 同期プロファイルオプション
- sp_hook_dbmlsync_end ストアドプロシージャ内の再実行パラメータ

例

次の例は、KeepPartialDownload 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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'KeepPartialDownload=TRUE';
```

関連情報

-dc dbmlsync オプション [125 ページ]
ContinueDownload (cd) 拡張オプション [163 ページ]
ContinueDownload 同期プロファイルオプション [203 ページ]

sp_hook_dbmlsync_end [253 ページ]
-kpd dbmlsync オプション [132 ページ]

1.6.5.14 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'KillConnections=on'
```

関連情報

同期中の同時実行性 [102 ページ]
-d dbmlsync オプション [124 ページ]

1.6.5.15 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'LogRenameSize=512k'
```

関連情報

-x dbmlsync オプション [154 ページ]

1.6.5.16 MobiLinkPwd 同期プロファイルオプション

Mobile Link ユーザのパスワードを指定します。

構文

```
mp=password
```

```
MobiLinkPwd=password
```

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'MobiLinkPwd=mypassword'
```

関連情報

-mp dbmlsync オプション [134 ページ]

MobiLinkPwd (mp) 拡張オプション [174 ページ]
NewMobiLinkPwd (mn) 拡張オプション [175 ページ]
-mn dbmlsync オプション [133 ページ]

1.6.5.17 MLUser 同期プロファイルオプション (旧式)

Mobile Link ユーザ名を指定します。

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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'MLUser=my_user_name'
```

関連情報

同期システムの Mobile Link ユーザ [8 ページ]
Subscription 同期プロファイルオプション [217 ページ]
-u dbmlsync オプション (旧式) [148 ページ]

1.6.5.18 NewMobiLinkPwd 同期プロファイルオプション

Mobile Link ユーザ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

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'NewMobiLinkPwd=new_password'
```

関連情報

同期システムの Mobile Link ユーザ [8 ページ]
-mp dbmlsync オプション [134 ページ]
-mn dbmlsync オプション [133 ページ]

1.6.5.19 Ping 同期プロファイルオプション

Mobile Link サーバを 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 サーバへの接続に必要な情報を取り出し、Mobile Link サーバに接続し、指定した Mobile Link ユーザを認証します。

Mobile Link サーバは、ping を受信すると、統合データベースに接続し、ユーザを認証し、ユーザ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 サーバがユーザを 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。

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。

```
CREATE SYNCHRONIZATION PROFILE myprofile 'Ping=on'
```

関連情報

-pi dbmlsync オプション [139 ページ]

1.6.5.20 パブリケーション同期プロファイルオプション (旧式)

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。

注記

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 オプションのいずれかを使用します。

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。

<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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'Publication=overnight'
```

関連情報

Subscription 同期プロファイルオプション [217 ページ]
-s dbmlsync オプション [143 ページ]
-u dbmlsync オプション (旧式) [148 ページ]
-n dbmlsync オプション (旧式) [134 ページ]

1.6.5.21 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*}

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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'RemoteProgressGreater=on'
```

関連情報

-r dbmlsync オプション [142 ページ]

1.6.5.22 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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'RemoteProgressLess=on'
```

関連情報

-r dbmlsync オプション [142 ページ]

1.6.5.23 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=subname, ...
```

```
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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'Subscription=mySubscription'
```

関連情報

-s dbmlsync オプション [143 ページ]

1.6.5.24 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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'TransactionalUpload=on'
```

関連情報

-tu dbmlsync オプション [147 ページ]

1.6.5.25 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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'UpdateGenNum=on'
```

関連情報

-bg dbmlsync オプション [120 ページ]

1.6.5.26 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、必要な同期スクリプトに関するマニュアルを参照してください。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'UploadOnly=on'
```

関連情報

-uo dbmlsync オプション [150 ページ]
UploadOnly (uo) 拡張オプション [186 ページ]

1.6.5.27 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。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'UploadRowCnt=100'
```

関連情報

-urc dbmlsync オプション [151 ページ]

1.6.5.28 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*}, ...

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します。

HIGH

可能な最高の冗長レベルを生成します。

CONNECT_STR

ログ内の接続文字列を公開します。

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ります。

OPTIONS

同期の指定に使用するオプションのログを取ります。

ML_PASSWORD

ログ内の Mobile Link パスワードを公開します。

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ります。

HOOK

フックスクリプトに関連するメッセージのログを取ります。

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かかわらず、動作は同じになるはずです。

例

```
CREATE SYNCHRONIZATION PROFILE myprofile 'Verbosity=OPTIONS, ML_PASSWORD'
```

関連情報

-v dbmlsync オプション [152 ページ]

1.6.6 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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

dbmlsync インタフェース

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フェースとも一緒に使用できます。これには、Dbmlsync API と Dbmlsync 用の DBTools インタフェースが含まれます。

このセクションの内容:

イベントフックプロシージャのを設計と使用に関しては、いくつかの考慮事項があります。

sp_hook_dbmlsync_abort [230 ページ]
このストアドプロシージャは、同期処理をキャンセルする場合に使用します。

sp_hook_dbmlsync_all_error [232 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_begin [235 ページ]
このストアドプロシージャを使用して、同期処理の開始時にカスタムアクションを追加します。

sp_hook_dbmlsync_communication_error [237 ページ]
このストアドプロシージャは、通信エラーを処理する場合に使用します。

sp_hook_dbmlsync_delay [240 ページ]
このストアドプロシージャを使用して、同期が行われるタイミングを制御します。

sp_hook_dbmlsync_download_begin [242 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_download_end [244 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_download_log_ri_violation [246 ページ]
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。

sp_hook_dbmlsync_download_ri_violation [248 ページ]
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。

sp_hook_dbmlsync_download_table_begin [250 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_download_table_end [251 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_end [253 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_log_rescan [256 ページ]
このストアドプロシージャを使用して、再スキャンがいつ必要かプログラム的に決定できます。

sp_hook_dbmlsync_logscan_begin [258 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_logscan_end [260 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_misc_error [261 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_ml_connect_failed [265 ページ]
このストアドプロシージャを使用して、Mobile Link サーバ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。

sp_hook_dbmlsync_process_exit_code [268 ページ]
このストアドプロシージャを使用して、終了コードを管理します。

sp_hook_dbmlsync_schema_upgrade [270 ページ]
このストアドプロシージャを使用して、スキーマを修正する SQL スクリプトを実行します。

sp_hook_dbmlsync_set_extended_options [273 ページ]
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を使用します。

sp_hook_dbmlsync_set_ml_connect_info [274 ページ]
このストアドプロシージャを使用して、Mobile Link サーバ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。

sp_hook_dbmlsync_set_upload_end_progress [276 ページ]
このストアドプロシージャは、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。

sp_hook_dbmlsync_sql_error [278 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_upload_begin [280 ページ]
このストアドプロシージャを使用して、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。

sp_hook_dbmlsync_upload_end [282 ページ]
Mobile Link サーバ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を使用してカスタムアクションを追加します。

sp_hook_dbmlsync_validate_download_file [285 ページ]
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。

関連情報

dbmlsync の同期のカスタマイズ [106 ページ]

1.6.6.1 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
// 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、プロシージャの実装時に使用できるパラメータ値が用意されています。パラメータ値の中には、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は、ストアドプロシージャ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 プロシージャには Mobile Link ユーザのパラメータが 1 つあります。これは、同期している Mobile Link ユーザです。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 つのインタフェー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。

1.6.6.2 イベントフックプロシージャ

イベントフックプロシージのを設計と使用に関しては、いくつかの考慮事項があります。

- イベントフックプロシージャでは、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は、dbmlsync 接続で使用されるユーザ名、またはユーザ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。
- MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。
  - SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
  - CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

このセクションの内容:

1.6.6.2.1 #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、パラメータ値を受け取ります。パラメータ値の中には、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。

Name	Value
subscription_0	sub1
subscription_1	sub2
publication_0	pub1
publication_1	pub2
MobiLink user	MyUser
Script version	v1

注記

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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

例

次のサンプル sp_hook_dbmlsync_delay プロシージャは、#hook_dict テーブルでの in/out パラメータの使用を示します。このプロシージャでは、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 ユーザ名 (sp_hook_dbmlsync_begin イベントに使用可能なパラメータの 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
```

1.6.6.2.2 イベントフックプロシージャ用の接続

各イベントフックプロシージャは、同期自体と同じ接続で実行されます。ただし、いくつかの例外があります。

例外を次に示します。

- 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。

1.6.6.2.3 イベントフックプロシージャ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。実装された各プロシージャは、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。

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。

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 パラメータを使用します。

エラーの無視

イベントフックプロシージャ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。

関連情報

sp_hook_dbmlsync_download_log_ri_violation [246 ページ]
sp_hook_dbmlsync_download_ri_violation [248 ページ]
sp_hook_dbmlsync_ml_connect_failed [265 ページ]
sp_hook_dbmlsync_communication_error [237 ページ]
sp_hook_dbmlsync_sql_error [278 ページ]
sp_hook_dbmlsync_misc_error [261 ページ]
sp_hook_dbmlsync_all_error [232 ページ]
sp_hook_dbmlsync_end [253 ページ]
IgnoreHookErrors (eh) 拡張オプション [170 ページ]

1.6.6.3 sp_hook_dbmlsync_abort

このストアドプロシージャは、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。
*publication*_n (in)	パブリケーション	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
exit code (in|out)	number	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。

名前	値	説明
script version (input)	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)	subscription name(s)	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が存在する場合、そのプロシージャは 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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

例

次のプロシージャは、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
  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;
```

関連情報

同期イベントフックの順序 [224 ページ]
sp_hook_dbmlsync_process_exit_code [268 ページ]

1.6.6.4 sp_hook_dbmlsync_all_error

このストアドプロシージャを使用して、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)	*publication*	推奨されていません。代わりに subscription_n instead 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)	*MobiLink user name*	同期対象となる MobileLink ユーザ。

名前	値	説明
script version (in)	*script version name*	同期に使用される Mobile Link スクリプトバージョン。
error message (in)	*error message text*	これは、dbmlsync ログに表示されるテキストと同じです。
error id (in)	*INTEGER*	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)	*INTEGER*	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)	*subscription name(s)*	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

イベントフックプロシージャ内でのエラーと警告の処理 [229 ページ]
sp_hook_dbmlsync_communication_error [237 ページ]
sp_hook_dbmlsync_misc_error [261 ページ]
sp_hook_dbmlsync_sql_error [278 ページ]
sp_hook_dbmlsync_end [253 ページ]

1.6.6.5 sp_hook_dbmlsync_begin

このストアドプロシージャを使用して、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)	publication	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	Mobile Link user name	同期対象となる Mobile Link ユーザ。

名前	値	説明
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.6 sp_hook_dbmlsync_communication_error

このストアドプロシージャは、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	numeric	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。

名前	値	説明
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)	subscription name(s)	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。

このプロシージャのアクション は、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

イベントフックプロシージャ内でのエラーと警告の処理 [229 ページ]
sp_hook_dbmlsync_all_error [232 ページ]
sp_hook_dbmlsync_misc_error [261 ページ]
sp_hook_dbmlsync_sql_error [278 ページ]

1.6.6.7 sp_hook_dbmlsync_delay

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	subscription name(s)	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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。

関連情報

同期イベントフックの順序 [224 ページ]
イベントフックを使用した同期の開始 [105 ページ]
sp_hook_dbmlsync_download_end [244 ページ]
sp_hook_dbmlsync_end [253 ページ]

1.6.6.8 sp_hook_dbmlsync_download_begin

このストアドプロシージャを使用して、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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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、このプロシージャのアクションがコミットまたはロールバックされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.9 sp_hook_dbmlsync_download_end

このストアドプロシージャを使用して、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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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、このプロシージャのアクションがコミットまたはロールバックされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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

    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
```

関連情報

同期イベントフックの順序 [224 ページ]
イベントフックを使用した同期の開始 [105 ページ]
sp_hook_dbmlsync_delay [240 ページ]

1.6.6.10 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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
foreign key table (in)	`table name`	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)	`table name`	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)	`role name`	フック呼び出し対象の外部キーのロール名。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
 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;
```

関連情報

同期イベントフックの順序 [224 ページ]
sp_hook_dbmlsync_download_ri_violation [248 ページ]

1.6.6.11 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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
foreign key table (in)	`table name`	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)	`table name`	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)	`role name`	フック呼び出し対象の外部キーのロール名。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

sp_hook_dbmlsync_download_log_ri_violation [246 ページ]

1.6.6.12 sp_hook_dbmlsync_download_table_begin

このストアドプロシージャを使用して、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)	`table name`	操作が適用される予定のテーブル。
*publication*_n (in)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
*subscription*_n (in)	`subscription name(s)`	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、このプロシージャのアクションがコミットまたはロールバックされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.13 sp_hook_dbmlsync_download_table_end

このストアドプロシージャを使用して、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)	`table name`	操作が直前に適用されたテーブル。
delete count (in)	`number of rows`	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。

名前	値	説明
upsert count (in)	`number of rows`	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)	`publication`	推奨されていません。代わりに *subscription_n* instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
subscription_n (in)	`subscription name(s)`	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、このプロシージャのアクションがコミットまたはロールバックされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.14 sp_hook_dbmlsync_end

このストアドプロシージャを使用して、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。

名前	値	説明
exit code (in)	数字	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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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。

名前	値	説明
restartable download size (in)	整数	restartable download パラメータが *true* である場合、このパラメータ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、このパラメータの値は無効です。
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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

同期イベントフックの順序 [224 ページ]
イベントフックプロシージャ内でのエラーと警告の処理 [229 ページ]

1.6.6.15 sp_hook_dbmlsync_log_rescan

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
discarded storage (in)	number	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

名前	値	説明
*subscription*_n (in)	subscription name(s)	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

HoverRescanThreshold (hrt) 拡張オプション [169 ページ]

1.6.6.16 sp_hook_dbmlsync_logscan_begin

このストアドプロシージャを使用して、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)	number	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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	subscription name(s)	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.17 sp_hook_dbmlsync_logscan_end

このストアドプロシージャを使用して、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)	数字	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)	数字	同期する各サブスクリプションの初期進行値。n 値は、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.18 sp_hook_dbmlsync_misc_error

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	subscription name(s)	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

イベントフックプロシージャ内でのエラーと警告の処理 [229 ページ]
sp_hook_dbmlsync_communication_error [237 ページ]
sp_hook_dbmlsync_all_error [232 ページ]
sp_hook_dbmlsync_sql_error [278 ページ]

1.6.6.19 sp_hook_dbmlsync_ml_connect_failed

このストアドプロシージャを使用して、Mobile Link サーバ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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
connection address (in|out)	`connection address`	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。
connection type (in|out)	`network protocol`	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は、CommunicationType (ctp) 拡張オプションによって指定されます。
user data (in|out)	`user-defined data`	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。

名前	値	説明
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。
retry (in|out)	*true* | *false*	失敗した接続をリトライするには、この値を true に設定します。デフォルトは *false* です。
*subscription*_n (in)	`subscription name(s)`	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が存在する場合、Mobile Link サーバへの接続で dbmlsync が失敗したときにそのプロシージャが呼び出されます。

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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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を試行します。

```
CREATE TABLE conn_list (
    label   INTEGER PRIMARY KEY,
    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;
```

関連情報

Mobile Link クライアントネットワークプロトコルオプション [23 ページ]
CommunicationType (ctp) 拡張オプション [161 ページ]

-r dbmlsync オプション [142 ページ]

1.6.6.20 sp_hook_dbmlsync_process_exit_code

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	number	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)	number	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)	number	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。

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)	subscription name(s)	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。

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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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;
```

関連情報

同期イベントフックの順序 [224 ページ]
sp_hook_dbmlsync_abort [230 ページ]

1.6.6.21 sp_hook_dbmlsync_schema_upgrade

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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* | *onsuccess*	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。 *onsuccess* - フックが正常に実行された場合は、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。

権限

MANAGE REPLICATION システム権限を持つユーザは、フックプロシージャを作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

例

次の例では、sp_hook_dbmlsync_schema_upgrade プロシージャを使用して、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;
```

関連情報

リモート Mobile Link クライアントでのスキーマの変更 [69 ページ]

1.6.6.22 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を使用します。

#hook_dict テーブルのロー

名前	値	説明
publication_n (in)	`publication`	推奨されていません。代わりに *subscription_n* instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
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。
subscription_n (in)	`subscription name(s)`	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。

- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

例

次の例では、sp_hook_dbmlsync_set_extended_options を使用して、Increment 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 Increment extended option
   UPDATE #hook_dict
          SET value = 'Increment=10K'
    WHERE name = 'extended options';
 END IF;
END;
```

関連情報

同期イベントフックの順序 [224 ページ]
Mobile Link SQL Anywhere クライアントの拡張オプション [155 ページ]
オプション競合の dbmlsync による解決方法 [159 ページ]

1.6.6.23 sp_hook_dbmlsync_set_ml_connect_info

このストアドプロシージャを使用して、Mobile Link サーバ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。

名前	値	説明
publication_n (in)	`publication`	推奨されていません。代わりに *subscription_n* instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	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, or https`	Mobile Link サーバへの接続に使用するネットワークプロトコル。

名前	値	説明
connection address (in/out)	`protocol options`	Mobile Link サーバへの接続に使用する通信アドレス。
*subscription*_n (in)	`subscription name(s)`	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 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 [23 ページ]
CommunicationType (ctp) 拡張オプション [161 ページ]
sp_hook_dbmlsync_set_extended_options [273 ページ]

1.6.6.24 sp_hook_dbmlsync_set_upload_end_progress

このストアドプロシージャは、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	推奨されていません。代わりに subscription_n instead 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。
*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。

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
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ。
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)	timestamp	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)	subscription name(s)	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

関連情報

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 [304 ページ]
同期イベントフックの順序 [224 ページ]
スクリプト化されたアップロード [295 ページ]

1.6.6.25 sp_hook_dbmlsync_sql_error

このストアドプロシージャを使用して、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	推奨されていません。代わりに subscription_n instead 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 ユーザ名	同期対象となる Mobile Link ユーザ。
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)	numeric	メッセージをユニークに識別する ID。

名前	値	説明
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。
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)	subscription name(s)	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。

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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

関連情報

イベントフックプロシージャ内でのエラーと警告の処理 [229 ページ]
sp_hook_dbmlsync_all_error [232 ページ]
sp_hook_dbmlsync_communication_error [237 ページ]
sp_hook_dbmlsync_misc_error [261 ページ]

1.6.6.26 sp_hook_dbmlsync_upload_begin

このストアドプロシージャを使用して、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)	`publication`	推奨されていません。代わりに *subscription_n* instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
Script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン
subscription_n (in)	`subscription name(s)`	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。

このプロシージャのアクションは、実行直後にコミットされます。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.27 sp_hook_dbmlsync_upload_end

Mobile Link サーバ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。
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 サーバと dbmlsync で異なっていました。Mobile Link サーバは、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 サーバがアップロードを受信し、コミットしました。 *failed* - Mobile LinkMobile Link サーバは、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)	`publication`	推奨されていません。代わりに *subscription*_n instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
script version (in)	`script version name`	同期に使用される Mobile Link スクリプトバージョン

名前	値	説明
authentication value (in)	`value`	この値は、dbmlsync による Mobile Link サーバに対する認証結果を示します。サーバ上の 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)	`subscription name(s)`	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が存在する場合、dbmlsync がアップロードを送信し、Mobile Link サーバ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値の範囲は、次のとおりです。

UPLD_ERR_INVALID_USERID_OR_PASSWORD

ユーザ 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 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

ユーザの要求によりアップロードが中止されました。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

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
```

関連情報

同期イベントフックの順序 [224 ページ]

1.6.6.28 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)	`publication`	推奨されていません。代わりに *subscription_n* instead 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。
Mobile Link user (in)	`Mobile Link user name`	同期対象となる Mobile Link ユーザ。
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)	`number`	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 *file generation number_n* があります。*subscription_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。
user data (in)	`string`	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。

名前	値	説明
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。
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)	`subscription name(s)`	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。

権限

MANAGE REPLICATION システム権限を持つユーザはフックプロシージャが作成できます。ただし、フックによる情報のやり取りに使用される #hook_dict table にフックがアクセスできることを確認するには、フックが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して定義されます。

例

次の例では、ユーザ文字列 '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;
```

関連情報

-be dbmlsync オプション [119 ページ]
-bg dbmlsync オプション [120 ページ]

1.6.7 Dbmlsync C++ API リファレンス

Dbmlsync C++ API は、C++ 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フェースです。

ヘッダファイル

```
dbmlsynccli.hpp
```

注記

API リファレンスマニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows のスタートメニューを使用してアクセスするか (Microsoft Windows)、`C:¥Program Files¥SQL Anywhere 17¥Documentation` にナビゲートします。

また、DocCommentXchange の Web で、SAP SQL Anywhere API リファレンスマニュアルにアクセスすることもできます。http://dcx.sap.com

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 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=passwd -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 );
}
```

1.6.8 Dbmlsync .NET API リファレンス

Dbmlsync .NET API 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フェースです。

ネームスペース

```
Sap.MobiLink.Client
```

注記

API リファレンスマニュアルをお探しですか。マニュアルをローカルにインストールした場合は、Windows **のスタートメニューを使用してアクセスするか** (Microsoft Windows)、`C:¥Program Files¥SQL Anywhere 17¥Documentation` **にナビゲートします。**

また、DocCommentXchange **の** Web **で、**SAP SQL Anywhere API **リファレンスマニュアルにアクセスすることもできます。** http://dcx.sap.com

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 Sap.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 not required unless
            // your SQL Anywhere install is not in your path or you have multiple
            // versions of the product installed.
            cli1.SetProperty("server path", "d:¥¥sap¥¥sqlany17¥¥bin32;"
            cli1.StartServer(3426,
             "-c server=cons;dbn=rem1;uid=dba;pwd=passwd -ve+ -ot c:¥¥temp¥
¥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);
            cli1.WaitForServerShutdown(10000);
            cli1.Disconnect();
            cli1.Fini();
            Console.ReadLine();
        }
```

```
        }
}
```

1.6.9 dbmlsync の DBTools インタフェース

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。

> **注記**
>
> アプリケーションに同期を統合する場合、Dbmlsync API インタフェースを使用することをお奨めします。DBTools インタフェースと非常に良く似た機能があり、使い方も簡単です。

dbmlsync 用の DBTools インタフェー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、このインタフェースを使用して、同期を起動し、カスタムユーザインタフェースに dbmlsync の出力メッセージを表示できます。

dbmlsync 用の DBTools インタフェースは、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も含まれます。

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。

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は、a_sync_db インスタンスへのポインタだけです。

このセクションの内容:

dbmlsync の DBTools インタフェースの設定 [291 ページ]
このタスクでは、C または C++ の DBTools インタフェースを使用することによって dbmlsync を設定し開始する基本的な手順を説明します。

関連情報

Dbmlsync API [107 ページ]

1.6.9.1 dbmlsync の DBTools インタフェースの設定

このタスクでは、C または C++ の DBTools インタフェースを使用することによって dbmlsync を設定し開始する基本的な手順を説明します。

手順

1. DBTools ヘッダファイルをインクルードします。

   DBTools ヘッダファイル `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 インタフェースを起動します。
   - 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");
     }
     ```

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。

     ```
     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=passwd";
     ```
   - 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。

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;
```

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。

   - 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 つのパラメータを受け取ります。

  index

  同期の現在の進行状況を表す整数です。

  max

  進行状況の最大値です。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 );
```

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、同じパラメータ値または異なるパラメータ値を指定して複数回繰り返すことができます。

7. DBTools インタフェースをシャットダウンします。

   ```
   DBToolsFini( &info );
   ```

   DBToolsFini 関数は、DBTools リソースを解放します。

結果

dbmlsync が設定され、開始されます。

関連情報

-c dbmlsync オプション [122 ページ]
-eu dbmlsync オプション [130 ページ]
-s dbmlsync オプション [143 ページ]
-n dbmlsync オプション (旧式) [134 ページ]

1.6.10 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奨めします。

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の名前を指定します。ストアドプロシージャは、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。

このセクションの内容:

1.6.10.1 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奨めします。

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。

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 (アップロード、挿入、削除用に 1 つずつ) を定義できます。

次を含むパブリケーションを作成します。

キーワード WITH SCRIPTED UPLOAD およびストアドプロシージャ名

関連情報

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 [303 ページ]
スクリプト化されたアップロードのパブリケーション [308 ページ]
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 [308 ページ]
LockTables (lt) 拡張オプション [172 ページ]

1.6.10.2 スクリプト化されたアップロードの設計上の考慮事項

スクリプト化されたアップロードでは、以下の設計事項を考慮してください。

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。

次のので例は、スクリプト化されたアップロードを使用したチュートリアルを用いて、競合解決を実装します。

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。

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。

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、スクリプト化されたアップロードをテーブルをロックしないで実行する場合についてのマニュアル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。

アップロードプロシージャの 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 (
   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。

時間	
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。

このセクションの内容:

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 [300 ページ]
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。

関連情報

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 [308 ページ]
sp_hook_dbmlsync_set_upload_end_progress [276 ページ]

1.6.10.2.1 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。

可能であれば、デフォルトのテーブルをロックした動作を使用します。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は、dbmlsync のコマンドラインで指定したデータベースユーザ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。

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は、dbmlsync が Mobile Link サーバに接

続している間に呼び出されるので、サーバ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。

時間	
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。

時間	
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。

サーバ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。

1.6.10.3 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

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を作成します。

ストアドプロシージャが呼び出されると、#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に渡すことができます。ストアドプロシージャは、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。

ストアドプロシージャによって #hook_dict テーブルにアクセスできることを確認するには、ストアドプロシージャが次のいずれかの要件を満たしている必要があります。

- SELECT ANY TABLE および UPDATE ANY TABLE システム権限を持つユーザが所有します。
- CREATE PROCEDURE 文の SQL SECURITY INVOKER 句を使用して作成されています。

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	*timestamp as string*	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-bit unsigned integer*	符号なし整数で示された開始進行状況
end progress	*timestamp as string*	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。
raw end progress	*64-bit unsigned integer*	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	*subscription name(s)*	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	*publication name*	推奨されていません。代わりに subscription_n instead 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	*version name*	同期に使用される Mobile Link スクリプトバージョン
MobiLink user	*MobiLink user name*	同期対象となる MobileLink ユーザ。

このセクションの内容:

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 [304 ページ]
スクリプト化されたアップロードプロシージャ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。

挿入用のストアドプロシージャ [305 ページ]
挿入用のストアドプロシージャは、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。

削除用のストアドプロシージャ [306 ページ]
削除用のストアドプロシージャは、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。

更新用のストアドプロシージャ [307 ページ]
更新用のストアドプロシージャは、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。

スクリプト化されたアップロードのパブリケーション [308 ページ]
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を指定します。

関連情報

#hook_dict テーブル [226 ページ]

1.6.10.3.1 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。

関連情報

sp_hook_dbmlsync_set_upload_end_progress [276 ページ]

1.6.10.3.2 挿入用のストアドプロシージャ

挿入用のストアドプロシージャは、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を定義する方法の詳細については、スクリプト化されたアップロードを使用するチュートリアル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 t1_insert を挿入プロシージャとして登録します。t1_insert ストアドプロシージャ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
   )
)
```

関連情報

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 [308 ページ]

1.6.10.3.3 削除用のストアドプロシージャ

削除用のストアドプロシージャは、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を定義する方法の詳細については、スクリプト化されたアップロードを使用するチュートリアル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 t1_delete を削除プロシージャとして登録します。t1_delete ストアドプロシージャ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
   )
)
```

関連情報

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 [308 ページ]

1.6.10.3.4 更新用のストアドプロシージャ

更新用のストアドプロシージャは、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)。

更新用のストアドプロシージャは、挿入または削除用のストアドプロシージャの 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

アップロード用のストアドプロシージャを定義する方法の詳細については、スクリプト化されたアップロードを使用するチュートリアル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 t1_update を更新プロシージャとして登録します。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
   )
)
```

関連情報

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 [308 ページ]

1.6.10.3.5 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
      )
   )
```

1.6.10.4 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。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、パブリケーション、サブスクリプションを作成します。

前提条件

次のロールおよび権限が必要です。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール
- CREATE ANY TRIGGER システム権限

1. レッスン 1: 統合データベースの作成 [309 ページ]
   このレッスンでは、統合データベースを作成し、Mobile Link で使用するために設定します。
2. レッスン 2: リモートデータベースの作成 [311 ページ]
   このレッスンでは、リモートデータベースを作成し、オブジェクトを移植します。
3. レッスン 3: 挿入の処理 [312 ページ]
   ストアドプロシージャと、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。

4. レッスン 4: 更新の処理 [314 ページ]
   このレッスンでは、テーブル、トリガ、および更新を処理するストアドプロシージャをそれぞれ作成します。
5. レッスン 5: 削除の処理 [316 ページ]
   このレッスンでは、テーブル、トリガ、および削除を処理するストアドプロシージャをそれぞれ作成します。
6. レッスン 6: 更新前イメージテーブルと削除テーブルのクリーンアップ [318 ページ]
   このレッスンでは、更新前イメージテーブルと削除テーブルをクリーンアップするための upload_end フックを作成します。
7. レッスン 7: パブリケーション、Mobile Link ユーザ、サブスクリプションの作成 [319 ページ]
   このレッスンでは、パブリケーション、Mobile Link ユーザ、サブスクリプションを作成します。
8. レッスン 8: スクリプト化されたアップロードの実行 [320 ページ]
   このレッスンでは、作成した SQL 文とコマンドを実行して、スクリプト化されたアップロードを確認します。
9. レッスン 9: クリーンアップ [321 ページ]
   チュートリアルをコンピュータから削除します。

1.6.10.4.1 レッスン 1: 統合データベースの作成

このレッスンでは、統合データベースを作成し、Mobile Link で使用するために設定します。

前提条件

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

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 (`scriptedupload`) 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。

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。

手順

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:¥scriptedupload
   cd c:¥scriptedupload
   ```

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。

   ```
   dbinit -dba DBA,passwd 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=passwd;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" "%SQLANY17%¥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。

結果

統合データベースが作成され、Mobile Link で使用できるように設定されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

次のタスク: レッスン 2: リモートデータベースの作成 [311 ページ]

1.6.10.4.2 レッスン 2: リモートデータベースの作成

このレッスンでは、リモートデータベースを作成し、オブジェクトを移植します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

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 -dba DBA,passwd 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=passwd;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。

結果

リモートデータベースが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 1: 統合データベースの作成 [309 ページ]

次のタスク: レッスン 3: 挿入の処理 [312 ページ]

1.6.10.4.3 レッスン 3: 挿入の処理

ストアドプロシージャと、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。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

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;
```

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;
```

結果

このプロシージャは、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。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 2: リモートデータベースの作成 [311 ページ]

次のタスク: レッスン 4: 更新の処理 [314 ページ]

1.6.10.4.4 レッスン 4: 更新の処理

このレッスンでは、テーブル、トリガ、および更新を処理するストアドプロシージャをそれぞれ作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

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。

手順

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を作成します。挿入トリガと同様、このトリガ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、タイムスタンプがほぼ同じである場合を除く)。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
    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。

結果

更新前イメージを保存するテーブル、更新される各ローの更新前イメージを保存するトリガ、更新を処理するストアドプロシージャが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 3: 挿入の処理 [312 ページ]

次のタスク: レッスン 5: 削除の処理 [316 ページ]

1.6.10.4.5 レッスン 5: 削除の処理

このレッスンでは、テーブル、トリガ、および削除を処理するストアドプロシージャをそれぞれ作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

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、このトリガは、ダウンロード中は呼び出されません。このトリガは、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。

結果

削除するローのリストを保存するテーブルと、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と、削除を処理するアップロードプロシージャが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 4: 更新の処理 [314 ページ]

次のタスク: レッスン 6: 更新前イメージテーブルと削除テーブルのクリーンアップ [318 ページ]

1.6.10.4.6 レッスン 6: 更新前イメージテーブルと削除テーブルのクリーンアップ

このレッスンでは、更新前イメージテーブルと削除テーブルをクリーンアップするための upload_end フックを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

コンテキスト

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。

手順

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;
```

結果

アップロードが成功したときに、更新前イメージと削除の結果がテーブルから削除されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 5: 削除の処理 [316 ページ]

次のタスク: レッスン 7: パブリケーション、Mobile Link ユーザ、サブスクリプションの作成 [319 ページ]

1.6.10.4.7 レッスン 7: パブリケーション、Mobile Link ユーザ、サブスクリプションの作成

このレッスンでは、パブリケーション、Mobile Link ユーザ、サブスクリプションを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

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、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が登録されます。u1 という Mobile Link ユーザと、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';
```

結果

パブリケーション、Mobile Link ユーザ、サブスクリプションが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 6: 更新前イメージテーブルと削除テーブルのクリーンアップ [318 ページ]

次のタスク: レッスン 8: スクリプト化されたアップロードの実行 [320 ページ]

1.6.10.4.8 レッスン 8: スクリプト化されたアップロードの実行

このレッスンでは、作成した SQL 文とコマンドを実行して、スクリプト化されたアップロードを確認します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

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 サーバを起動します。

```
mlsrv17 -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. 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次の SQL 文を実行し、挿入がアップロードされていることを確認します。

```
SELECT * FROM employee
```

結果

このレッスンの最初に挿入された値が表示されます。

次のステップ

次のレッスンに進みます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 7: パブリケーション、Mobile Link ユーザ、サブスクリプションの作成 [319 ページ]

次のタスク: レッスン 9: クリーンアップ [321 ページ]

1.6.10.4.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に一覧されているロールと権限を持っている必要があります。

手順

1. Interactive SQL のすべてのインスタンスを閉じます。

2. SQL Anywhere、Mobile Link、同期クライアントの各ウィンドウを閉じます。
3. 次の手順で、チュートリアルに関連するすべての ODBC データソースを削除します。
   a. ODBC アドミニストレータを起動します。

      次のコマンドを実行します。

      ```
      odbcad32
      ```

   b. **dsn_consol** と **dsn_remote** の各データソースを削除します。
4. 統合データベースとリモートデータベースが保存されているディレクトリ `c:¥scriptedupload` に移動し、すべてのファイルを削除します。

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: 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 [308 ページ]

前のタスク: レッスン 8: スクリプト化されたアップロードの実行 [320 ページ]

1.7 高度な機能: Mobile Link テンプレートシステム (mltemplate)

Moble Link テンプレートシステムユーティリティ (mltemplate) は、Mobile Link と SAP HANA リモートデータ同期の開発者のための開発支援ツールです。

mltemplate ユーティリティは、数多くのテーブルを使用する大規模プロジェクトで有用です。同期プロジェクトに必要なソースファイルの数が減少するため、修正が容易になり、プロジェクトの拡大に応じて新しいテーブルを追加できます。このユーティリティはカスタマイズ可能であり、同期ソリューションを反復的に開発できるように、構築ツールと組み合わせることができます。

mltemplate を使用するには、Moble Link をよく理解したうえで、同期スクリプトと他の関連オブジェクトの作成経験が必要です。

mltemplate のコマンドを実行すると、プロジェクトファイルが JSON で作成され、データベーススキーマがファイルにインポートされます。スキーマは同期されるテーブルを表します。mltemplate では、プロジェクトファイル内のスキーマオブジェクトに対して Handlebars テンプレート (ユーザが作成してカスタマイズするテンプレート) を実行することで、SQL ファイルが生成されます。どのテンプレートがどのスキーマオブジェクトに適用されるのかは、ユーザが手作業でプロジェクトファイルに追加するルールによって決定されます。出力の SQL ファイルは、同期スクリプトと他の同期オブジェクトを作成するために、統合データベースまたはリモートデータベースに対して実行できます。

このセクションの内容:

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]
mltemplate はクライアントデータベースの作成と更新に役立つ拡張ユーティリティです。

プロジェクトファイル (mltemplate) [330 ページ]
プロジェクトファイルには、データベーススキーマの定義、およびルールと変数の定義が保存されます。

ルール (mltemplate) [334 ページ]
ルールは、どのテンプレートファイルをどのスキーマオブジェクトに適用するのかを指定します。プロジェクトファイル内でルールを手作業で定義した後、*mltemplate gen* を実行することで、ルールを実行します。

カスタム変数 (mltemplate) [336 ページ]
カスタム変数はプロジェクトファイル内で定義し、テンプレートファイル内で使用します。

スキーマ変数 (mltemplate) [338 ページ]
スキーマ変数は、データベーススキーマをプロジェクトにインポートするときに生成されます。スキーマ変数はテンプレートファイルで使用します。

Handlebars テンプレートとヘルパー (mltemplate) [341 ページ]
mltemplate では、通常は Mobile Link 同期スクリプトが記載された Handlebars テンプレートが使用されます。テンプレートには同期追跡スキーマ (シャドウテーブル、システムプロシージャ、トリガの定義など) も記載できます。

出力ファイル (mltemplate) [345 ページ]
mltemplate から生成されるコンテンツを編成するために、テンプレート別および/またはルール別に出力をグループ化します。

チュートリアル: Moble Link テンプレートシステムユーティリティ (mltemplate) [347 ページ]
SQL Anywhere の統合データベースとクライアントデータベースを使用する同期プロジェクトを mltemplate によって作成します。

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.1 同期ソリューション用の SQL ファイルの生成 (mltemplate)

同期プロジェクト用の同期スクリプトと他のオブジェクトを作成する SQL ファイルを生成するには、mltemplate を使用します。

前提条件

データベーススキーマの取得元とする既存の統合データベースまたはクライアントデータベースが存在する必要があります。

コンテキスト

mltemplate はプロジェクトファイルが存在するディレクトリから実行してください。mltemplate では、プロジェクトが存在するディレクトリではなく、mltemplate が実行されるディレクトリを基準として、相対パスが解釈されます。

mltemplate では、デフォルトの Handlebars テンプレート、Handlebars ヘルパー、グローバルスキーマ変数定義を含んだファイルを見つけるために、常にデフォルトのインストールディレクトリが検索されます。デフォルトのテンプレートファイルをコピーして、独自の目的でカスタマイズした場合は、そのテンプレートファイル名を変更してください。

mltemplate では、SAP HANA と SAP SQL Anywhere の統合データベース専用のデフォルトの Handlebars テンプレートが提供されます。SQL Anywhere クライアントデータベースおよびその他のすべての統合データベースに対しては、テンプレートを作成する必要があります。

同期ソリューションを繰り返し開発できるように、mltemplate を構築ツールと組み合わせることができます。

手順

1. 同期プロジェクト用のディレクトリ (Handlebars テンプレート用のディレクトリと作業ディレクトリなど) を設定します。

   次の表に、ディレクトリ構造の例を示します。

ディレクトリ名	説明
`sync-project`	プロジェクトファイル用の作業ディレクトリ。mltemplate は常にこのディレクトリから実行してください。
`sync-project ¥templates¥cdb`	統合データベース用の Handlebars テンプレートを保存するためのディレクトリ。 mltemplate では、SAP HANA と SAP SQL Anywhere の統合データベース専用のデフォルトの Handlebars テンプレートが提供されます。これらのテンプレートはサンプル専用に提供されるもので、本稼働用途としては保証されません。他の統合データベースに対しては、テンプレートを作成する必要があります。
`sync-project ¥templates¥rem`	クライアントデータベース用の Handlebars テンプレートを保存するためのディレクトリ。これらのテンプレートはユーザが作成する必要があります。
`sync-project ¥helpers`	ユーザが作成する追加の Handlebars ヘルパーを保存するためのディレクトリ。 デフォルトのヘルパーは *%SQLANY17%*`¥MobiLink¥MLTemplate¥helpers ¥sqlhelpers.js` にあります。
`sync-project¥out`	*mltemplate gen* から生成される出力を保存するためのディレクトリ。

2. *mltemplate new* を実行して、プロジェクトファイルを作成します。

   例:

   ```
   mltemplate new project.json  -ConsType sqla -t sync-project¥templates¥cdb
   ```

   プロジェクトファイルが作成されます。
3. mltemplate import を実行して、既存の統合データベースまたはクライアントデータベースからプロジェクトファイルにデータベーススキーマをインポートします。

テーブルのスキーマ定義は、プロジェクトファイル内の SyncTables 配列に追加されます。mltemplate はデータベースのスキーマに基づいてスキーマ変数を生成します。これらのスキーマ変数は、SyncTables 配列の最後に追加されます。

4. プロジェクトファイル内で、
   a. mltemplate によって生成されたスキーマ変数を確認します。これらの変数はデフォルトテンプレートで使用されます。
   b. テーブルを Handlebars テンプレートにリンクするルールを手作業で作成します。

      同じ動作のテーブルをグループにまとめ、Handlebars テンプレートをそのグループに割り当てるルールを作成します。
   c. テンプレートで使用できるカスタム変数を手作業で定義します。
5. 同期プロジェクト用の Handlebars テンプレートを作成します。
   a. *%SQLANY17%*¥MobiLink¥MLTemplate¥Templates¥ConsDB¥ からテンプレートディレクトリ sync-project¥templates¥cdb に、デフォルトの Handlebars テンプレートをコピーします。このテンプレートディレクトリをプロジェクトファイル内の templateRoots 配列に追加します。
   b. コピーしたテンプレートファイルの名前を変更します。mltemplate で、これらのテンプレートファイルがインストールディレクトリ内のバージョンと混同されないようにするためです。
   c. テンプレートファイルをカスタマイズし、必要に応じて新しいテンプレートファイルを作成します。
   d. プロジェクトファイル内のルールで、これらのテンプレートファイルを参照します。
6. Handlebars ヘルパーを作成します。必要な場合は、プロジェクトファイル内の templateHelpers 配列にヘルパーファイルを追加します。
   a. JavaScript ファイルを作成します。このファイルは、デフォルトのヘルパーファイル sqlhelpers.js と同じ名前にしないでください。
   b. ヘルパーを作成し、Handlebars.registerHelper の呼び出しをファイルに追加することで、ヘルパーを登録します。
   c. プロジェクトファイル内の templateHelpers 配列にヘルパーファイルを追加します。
7. *mltemplate gen* を実行して、プロジェクトファイルとテンプレートから SQL ファイルを生成します。
8. SQL ファイルを使用して、統合データベース内またはリモートデータベース内に定義を作成します。

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.2 Mobile Link テンプレートユーティリティ (mltemplate)

mltemplate はクライアントデータベースの作成と更新に役立つ拡張ユーティリティです。

構文

```
mltemplate
   new project-filename.json new-options
| import project-filename.json import-options
| gen project-filename.json gen-options
| modify project-filename.json javascript-expression
```

new

同期プロジェクトのスキーマと設定が保存されるプロジェクトファイルを作成します。

new-オプション	説明
-consType `database-type`	統合データベースのデータベースタイプを指定します。このオプションは、プロジェクトファイル内の consType フィールドの値を設定します。このオプションは必須です。 *sqla* (SAP SQL Anywhere の場合)、*hana* (SAP HANA の場合)、*ase* (SAP Adaptive Server Enterprise の場合)、*db2* (IBM DB2 の場合)、*mss* (Microsoft SQL Server の場合)、*mys* (MySQL の場合)、*orac* (Oracle の場合)、*iq* (SAP IQ の場合) を指定します。
-js `javascript-helper-file` [...]	カスタマイズされたテンプレートヘルパーを含む JavaScript ファイルを指定します。このオプションは、プロジェクトファイル内の templateHelpers 配列に列挙されたヘルパーファイルに、指定されたテンプレートヘルパーファイルを追加します。 デフォルトでは、templateHelpers 配列にはデフォルトのテンプレートヘルパーファイル `sqlhelpers.js` が記載されています。
-o `output-file`	グローバル出力ファイル名を指定します。このオプションは、プロジェクトファイル内の outFile フィールドの値を設定します。 デフォルトでは、生成されたコンテンツはこの SQL ファイルに保存されます。デフォルト値は `generatedScripts.txt` です。
-remType `database-type`	クライアントデータベースのデータベースタイプを指定します。このオプションは、プロジェクトファイル内の remoteDBType フィールドの値を設定します。 *ul* (Ultra Light の場合) または *sqla* (SAP SQL Anywhere の場合) を指定します。デフォルトは *sqla* です。
-s `script-version-string`	プロジェクトの Mobile Link スクリプトバージョンを指定します。このオプションは、プロジェクトファイル内の scriptVersion フィールドの値を設定します。バージョンにとって意味のある文字列を指定します。
-t `template-root-directory`	カスタマイズされた Handlebars テンプレートが存在するディレクトリを指定します。このオプションは、プロジェクトファイル内の templateRoots 配列のリストに、指定されたテンプレートディレクトリを追加します。このオプションは必須です。 SAP SQL Anywhere および SAP HANA 統合データベースのためのデフォルトテンプレートは、インストールディレクトリ (*%SQLANY17%*`¥MobiLink¥MLTemplate¥Templates`) にあります。これらのデフォルトテンプレートのみで同期システムを作成できます。ただし、独自のテンプレートを作成して、デフォルトテンプレートを置き換えるか、デフォルトテンプレートを補足する必要が生じる場合がよくあります。 サポートされる他の統合データベースに対しては、テンプレートを提供する必要があります。
-y	このプロジェクトファイルを作成するために、既存のプロジェクトファイルを警告なしで上書きします。

import

統合データベースのスキーマまたは既存のクライアントデータベースのスキーマをインポートします。スキーマは同期されるテーブルを表し、*mltemplate gen* によって使用されます。データベーススキーマは、プロジェクトファイル内の SyncTables 配列に追加されます。スキーマ変数は、データベーススキーマから生成されて、プロジェクトファイルに追加されます。カラムのデータ型は、インポートの一部として統合データベースとリモートデータベース間で自動的に変換されます。

import-オプション	説明
-catalog `catalog-name`	テーブルが属するカタログを指定します。このオプションが指定されない場合は、すべてのカタログがインポートされます。
-cons	スキーマがインポートされるデータベースが統合データベースであることを指定します。統合データベースからインポートする場合は、このオプションが必要です。このオプションと -consType オプションを同時に指定しないでください。
-consType `database-type`	統合データベースのタイプを指定します。この情報はデータ型変換に必要です。このオプションは、プロジェクトファイル内の consType フィールドを更新します。このオプションは、-rem オプションの指定時に必要であり、-cons オプションの指定時には指定できません。 *sqla* (SAP SQL Anywhere の場合)、*hana* (SAP HANA の場合)、*ase* (SAP Adaptive Server Enterprise の場合)、*db2* (IBM DB2 の場合)、*mss* (Microsoft SQL Server の場合)、*mys* (MySQL の場合)、*oracle* (Oracle の場合)、*iq* (SAP IQ の場合) を指定します。
-f `file`	テキストファイルに列挙されたテーブルのスキーマをインポートします。テキストファイル内では、1 行に 1 つのテーブルのみを列挙します。
-jdbc `jdbc-connection-string`	スキーマがインポートされるデータベースに対する JDBC 接続文字列を指定します。例: `-jdbc jdbc:sqlanywhere:DBF=Remotedatabase.db;UID=user;PWD=pwd;SERVER=remote`.-odbc または -jdbc を指定します。 SQL Anywhere データベースの場合、mltemplate では SQL Anywhere JDBC ドライバが使用されます。他のすべてのデータベースの場合、JDBC ドライバの JAR ファイルへのパスを指定する必要があります。JAR ファイルを指定するには、MLTEMPLATE_JARS 環境変数を使用します。
-odbc `odbc-connection-string`	スキーマがインポートされるデータベースに対する ODBC 接続文字列を指定します。例: `-odbc DSN=mydsn;UID=user;PWD=pwd`.-odbc または -jdbc を指定します。
-p `regular-expression` [...]	指定された正規表現にテーブル名が一致するテーブルのスキーマをインポートします。
-schema `schema-name` [...]	テーブルが属するスキーマを指定します。このオプションが指定されない場合は、すべてのスキーマがインポートされます。
-re	既存のプロジェクトファイルに対してインポートするときに、プロジェクト内ですでに定義されたテーブルのスキーマを再インポートします。
-rem	スキーマがインポートされるデータベースが既存のクライアントデータベースであることを指定します。-consType も指定する必要があります。このオプションはデフォルトです。

import-オプション	説明
-rule { `rule-name` | *none* }	指定されたルールにテーブルスキーマ定義を追加します。デフォルトでは、*default* というルールに定義が追加されます。*none* を指定すると、定義はルールに追加されません。
-t `tableName` [...]	指定されたテーブルのスキーマをインポートします。デフォルトでは、すべてのテーブルがインポートされます。

gen

プロジェクトファイル内で定義されたルールに基づいて SQL ファイルを生成します。これらのルールでは、インポートされたデータベーススキーマ内のテーブルに対して Handlebars テンプレートが実行されます。通常、同期プロジェクト用のデータベースの構築に使用する同期スクリプトと他のオブジェクトを含んだ SQL ファイルが出力されます。

gen-オプション	説明
-d `outDir`	生成された SQL ファイルを保存するためのディレクトリを指定します。このオプションは、プロジェクトファイル内の outDir フィールドを上書きします。
-dump	mltemplate によって Handlebars テンプレートファイルに送信されるすべての変数を出力します。これはテンプレートと変数のデバッグに役立ちます。
-o `out-file.SQL`	生成された出力内容を保存するための SQL ファイルを指定します。このオプションは、プロジェクトファイル内の outFile フィールドで指定された値を上書きします。
-rule `rule-name` [...]	指定されたルールのみに関する出力を生成します。デフォルトでは、すべてのルールに関する出力が生成されます。
-table `table-name` [...]	指定されたテーブルのみに関する出力を生成します。デフォルトでは、プロジェクトファイル内で指定されたすべてのテーブルに関する出力が生成されます。
template `template-name` [...]	指定された Handlebars テンプレートファイルのみに関する出力を生成します。デフォルトでは、プロジェクトファイル内で指定されたすべてのテンプレートファイルに関する出力が生成されます。
-v `variable-name variable-value`	カスタム変数を定義します。プロジェクト内に同じ名前の変数がある場合は、指定した変数によって既存の変数定義が上書きされます。-v オプションは、プロジェクトファイル内での変数定義に代わる手法を提供するものです。たとえば、-v オプションを使用してユーザ名を指定した後、複数の異なるユーザに対して mltemplate を複数回実行できます。

modify

指定された JavaScript 式またはファイルに基づいてプロジェクトファイルを更新します。modify コマンドを使用すると、プロジェクトファイル内のデータベーススキーマを編集できます。たとえば、mltemplate modify を実行すると、プロジェクトファイル内のカラム名の大文字/小文字を変更できます。modify コマンドを実行すると、syncModel という JavaScript グローバル変数にプロジェクトファイルがロードされ、指定された JavaScript 式が実行された後、変更されたプロジェクトファイルが保存されます。

modify-オプション	説明
`javascript-expression`	指定された JavaScript 式またはファイルに基づいてプロジェクトファイルを変更します。

備考

mltemplate はプロジェクトファイルが存在するディレクトリから実行してください。mltemplate では、プロジェクトが存在するディレクトリではなく、mltemplate が実行されるディレクトリを基準として、相対パスが解釈されます。

mltemplate ユーティリティでは、デフォルトの Handlebars テンプレート、Handlebars ヘルパー、グローバルスキーマ変数定義を含んだファイルを見つけるために、常にデフォルトのインストールディレクトリが検索されます。デフォルトのテンプレートファイルをコピーして、独自の目的でカスタマイズした場合は、そのテンプレートファイル名を変更してください。

mltemplate ユーティリティでは、SAP HANA と SAP SQL Anywhere の統合データベース専用のデフォルトの Handlebars テンプレートが提供されます。SQL Anywhere クライアントデータベースおよびその他のすべての統合データベースに対しては、テンプレートを作成する必要があります。

例

次の例では、統合データベースとクライアントデータベースが SQL Anywhere データベースである同期システムに対して、`project.json` というプロジェクトファイルを作成します。

```
mltemplate new project.json -t template/mobilink/cdb/sqla -s V1 -remType sqla -
consType sqla -y
```

次の例では、SQL Anywhere サンプルデータベースからプロジェクトに対して、Products テーブルをインポートします。

```
mltemplate import project.json -jdbc "jdbc:sqlanywhere:DSN=SQL Anywhere 17
Demo;uid=DBA;pwd=sql" -rem -t Products
```

関連情報

http://handlebarsjs.com/
http://handlebarsjs.com/
プロジェクトファイル (mltemplate) [330 ページ]
http://handlebarsjs.com/
ルール (mltemplate) [334 ページ]
Handlebars テンプレートとヘルパー (mltemplate) [341 ページ]
ルール (mltemplate) [334 ページ]

1.7.3 プロジェクトファイル (mltemplate)

プロジェクトファイルには、データベーススキーマの定義、およびルールと変数の定義が保存されます。

プロジェクトファイルは、*mltemplate new* の実行時に作成される JSON ファイルです。プロジェクトファイルには、次の情報が保存されます。

- プロジェクトに関する環境情報 (クライアントデータベースと統合データベースのデータベースタイプ、Handlebars テンプレートファイルなど)。この情報は JSON オブジェクトの最上位フィールドにあります。この情報を設定するには、*mltemplate new* および *mltemplate import* を実行します。
- *mltemplate import* の実行時にインポートされるテーブルスキーマ定義。この定義は、ユーザが作成する関連変数またはルールとともに、SyncTables 配列にあります。データベーススキーマ定義を設定するには、*mltemplate import* を実行します。定義に関連する変数とルールの追加および管理を行うには、プロジェクトファイルを手作業で編集します。
- ルールとカスタム変数の定義。グローバル変数定義とグローバルルールを設定するには、プロジェクトファイルを手作業で編集します。

例

プロジェクトファイル内のグローバルオプションの例:

```
{
  "outDir": null,
  "outFile": "generatedScripts.txt",
  "remoteDBType": "SQL Anywhere",
  "consDBType": "SQL Anywhere",
  "templateRoots": [
      "template"
  ],
  "templateVariables": {
      "scriptVersion": "V1",
      "consDBTableOwner": "GROUPO",
      "projectName": "sqla_demo"
  },
  "templateHelpers": [
      "sqlhelpers.js",
      "helpers/custom_helpers.js"
  ],
  "templateOutFiles": {},
  "templateRules": [
    {
      "ruleName": "default",
      "outFile": null,
      "tables": [
        "Products"
      ],
      "templateVariables": {},
      "templateFiles": [
          "mobilink/cdb/sqla/basic/shadow_table.hbs",
          "mobilink/cdb/sqla/basic/trigger.hbs",
          "mobilink/cdb/sqla/basic/sync_script.hbs",
          "mobilink/cdb/sqla/basic/stored_procedure.hbs",
          "mobilink/cdb/sqla/basic/load.hbs"
      ]
    },
    { "ruleName": "cdb_report",
      "tables": null,
      "templateVariables": { },
      "templateFiles": [
          "mobilink/cdb/sqla/test/report.hbs"
      ]
    },
```

```
    {
        "ruleName": "rdb_tables",
        "outFile": null,
        "tables": [
            "Products"
        ],
        "templateVariables": {},
        "templateFiles": [
            "mobilink/rdb/sqla/table.hbs"
        ]
      },
      { "ruleName": "publication",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "mobilink/rdb/sqla/publication.hbs"
        ]
      },
      { "ruleName": "subscription",
        "tables": null,
        "templateVariables": {
            "ml_host": "localhost",
            "ml_port": "2439"
        },
        "templateFiles": [
            "mobilink/rdb/sqla/subscription.hbs"
        ]
      },
      { "ruleName": "rdb_report",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "mobilink/rdb/sqla/test/report.hbs"
        ]
      },
      { "ruleName": "rdb_insert",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "mobilink/rdb/sqla/test/insert.hbs"
        ]
      }
 ],
```

プロジェクトファイル内の syncTables オブジェクトの例:

```
 "syncTables": [
    {
      "tableName": "Products",
      "consDBTableName": "Products",
      "templateVariables": {},
      "columns": [
        {
          "columnName": "ID",
          "templateVariables": {},
          "baseDomain": "INTEGER",
          "domain": "INTEGER",
          "size": null,
          "scale": null,
          "nullable": 0,
          "ordinal": 1,
          "cons": {
            "columnName": "ID",
            "baseDomain": "INTEGER",
            "domain": "INTEGER",
            "size": null,
            "scale": null,
```

```
            "nullable": 0
          }
        },
        {
          "columnName": "Name",
          "templateVariables": {},
          "baseDomain": "CHAR",
          "domain": "CHAR(15)",
          "size": 15,
          "scale": null,
          "nullable": 0,
          "ordinal": 2,
          "cons": {
            "columnName": "Name",
            "baseDomain": "CHAR",
            "domain": "CHAR(15)",
            "size": 15,
            "scale": null,
            "nullable": 0
          }
        },
        {
          "columnName": "Description",
          "templateVariables": {},
          "baseDomain": "CHAR",
          "domain": "CHAR(30)",
          "size": 30,
          "scale": null,
          "nullable": 0,
          "ordinal": 3,
          "cons": {
            "columnName": "Description",
            "baseDomain": "CHAR",
            "domain": "CHAR(30)",
            "size": 30,
            "scale": null,
            "nullable": 0
          }
        },
        {
          "columnName": "Size",
          "templateVariables": {},
          "baseDomain": "CHAR",
          "domain": "CHAR(18)",
          "size": 18,
          "scale": null,
          "nullable": 0,
          "ordinal": 4,
          "cons": {
            "columnName": "Size",
            "baseDomain": "CHAR",
            "domain": "CHAR(18)",
            "size": 18,
            "scale": null,
            "nullable": 0
          }
        },
        {
          "columnName": "Color",
          "templateVariables": {},
          "baseDomain": "CHAR",
          "domain": "CHAR(18)",
          "size": 18,
          "scale": null,
          "nullable": 0,
          "ordinal": 5,
          "cons": {
            "columnName": "Color",
```

```
            "baseDomain": "CHAR",
            "domain": "CHAR(18)",
            "size": 18,
            "scale": null,
            "nullable": 0
          }
        },
        {
          "columnName": "Quantity",
          "templateVariables": {},
          "baseDomain": "INTEGER",
          "domain": "INTEGER",
          "size": null,
          "scale": null,
          "nullable": 0,
          "ordinal": 6,
          "cons": {
            "columnName": "Quantity",
            "baseDomain": "INTEGER",
            "domain": "INTEGER",
            "size": null,
            "scale": null,
            "nullable": 0
          }
        },
        {
          "columnName": "UnitPrice",
          "templateVariables": {},
          "baseDomain": "NUMERIC",
          "domain": "NUMERIC(15, 2)",
          "size": 15,
          "scale": 2,
          "nullable": 0,
          "ordinal": 7,
          "cons": {
            "columnName": "UnitPrice",
            "baseDomain": "NUMERIC",
            "domain": "NUMERIC(15, 2)",
            "size": 15,
            "scale": 2,
            "nullable": 0
          }
        },
        {
          "columnName": "Photo",
          "templateVariables": {},
          "baseDomain": "LONG BINARY",
          "domain": "LONG BINARY",
          "size": null,
          "scale": null,
          "nullable": 1,
          "ordinal": 8,
          "cons": {
            "columnName": "Photo",
            "baseDomain": "LONG BINARY",
            "domain": "LONG BINARY",
            "size": null,
            "scale": null,
            "nullable": 1
          }
        }
      ],
      "consDBSchema": "GROUPO",
      "consDBCatalog": null,
      "primaryKeyColumns": [
        "ID"
      ],
      "foreignKeys": []
```

```
    }
  ]
```

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.4 ルール (mltemplate)

ルールは、どのテンプレートファイルをどのスキーマオブジェクトに適用するのかを指定します。プロジェクトファイル内でルールを手作業で定義した後、*mltemplate gen* を実行することで、ルールを実行します。

構文

```
"templateRules":
[
    {
       "ruleName":  "rule-name",
        "outFile":  "output-filename.sql",
       "tables":  [
                     table-name[, ...]
                  ],
       "templateVariables":  {
                variable-name:variable-value[, ...]
                }
        "templateFiles":[
                handlebars-filename.hbs"[, ...]
                ]
    }[,...]
]
```

パラメータ

rule-name ルールの名前。

output-filename.sql ルールによって生成される出力を保存するための SQL ファイル。*null* を指定すると、プロジェクトファイル内の最上位の outFile フィールドで定義されるデフォルトの出力ファイルに、ルールの出力が保存されます。個々のルールに 1 つの出力ファイルを指定すると、特に 1 つのルールのみを変更する必要があるときに、プロジェクトの管理が容易になります。

table-name スキーマ内のテーブルのうち、ルールが適用されるテーブル。グローバルルール (関連付けられたテーブルが存在しないルール) を作成するには、`null` を指定します。

variable-name:variable-value

Handlebars テンプレートファイル内で使用されるカスタム変数の定義。これらの定義は、ルールの実行時にテンプレートファイルに渡されます。

handlebars-filename.hbs ルールによって適用される Handlebars テンプレートファイルのリスト。

備考

ルールは、プロジェクトファイルに保存されたスキーマオブジェクトと、テンプレートファイル内で定義されたコンテンツを関連付けます。

テーブルルールは、templateRules フィールドを使用してテーブルオブジェクト内部で定義されます。

グローバルルールは、どのテーブルにも適用されないルールです。グローバルルールは、認証スクリプトなどの接続スクリプトを作成する際に有用です。グローバルルールを作成するには、*tables* フィールドを null に設定します。

例

Products テーブルのみに適用されるルールの例:

```
 "templateRules": [
    {
      "ruleName": "default",
      "outFile": null,
      "tables": [
        "Products"
      ],
      "templateVariables": {},
      "templateFiles": [
          "mobilink/cdb/sqla/basic/shadow_table.hbs",
          "mobilink/cdb/sqla/basic/trigger.hbs",
          "mobilink/cdb/sqla/basic/sync_script.hbs",
          "mobilink/cdb/sqla/basic/stored_procedure.hbs",
          "mobilink/cdb/sqla/basic/load.hbs"
      ]
    },
...
```

グローバルルールの例:

```
 { "ruleName": "cdb_report",
      "tables": null,
      "templateVariables": { },
      "templateFiles": [
          "mobilink/cdb/sqla/test/report.hbs"
      ]
    },
...
```

ルール内で使用される変数も定義するグローバルルールの例:

```
 { "ruleName": "subscription",
        "tables": null,
        "templateVariables": {
            "ml_host": "localhost",
            "ml_port": "2439"
        },
        "templateFiles": [
            "mobilink/rdb/sqla/subscription.hbs"
        ]
      },
...
```

プロジェクトファイル内のデフォルトルールの例:

```
{
  "outDir": null,
  "outFile": "generatedScripts.txt",
  "remoteDBType": "SQL Anywhere",
  "consDBType": "SQL Anywhere",
  "templateRoots": [
      "template"
  ],
  "templateVariables": {
      "scriptVersion": "V1",
      "consDBTableOwner": "GROUPO",
      "projectName": "sqla_demo"
  },
  "templateHelpers": [
      "sqlhelpers.js",
      "helpers/custom_helpers.js"
  ],
  "templateOutFiles": {},
  "templateRules": [
    {
      "ruleName": "default",
      "outFile": null,
      "tables": [
        "Products"
      ],
...
```

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.5 カスタム変数 (mltemplate)

カスタム変数はプロジェクトファイル内で定義し、テンプレートファイル内で使用します。

構文

```
"templateVariables": {
    "variable-name": "variable-value" [ , ... }
```

パラメータ

variable-name 変数の名前。

variable-value 変数の値。

備考

ほとんどのカスタム変数はプロジェクトファイル内で定義されます。ただし、-v オプションを指定して *mltemplate gen* を実行すると、生成時にカスタム変数を定義できます。-v オプションで定義される変数は、プロジェクトファイル内で定義される変数よりも優先されます。

プロジェクトファイル内では、カスタム変数の位置によって変数のタイプとスコープが指定されます。次の表に、プロジェクトファイル内で定義されるカスタム変数を優先順位の高い方から低い方に示します。

カスタム変数のタイプ	変数の位置
グローバル変数	プロジェクトファイルの先頭の templateVariables フィールドで定義される変数。このタイプのカスタム変数は、プロジェクトファイル内で定義されるすべての変数のうち、最高の優先順位を持ちます。このタイプの変数を上書きするには、-v オプションを指定して *mltemplate gen* を実行します。
ルール固有変数	templateVariables フィールドを使用してルールオブジェクト内部で定義される変数。
テーブル固有変数	templateVariables フィールドを使用してテーブルオブジェクト内部で定義される変数。
カラムオブジェクト変数	templateVariables フィールドを使用してカラムオブジェクト内部で定義される変数。

mltemplate gen を実行すると、変数がプロジェクトファイルから抽出され、入力として Handlebars テンプレートに渡されます。そのため、テンプレート内では、実際の値の代替として変数を使用できます。

Handlebars テンプレートに渡されるすべての変数を取得するには、-dump オプションを指定して *mltemplate gen* を実行します。

例

次の例では、-dump オプションを指定して *mltemplate gen* を実行し、Handlebars テンプレートに渡される変数を記載した JSON ファイルを生成します。

```
mltemplate gen project.json -dump
INFO: Applying template snapshot_download_scripts.hbs for table "Products" and
rule "default"
INFO: Template input variables dumped to file:
varDump_snapshot_download_scripts_Products.json
INFO: Applying template upload_scripts.hbs for table "Products" and rule "default"
INFO: Template input variables dumped to file:
varDump_upload_scripts_Products.json
INFO: Writing output file generatedScripts.txt
```

グローバル変数の定義の例:

```
{
  "outDir": null,
  "outFile": "generatedScripts.txt",
  "remoteDBType": "SQL Anywhere",
  "consDBType": "SQL Anywhere",
  "templateRoots": [
```

```
        "template"
    ],
    "templateVariables": {
     "scriptVersion": "V1",
     "consDBTableOwner": "GROUPO",
     "projectName": "sqla_demo"
   },
    "templateHelpers": [
        "sqlhelpers.js",
        "helpers/custom_helpers.js"
    ],
    ...
```

ルール内部で定義される変数の例:

```
{ "ruleName": "subscription",
        "tables": null,
        "templateVariables": {
      "ml_host": "localhost",
      "ml_port": "2439"
    },
        "templateFiles": [
            "mobilink/rdb/sqla/subscription.hbs"
        ]
      },
```

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]
スキーマ変数 (mltemplate) [338 ページ]

1.7.6 スキーマ変数 (mltemplate)

スキーマ変数は、データベーススキーマをプロジェクトにインポートするときに生成されます。スキーマ変数はテンプレートファイルで使用します。

mltemplate import を実行してスキーマ変数を生成し、プロジェクトファイル内の syncTables 配列にスキーマ変数を追加します。スキーマ変数は、インポートされるデータベースのスキーマに固有のものです。インポートされるデータベースのスキーマにスキーマ変数が適用されない場合、そのスキーマ変数の値は空と定義されます。

Handlebars テンプレートに渡されるすべての変数を取得するには、-dump オプションを指定して *mltemplate gen* を実行します。テンプレートが実行されるたびに、テンプレートに渡されたスキーマ変数とカスタム変数を含む JSON ファイルが *mltemplate gen* によって作成されます。

次の表に、使用可能なデフォルトのスキーマ変数を示します。

スキーマ変数	説明
consDBTableName	統合データベーステーブルの名前。

スキーマ変数	説明
columns	カラムオブジェクトの配列。
primaryKeyColumns	カラム名の配列。この配列は、Handlebars テンプレートに渡される前に、カラムオブジェクトの配列に展開されます。
foreignKeys	外部キーオブジェクトの配列で、次のフィールドがあります。 • *foreignKeyName* • *primaryTableSchema* • *primaryTableName* • *columns* columns フィールドには、*primaryColumnName* と *foreignColumnName* を含む配列が収録されます。
columnNames	カラム名のカンマ区切りのリスト。
consDBColumnNames	統合カラム名のカンマ区切りのリスト。
primaryKeyColumnNames	すべてのプライマリキーカラム名のカンマ区切りのリスト。
consDBPrimaryKeyColumnNames	統合データベース内にあるすべてのプライマリキーカラム名のカンマ区切りのリスト。
nonPrimaryKeyColumns	すべての非プライマリキーカラム名を含む配列。
nonPrimaryKeyColumnNames	すべての非プライマリキーカラム名のカンマ区切りのリスト。
consDBNonPrimaryKeyColumnNames	統合データベース内にあるすべての非プライマリキーカラム名のカンマ区切りのリスト。
dotIfDefined`name`	テーブルの所有者、スキーマ、またはカタログの名前 (インポートされるデータベースのタイプに依存)。 別の名前を指定するには、dotIfDefined 変数を編集します。指定された名前 (ドット付き) は出力のテーブル名の前に表示されます。
nullClause { *null* | *not null* }	この変数を使用すると、Handlebars テンプレートの nullClause フィールドの値を参照できます。CREATE TABLE 文でカラム宣言を記述するときにも有用です。 プロジェクトファイル内のすべてのカラム定義には nullClause フィールドがあります。このフィールドの値は、カラムで null 値が許可される場合は文字列 *null*、許可されない場合は *not null* です。

ルールスキーマ変数	説明
ruleTables	現在のルールの一部であるテーブル名の配列。
ruleName	ルールの名前。

ルールスキーマ変数	説明
syncTables	プロジェクトファイル内の syncTables 配列の完全な内容。

スキーマ変数定義の拡張

スキーマ変数の定義は、*%SQLANY17%*¥MobiLink¥JavaScript¥prepare.js 内の JavaScript 関数に保存されます。これらのスキーマ変数を拡張するには、拡張関数を含む JavaScript ファイルを作成した後、そのファイルをプロジェクトファイル内の templateHelpers 配列に追加します。mltemplate は常にデフォルトディレクトリ内でこの JavaScript ファイルを検索するため、このファイルに `prepare.js` という名前を付けないでください。

スキーマ変数を拡張するには、次のいずれかの拡張関数を使用します。

拡張関数	注記
`customGlobalVars( project )`	この関数は、defaultGlobalVars 関数を置き換えます。テーブルスキーマ変数を引き続き使用できるように、defaultGlobalVars 関数の呼び出しを JavaScript ファイルに追加してください。
`customTableVars( project, table )`	この関数は、defaultTableVars 関数を上書きします。テーブルスキーマ変数を引き続き使用できるように、defaultTableVars 関数の呼び出しを JavaScript ファイルに追加してください。
`customRuleVars( project, rule )`	この関数は、defaultRuleVars 関数を上書きします。テーブルスキーマ変数を引き続き使用できるように、defaultRuleVars 関数の呼び出しを JavaScript ファイルに追加してください。

例

次の JavaScript コードは customGlobalVars 関数を上書きして、myCustomVar という新しい変数を追加します。

```
function customGlobalVars( model )
{
    var globalVars = defaultGlobalVars( model );
    globalVars.myCustomVar = "ABCD";
    return globalVars;
}
```

次の例には 4 つのスキーマ変数があります。

```
...
  "syncTables":
  [
    {
      "tableName": "Products",
      "consDBTableName": "Products",
      "templateVariables": {},
      "columns":
      [
        {
          "columnName": "ID",
          "templateVariables": {},
```

```
            "baseDomain": "INTEGER",
            "domain": "INTEGER",
            "size": null,
            "scale": null,
            "nullable": 0,
            "ordinal": 1,
            "cons":
              {
              "columnName": "ID",
              "baseDomain": "INTEGER",
              "domain": "INTEGER",
              "size": null,
              "scale": null,
              "nullable": 0
              }
          },
          ...
        ],
      "consDBSchema": "GROUPO",
    "consDBCatalog": null,
    "primaryKeyColumns": [ "ID" ],
    "foreignKeys": []
  }
  ]
}
```

関連情報

カスタム変数 (mltemplate) [336 ページ]
Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.7 Handlebars テンプレートとヘルパー (mltemplate)

mltemplate では、通常は Mobile Link 同期スクリプトが記載された Handlebars テンプレートが使用されます。テンプレートには同期追跡スキーマ (シャドウテーブル、システムプロシージャ、トリガの定義など) も記載できます。

mltemplate では、SAP HANA または SAP SQL Anywhere の統合データベース用の同期スクリプトを作成するためのデフォルトの Handlebars テンプレートが提供されます。これらのデフォルトテンプレートは、*%SQLANY17%*¥MobiLink¥MLTemplate¥Templates ディレクトリにあります。デフォルトテンプレートの説明を表示するには、テキストエディタでテンプレートファイルを開きます。

同期プロジェクト用のテンプレートを作成するときには、これらの Handlebars テンプレートをガイドとして使用します。次のような目的のカスタマイズしたテンプレートは、ユーザが作成する必要があります。

- SAP HANA または SAP SQL Anywhere の統合データベース用のデフォルトテンプレートを補足するため。一般的に、同期プロジェクトでは、デフォルトテンプレートで使用できない機能が必要とされることがあります。たとえば、クライアントデータベースにダウンロードされるデータをフィルタするには、独自のテンプレートを作成する必要があります。
- SQL Anywhere リモートデータベース、およびサポートされる他のタイプの統合データベースで使用するため。

独自のテンプレートを作成するには、デフォルトテンプレートをガイドとして使用します。

1. *%SQLANY17%*¥MobiLink¥MLTemplate¥Templates から読取/書込権限のある場所にデフォルトファイルをコピーし、ファイルを編集した後、ファイル名を変更します。ファイル名を変更すれば、mltemplate はデフォルトバージョンを

使い続けることができます。プロジェクトファイル内の templateRoots 配列でデフォルトディレクトリが参照されるかどうかにかかわりなく、mltemplate は常にデフォルトのテンプレートディレクトリを検索します。

2. プロジェクトファイル内の最上位の templateRoots 配列にテンプレートディレクトリを追加します。
3. プロジェクトファイル内のルール用の templateFiles 配列にテンプレートファイルを追加します。

テンプレートでは次の操作が可能です。

- データベースオブジェクトを作成するための Handlebars スクリプト (同期スクリプトなど) を収録する。
- データベーススキーマのインポート時にプロジェクトファイルに追加されるスキーマ変数を使用する。
- プロジェクトファイル内の templateVariable 配列でユーザが定義するカスタム変数を使用する。
- Handlebars で提供される組込ヘルパーを使用する。
- SQL 文を書式設定するために `sqlhelpers.js` で定義されるデフォルトヘルパーを使用する。
- ユーザが JavaScript ファイルで作成するカスタムヘルパーを使用し、プロジェクトファイル内の templateHelpers 配列にカスタムヘルパーを追加する。

デフォルト **Handlebars** テンプレートヘルパー (`sqlhelpers.js`)

デフォルトヘルパーは、*%SQLANY17%*`¥MobiLink¥MLTemplate¥Helpers` にある `sqlhelpers.js` で定義されます。プロジェクトファイル内の templateHelpers 配列には、`sqlhelpers.js` への参照が記述されます。この参照にパスは含まれません。mltemplate は常にデフォルトの `Helpers` ディレクトリでファイルを検索するため、パスの指定は不要であるためです。

次の表に、`sqlhelpers.js` で定義されるデフォルトヘルパーを示します。

Handlebar ヘルパー	説明
commaList	入力として配列を受け取り、各要素のブロックテキストをカンマで分離します。カラムパラメータのリストを構成するときには、このヘルパーが有用です。
orList	入力として配列を受け取り、各要素のブロックテキストを論理 OR 演算子によって分離します。カラムのリストに基づいて WHERE 句述部を構成するときには、このヘルパーが有用です。
andList	入力として配列を受け取り、各要素のブロックテキストを論理 AND 演算子によって分離します。カラムのリストに基づいて WHERE 句述部を構成するときには、このヘルパーが有用です。
withVar	入力として配列を受け取りますが、ブロックの別の書式は追加しません。Handlebars ハッシュ属性でフィルタリングするときには、このヘルパーが有用です。

これらの各デフォルトヘルパーでは、2 つの Handlebars ハッシュ属性 Value および NotValue を任意で使用できます。これらのハッシュ属性を使用すると、カラムプロパティまたはカスタム変数に基づいて、カラムまたはテーブルに単純なフィルタを適用できます。

ハッシュ属性 (ヘルパー andList、orList、commaList、withVar で使用)	説明
Value="string"	指定した値を持つカラムを選択します。
NotValue="string"	指定した値を持たないカラムを選択します。

カスタム Handlebars ヘルパー

独自のカスタムヘルパーを作成する手順:

1. JavaScript ファイル内で、ヘルパーを作成し、Handlebars.registerHelper の呼び出しをファイルに追加することで、ヘルパーを登録します。
   デフォルトヘルパーファイルと同じ名前 (`sqlhelpers.js`) をカスタムヘルパーファイルに付けないでください。
2. プロジェクトファイル内の templateHelpers 配列にヘルパーファイルを追加します。

例

ハッシュ属性の例

次の例では、customVar という templateVariable を持つカラム名のリストを作成します。

```
{{commaList columns varName="customVar"}}
 {{columnName}}
{{/commaList}}
```

次の例では、customVar という templateVariable が値 1234 に設定されたカラムのみを選択します。

```
{{commaList columns varName="customVar" value="1234" }}
 {{columnName}}
 {{/commaList}}
```

次の例では、customVar というテンプレート変数が値 1234 に設定されていないカラムのみを選択します。

```
{{commaList columns varName="customVar" notValue="1234" }}
{{columnName}}
{{/commaList}}
```

カスタムヘルパーの例

次のコードは、カスタマイズした Handlebars ヘルパーファイルの例です。

```
Handlebars.registerHelper( "spaceList", function( items, options ) {
    var ret = "";
    for( var i=0; i < items.length; i++ ) {
        var item = items[i];
        if( filterByOptions( item, options ) ) continue;
        if( i != 0 ) {
            ret += " ";
        }
        ret += options.fn( item );
    }
    return ret;
});
```

テンプレートの例

次の例はカスタマイズした Handlebars テンプレートファイルであり、同期スクリプトが記述されています。

```
{!-- This is a Handlebars template.  See http://handlebarsjs.com for
documentation. --}}
-----------------------------------------------------------------
-- APPLICATION:    Data Synchronization
--
-- Synchronization scripts for: {{consDBTableName}}
--
-- AUTHOR:     Generated
--
-- Notes:
-----------------------------------------------------------------
CALL ml_add_table_script (
    '{{scriptVersion}}',
    '{{consDBTableName}}',
    'download_cursor',
    'CALL {{scriptVersion}}_{{consDBTableName}}_DC (
        {ml s.last_table_download}, {ml s.username} )'
);
CALL ml_add_table_script (
    '{{scriptVersion}}',
    '{{consDBTableName}}',
    'download_delete_cursor',
    'CALL {{scriptVersion}}_{{consDBTableName}}_DDC (
        {ml s.last_table_download}, {ml s.username} )'
);
CALL ml_add_table_script (
    '{{scriptVersion}}',
    '{{consDBTableName}}',
    'upload_insert',
    'CALL {{scriptVersion}}_{{consDBTableName}}_UI (
     {{#commaList columns}}{ml r.{{columnName}} }{{/commaList}})'
);
CALL ml_add_table_script (
    '{{scriptVersion}}',
    '{{consDBTableName}}',
    'upload_delete',
    'CALL {{scriptVersion}}_{{consDBTableName}}_UD (
     {{#commaList primaryKeyColumns}}{ml r.{{columnName}} }{{/commaList}})'
);
CALL ml_add_table_script (
    '{{scriptVersion}}',
    '{{consDBTableName}}',
    'upload_update',
    'CALL {{scriptVersion}}_{{consDBTableName}}_UU (
     {{#commaList nonPrimaryKeyColumns}}{ml r.{{columnName}} }{{/commaList}},
     {{#commaList primaryKeyColumns}}{ml r.{{columnName}} }{{/commaList}}
     )'
);
```

次の例はカスタマイズした Handlebars テンプレートであり、トリガ定義が記述されています。

```
{{!-- This is a Handlebars template.  See http://handlebarsjs.com for
documentation. --}}
-----------------------------------------------------------------
-- APPLICATION:    Data Synchronization
--
-- Trigger script for: {{consDBTableName}}
--
-- AUTHOR:     Generated
--
-- Notes:
-----------------------------------------------------------------
```

```
CREATE OR REPLACE TRIGGER {{consDBTableOwner}}.TRIG_INSERT_{{consDBTableName}}
AFTER INSERT, UPDATE ON {{consDBTableName}}
REFERENCING NEW AS new_row
FOR EACH ROW
BEGIN
    INSERT INTO {{consDBTableName}}_SHADOW ({{primaryKeyColumnNames}},
        OPERATION, LAST_MODIFIED)
    ON EXISTING UPDATE
    VALUES ({{#commaList primaryKeyColumns}}new_row.{{columnName}} {{/
commaList}},
        'U', CURRENT_TIMESTAMP);
 END;
CREATE OR REPLACE TRIGGER {{consDBTableOwner}}.TRIG_DELETE_{{consDBTableName}}
AFTER DELETE ON {{consDBTableName}}
REFERENCING OLD AS old_row
FOR EACH ROW
BEGIN
   UPDATE {{consDBTableName}}_SHADOW as SHADOW
   SET OPERATION = 'D'
   WHERE {{#andList primaryKeyColumns}}SHADOW.{{columnName}} = old_row.
{{columnName}}
   {{/andList}}
END;
```

関連情報

http://handlebarsjs.com/

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]
ルール (mltemplate) [334 ページ]
Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.8 出力ファイル (mltemplate)

mltemplate から生成されるコンテンツを編成するために、テンプレート別および/またはルール別に出力をグループ化します。

デフォルトでは、mltemplate から生成されるすべての出力が単一のファイルに保存されます。すべての出力が単一のファイルに保存される場合、ファイルが非常に大きくなるため、別の生成処理からの出力との比較が困難になる可能性があります。増分変更を計画する場合や、統合データベースとリモートデータベースの両方の SQL を生成する場合にも、単一のファイルは不便です。多くの場合、複数の出力ファイルにコンテンツを生成するほうが適切です。

出力ファイルを保存する専用のディレクトリを使用するには、プロジェクトファイル内で outDir フィールドにディレクトリを指定します。outDir のデフォルト値は null です。この場合、出力ファイルが mltemplate の作業ディレクトリに保存されます。

出力ファイルは次の方法でカスタマイズします。

出力ファイルのレベル	説明
グローバル出力ファイル	プロジェクトファイル内の最上位の outFile フィールドに別の値を指定することで、デフォルトの出力ファイルの名前を変更します。下記のいずれかの方法で別のファイルを指定しない限り、デフォルトでは、生成されるすべてのコンテンツがこの SQL ファイルに保存されます。デフォルト値は `generatedScripts.txt` です。 より詳細な情報をプロジェクトファイル (下記) に入力するか、*mltemplate gen* に -o オプションを追加することで、指定した値を生成時に上書きできます。
テンプレート別の出力ファイル	出力の生成に使用されるテンプレート別に、出力をグループ化します。テンプレートファイルとその出力ファイルを指定するには、templateOutFiles オブジェクトを使用します。 templateOutFiles オブジェクトで指定されたテンプレートファイルごとに、テンプレートの実行によって生成されるすべての出力を含む出力ファイルが作成されます。 たとえば、プロジェクト内のすべてのテーブル用のシャドウテーブル定義を生成して、その定義を 1 つのファイルに保存するときには、この方式が便利です。 テンプレート別の出力ファイルを定義するには、次の構文を使用します。 `"templateOutFiles": { { "templateFileName1": "outputFile1" [, ...]},`
ルール別の出力ファイル	ルールに対して outFile フィールドを定義することで、ルール別の出力ファイルを指定します。 outFile フィールドが定義された各ルールに対して、出力ファイルが作成されます。このルールは他のルールよりも優先されます。

例

次の例では、デフォルトの出力ファイルの名前を mobiLinkScripts.sql に設定します。

```
...
 "outDir": null,
 "outFile" : "mobiLinkScripts.sql",
...
```

1.7.9 チュートリアル: Moble Link テンプレートシステムユーティリティ (mltemplate)

SQL Anywhere の統合データベースとクライアントデータベースを使用する同期プロジェクトを mltemplate によって作成します。

前提条件

mltemplate のサンプル (*%SQLANYSAMP17%*¥Mobilink¥MLTemplate ディレクトリにあります)。このチュートリアルでは、このサンプルの一部を使用します。サンプルに付属の read me ファイルを参照することをおすすめします。

SQL Anywhere 17 Demo データベース (demo.db)。

SYS_AUTH_RESOURCE_ROLE - 統合データベースとクライアントデータベースにデータベースオブジェクトを作成するために必要です。

(オプション) Apache Ant ビルドツール。このチュートリアルでは、Apache Ant ツールを使用して統合データベースとクライアントデータベースを作成します。なお、サンプルの read me ファイル内の指示に従って、Apache Ant ツールを使用しないでデータベースを作成することもできます。Apache Ant は SQL Anywhere には付属していませんが、http://ant.apache.org/ から入手できます。

コンテキスト

mltemplate の使用方法を習得する最適な方法は、サンプルプロジェクトを研究/試行することです。このチュートリアルでは、mltemplate のサンプルをサンプルプロジェクトとして使用します。mltemplate のサンプルは完全なプロジェクトであるため、すべてのレッスンでサンプルの操作を使用するわけではなく、一部のレッスンでサンプルのオブジェクトを取り扱います。

mltemplate のサンプルは、SQL Anywhere サンプルデータベースを統合データベースとして使用する同期プロジェクトであり、Products テーブルと Customers テーブルを含むクライアントデータベースを作成します。このサンプルで使用される単純な同期ロジックでは、リモートデータベース (または統合データベース) 内のテーブルに対する変更が統合データベース (またはリモートデータベース) に複製されます。この同期ロジックには、競合解決、特定の形式のデータパーティショニングは含まれず、プライマリキー値の一意性も保証されません。

このセクションの内容:

このレッスンでは、mltemplate のサンプル用の統合データベースとクライアントデータベースを構築し、それらのデータベース間で同期のシミュレーションを行います。

レッスン 5: クリーンアップ [357 ページ]
チュートリアルをコンピュータから削除します。

関連情報

プロジェクトファイル (mltemplate) [330 ページ]
http://handlebarsjs.com/
Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.9.1 レッスン 1: プロジェクトファイルの作成

同期プロジェクト用のディレクトリ構造を設定し、mltemplate を使用してプロジェクトファイルを作成します。

前提条件

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

mltemplate のサンプルには、保持する必要のある完成したプロジェクトファイルが収録されています。このレッスンでは、別のプロジェクトファイルを作成する最初のステップを紹介します。

手順

1. 作業用に mltemplate サンプルのコピーを作成します。*%SQLANYSAMP17%*¥Mobilink¥MLTemplate ディレクトリ (配下のサブディレクトリとファイルを含む) を、読取/書込権限のある場所にコピーします。

   たとえば、ファイルを C:¥ にコピーします。

   MLTemplate ディレクトリには次のサブディレクトリがあります。

   - Helpers ディレクトリには、カスタマイズ済みの Handlebars ヘルパーファイルがあります。
   - Templates ディレクトリには、カスタマイズ済みの Handlebars テンプレートファイルがあります。

MLTemplate ディレクトリには、プロジェクトファイルがあります。そのため、これが作業ディレクトリです。mltemplate は常にこのディレクトリから実行してください。その理由は、mltemplate が実行されるディレクトリを基準として、mltemplate の相対パスが決定されるためです。

2. プロジェクトファイルを作成します。コマンドプロンプトで、MLTemplate ディレクトリから次の mltemplate コマンドを実行します (すべてを 1 行に入力します)。

```
mltemplate new myproject.json
-t Template -js Helper/custom_helpers.js
-s V1 -remType sqla -consType sqla
```

このコマンドによって、次の設定を持つプロジェクトファイルが作成されます。

- Handlebars テンプレートファイルのディレクトリが、サンプルのカスタマイズ済みテンプレートディレクトリ (c:¥MLTemplate¥Template) に設定されます。プロジェクトファイル内では、このディレクトリが TemplateRoots 配列に追加されます。
  c:¥MLTemplate¥Template 内のファイルは、*%SQLANY17%*¥MobiLink¥MLTemplate¥Template からデフォルトの Handlebars テンプレートをコピーし、(mltemplate でデフォルトファイルが使用されないように) テンプレート名を変更した後、この同期プロジェクト用にテンプレートをカスタマイズすることによって作成されたものです。
  サンプルのカスタマイズ済み Handlebars ヘルパーファイル (*%SQLANYSAMP17%*¥Mobilink¥MLTemplate¥Helper¥custom_helpers.js) が、プロジェクトで使用されるヘルパーファイルのリストに追加されます。プロジェクトファイル内では、このヘルパーファイルが templateHelpers 配列 (この配列にはデフォルトの sqlhelpers.js ファイルも記載されています) に追加されます。
- 統合データベースとクライアントデータベースのデータベースタイプが両方とも SQL Anywhere であることが指定されます。
- Moble Link 同期スクリプトのバージョンが V1 であることが指定されます。

作成されるプロジェクトファイルには、プロジェクトに関する環境情報 (クライアントデータベースと統合データベースのデータベースタイプ、Handlebars テンプレートディレクトリと Handlebars ヘルパーファイルの場所など) が記載されています。

3. テキストエディタで myproject.json を開き、(プロジェクトファイルの先頭にある) グローバル配列 templateVariables を次の配列で置き換えます。

```
"templateVariables": {
      "scriptVersion": "V1",
      "consDBTableOwner": "GROUPO",
      "projectName": "sqla_demo"
  },
```

4. プロジェクトファイル内で、(default という名前の) デフォルトルールを見つけて、デフォルトの Handlebars テンプレートをサンプル内のカスタマイズ済みテンプレートで置き換えます。

SQL Anywhere 統合データベースの場合、デフォルトルールによって 2 つのデフォルトテンプレートファイル (snapshot_download_scripts.hbs および upload_scripts.hbs) が指定されます。これらのデフォルトテンプレートは、デフォルトのテンプレートディレクトリ *%SQLANY17%*¥MobiLink¥MLTemplate¥Templates にあります。

```
"templateRules": [
    {
      "ruleName": "default",
      "outFile": null,
      "tables": [],
      "templateVariables": {},
      "templateFiles": [
        "snapshot_download_scripts.hbs",
        "upload_scripts.hbs"
```

```
      ]
    },
```

mltemplate のサンプルではカスタマイズ済みテンプレートのみが使用されるため、デフォルトファイルを削除して、サンプル内のカスタマイズ済みファイルで置き換える必要があります。

デフォルトルール内のデフォルトテンプレートファイルへの参照を、次のカスタマイズ済み Handlebars テンプレートファイルで置き換えてください。

```
...
      "templateFiles": [
        "template/mobilink/cdb/sqla/basic/shadow_table.hbs",
        "template/mobilink/cdb/sqla/basic/trigger.hbs",
        "template/mobilink/cdb/sqla/basic/sync_script.hbs",
        "template/mobilink/cdb/sqla/basic/stored_procedure.hbs",
        "template/mobilink/cdb/sqla/basic/load.hbs"
      ]
...
```

mltemplate は、カスタマイズ済みテンプレートを検索する前に、常にデフォルトのテンプレートディレクトリを検索します。したがって、デフォルトのテンプレートファイルと同じ名前を、カスタマイズ済みテンプレートファイルに決して指定しないでください。

5. デフォルトルール conn_script を次のグローバルルールで置き換えます。

```
{ "ruleName": "cdb_report",
      "tables": null,
      "templateVariables": { },
      "templateFiles": [
          "template/mobilink/cdb/sqla/test/report.hbs"
      ]
    },
    {
        "ruleName": "rdb_tables",
        "outFile": null,
        "tables": [
            "Products",
            "Customers"
        ],
        "templateVariables": {},
        "templateFiles": [
            "template/mobilink/rdb/sqla/table.hbs"
        ]
      },
      { "ruleName": "publication",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "template/mobilink/rdb/sqla/publication.hbs"
        ]
      },
      { "ruleName": "subscription",
        "tables": null,
        "templateVariables": {
            "ml_host": "localhost",
            "ml_port": "2439"
        },
        "templateFiles": [
            "template/mobilink/rdb/sqla/subscription.hbs"
        ]
      }
```

myproject.json 内の templateRules 配列は、次のようなコードになるはずです。

```
...
  "templateRules": [
    {
      "ruleName": "default",
      "outFile": null,
      "tables": [
        "Products"
      ],
      "templateVariables": {},
      "templateFiles": [
          "template/mobilink/cdb/sqla/basic/shadow_table.hbs",
          "template/mobilink/cdb/sqla/basic/trigger.hbs",
          "template/mobilink/cdb/sqla/basic/sync_script.hbs",
          "template/mobilink/cdb/sqla/basic/stored_procedure.hbs",
          "template/mobilink/cdb/sqla/basic/load.hbs"
      ]
    },
    { "ruleName": "cdb_report",
      "tables": null,
      "templateVariables": { },
      "templateFiles": [
          "template/mobilink/cdb/sqla/test/report.hbs"
      ]
    },
    {
        "ruleName": "rdb_tables",
        "outFile": null,
        "tables": [
            "Products"
        ],
        "templateVariables": {},
        "templateFiles": [
            "template/mobilink/rdb/sqla/table.hbs"
        ]
      },
      { "ruleName": "publication",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "template/mobilink/rdb/sqla/publication.hbs"
        ]
      },
      { "ruleName": "subscription",
        "tables": null,
        "templateVariables": {
            "ml_host": "localhost",
            "ml_port": "2439"
        },
        "templateFiles": [
            "template/mobilink/rdb/sqla/subscription.hbs"
        ]
      },
      { "ruleName": "rdb_report",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "template/mobilink/rdb/sqla/test/report.hbs"
        ]
      },
      { "ruleName": "rdb_insert",
        "tables": null,
        "templateVariables": { },
        "templateFiles": [
            "template/mobilink/rdb/sqla/test/insert.hbs"
        ]
```

```
      }
  ],
  "syncTables": [
...
```

次のステップ

次のレッスンに進みます。

1.7.9.2 レッスン 2: データベーススキーマのインポート

SQL Anywhere サンプルデータベースから `myproject.js` プロジェクトファイルに、2 つのテーブルのデータベーススキーマをインポートします。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

手順

1. 次のコマンド (1 行に入力) を実行して、SQL Anywhere サンプルデータベースのデータベーススキーマをインポートします。

   ```
   mltemplate import myproject.json
   -jdbc "jdbc:sqlanywhere:DSN=SQL Anywhere 17 Demo;Password=sql"
   -t Products -t Customers
   ```

   テーブル Products と Customers のテーブル定義、およびこれらのテーブルのスキーマ変数が、`myproject.json` ファイルの syncTables 配列に追加されます。
2. テキストエディタで `myproject.json` を開き、SyncTables 配列に 2 つのテーブルのスキーマがあることを確認します。

   各テーブル内には、カラムの配列があります。各カラムオブジェクトには、リモートスキーマのプロパティフィールドとともに、統合スキーマのプロパティフィールドを持つ cons フィールドがあります。

次の例に、Products テーブルの ID カラムのプロパティを示します。

```
"syncTables": [
    {
      "tableName": "Products",
      "consDBTableName": "Products",
      "templateVariables": {},
      "columns": [
        {
          "columnName": "ID",
          "templateVariables": {},
          "baseDomain": "INTEGER",
          "domain": "INTEGER",
          "size": null,
          "scale": null,
          "nullable": 0,
          "ordinal": 1,
          "cons": {
            "columnName": "ID",
            "baseDomain": "INTEGER",
            "domain": "INTEGER",
            "size": null,
            "scale": null,
            "nullable": 0
          }
        },
...
```

プロジェクトファイルの内容と、サンプルの `project.json` ファイルの内容を確認できます。

次のステップ

次のレッスンに進みます。

関連情報

Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.9.3 レッスン 3: ルール、変数、および Handlebars テンプレート間の相互作用に関する学習

このレッスンでは、mltemplate のサンプルで提供されるカスタマイズ済み Handlebars テンプレートについて説明します。

コンテキスト

mltemplate のサンプルには、統合データベースとリモートデータベースのためのカスタマイズ済み Handlebars テンプレートが用意されています。各テンプレートファイルの説明については、サンプルの readme.txt ファイルを参照してください。

Handlebars テンプレートには次のタイプの変数を記述できます。

1. スキーマ変数 - データベーススキーマのインポート時に、プロジェクトファイルに追加されます。パブリケーションテンプレートでは、syncTables および tableName がスキーマ変数の例です。
2. カスタム変数 - プロジェクトファイル内の templateVariables オブジェクトに手作業で追加します。パブリケーションテンプレートでは、projectName がカスタム変数の例です。
3. カスタマイズ済み Handlebars ヘルパー。mltemplate のサンプルでは、カスタマイズ済み Handlebars ヘルパーが `MLTemplates¥helper¥custom_helpers.js` にあります。パブリケーションテンプレートでは、commaList がカスタマイズ済みヘルパーの例です。

手順

1. テキストエディタで `myproject.json` を開き、publication ルールを見つけます。

```
 {
      "ruleName": "publication",
      "tables": null,
      "templateVariables": {},
      "templateFiles": [
        "mobilink/rdb/sqla/publication.hbs"
      ],
      "outFile": null
    },
```

   tables フィールドが null に設定されているため、このルールはグローバルルールです。つまり、プロジェクトファイル内で定義されたすべてのテーブルにルールが適用されます。この publication ルールは、プロジェクトファイル内のテーブルに Handlebars テンプレートファイル `publication.hbs` を適用します。

2. テキストエディタで `publication.hbs` を開きます。

   次のコードに、パブリケーションテンプレート (`publication.hbs`) の内容を示します。

```
-- create a synchronization profile and other artifacts for the client
-- Publication
DROP PUBLICATION IF EXISTS  PUB_{{projectName}}
GO
CREATE PUBLICATION PUB_{{projectName}}
(
    {{#commaList syncTables}}TABLE {{tableName}}{{/commaList}}
)
GO
```

3. 次のコマンドを実行することで、出力を生成し、デフォルトファイル `generatedScripts.sql` に出力を保存します。

```
mltemplate gen myproject.json
```

   mltemplate gen を実行すると、プロジェクトファイル内の各ルールが実行されます。プロジェクトファイル、テンプレートファイル、ヘルパーファイルから変数が収集されて、テンプレートに渡されるため、それらの変数をテンプレート内で使用できます。

   次のコードに、Publication テンプレートから Products テーブル用に生成された SQL 出力を示します。

```
DROP PUBLICATION IF EXISTS  PUB_sqla_demo
GO
CREATE PUBLICATION PUB_sqla_demo
```

```
(
    TABLE Products
)
GO
```

次のステップ

次のレッスンに進みます。

関連情報

ルール (mltemplate) [334 ページ]
Handlebars テンプレートとヘルパー (mltemplate) [341 ページ]
Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.9.4 レッスン 4: mltemplate のサンプル用の統合データベースとクライアントデータベースの構築

このレッスンでは、mltemplate のサンプル用の統合データベースとクライアントデータベースを構築し、それらのデータベース間で同期のシミュレーションを行います。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

このチュートリアルの冒頭に記載されているロールと権限を持っている必要があります。

コンテキスト

mltemplate のサンプルに収録された Ant ビルドツールと `build.xml` ファイルを使用して、統合データベースとクライアントデータベースを構築します。次に、それらのデータベース間で同期のシミュレーションを行います。build.xml ファイルは Ant に対して mltemplate gen を実行するように指示するため、ユーザ自身がこのコマンドを実行する必要はありません。

mltemplate のサンプルの readme ファイルには、Ant を使用しないで同じ操作を実行する方法が記載されています。

mltemplate の実行に使用したのと同じディレクトリから、Ant を実行してください。

このレッスンでは、ユーザが作成したプロジェクトファイルではなく、mltemplate に付属のプロジェクトファイルを使用します。

手順

1. 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。

   ```
   ant rdb
   ```

   ML_USER_1 という同期ユーザに対してクライアントデータベースが作成されます。別のユーザに対してリモートデータベースを作成するには、MLUSER 環境変数を変更して、コマンドを再実行します。
2. 次のコマンドを実行して、統合データベース内に同期オブジェクトを作成します。

   ```
   ant cdb
   ```

   デフォルトの DBA ユーザが SQL Anywhere サンプルデータベースに接続して、必要な同期設定スクリプトとオブジェクトを追加します。
3. 統合データベースに対して Mobile Link サーバを起動します。

   ```
   ant mobilink
   ```
4. クライアントデータベースを同期します。

   ```
   ant sync
   ```
5. 次のコマンドを実行して、統合データベースとクライアントデータベースのテーブルを検証します。

   ```
   ant report
   ```

   このサンプルの同期スクリプトは単純であり、データのパーティショニングは実行しません。リモートデータベースで変更が行われた場合に、競合を回避する手段はありません。

次のステップ

次のレッスンに進みます。

関連情報

ルール (mltemplate) [334 ページ]
Mobile Link テンプレートユーティリティ (mltemplate) [325 ページ]

1.7.9.5 レッスン 5: クリーンアップ

チュートリアルをコンピュータから削除します。

手順

1. このチュートリアルのために実行しているアプリケーションのすべてのインスタンスを閉じます。
2. mltemplate のサンプルファイルを保存するために作成したディレクトリを削除します。
3. 次のコマンドを実行して、サンプルデータベースを消去し、サンプルデータベースの新しいコピーを元のオブジェクトおよびデータとともに作成します。

   ```
   newdemo "%SQLANYSAMP17%¥demo.db"
   ```

   プロンプトが表示されたら、既存のファイルをすべて消去することを選択します。

   チュートリアルがコンピュータから削除されます。

結果

mltemplate のチュートリアルを完了しました。

1.8 このマニュアルの印刷、再生、および再配布

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

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

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

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

アクセシビリティ

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

ジェンダーニュートラルな表現

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

インターネットハイパーリンク

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

go.sap.com/registration/contact.html