SYBASE iAnywhere

Mobile Link
クライアント管理

改訂 2007 年 3 月

版権と商標

目次

はじめに

このマニュアルの内容

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。

対象読者

このマニュアルは、情報システムに同期を追加したいと考えているユーザを対象としています。

始める前に

Mobile Link と他の同期／レプリケーション・テクノロジの比較については、「データ交換テクノロジの概要」『SQL Anywhere 10 - 紹介』を参照してください。

SQL Anywhere のマニュアル

このマニュアルは、SQL Anywhere のマニュアル・セットの一部です。この項では、マニュアル・セットに含まれる各マニュアルと使用法について説明します。

SQL Anywhere のマニュアル

SQL Anywhere の完全なマニュアルは、各マニュアルをまとめたオンライン形式とマニュアル別の PDF ファイルで提供されます。いずれの形式のマニュアルも、同じ情報が含まれ、次のマニュアルから構成されます。

- **『SQL Anywhere 10 - 紹介』** このマニュアルでは、データの管理および交換機能を提供する包括的なパッケージである SQL Anywhere 10 について説明します。SQL Anywhere を使用すると、サーバ環境、デスクトップ環境、モバイル環境、リモート・オフィス環境に適したデータベース・ベースのアプリケーションを迅速に開発できるようになります。
- **『SQL Anywhere 10 - 変更点とアップグレード』** このマニュアルでは、SQL Anywhere 10 とそれ以前のバージョンに含まれる新機能について説明します。
- **『SQL Anywhere サーバ - データベース管理』** このマニュアルでは、SQL Anywhere データベースの実行、管理、設定について説明します。管理ユーティリティとオプションのほか、データベース接続、データベース・サーバ、データベース・ファイル、バックアップ・プロシージャ、セキュリティ、高可用性、Replication Server を使用したレプリケーションについて説明します。
- **『SQL Anywhere サーバ - SQL の使用法』** このマニュアルでは、データベースの設計と作成の方法、データのインポート・エクスポート・変更の方法、データの検索方法、ストアド・プロシージャとトリガの構築方法について説明します。
- **『SQL Anywhere サーバ - SQL リファレンス』** このマニュアルは、SQL Anywhere で使用する SQL 言語の完全なリファレンスです。また、SQL Anywhere のシステム・ビューとシステム・プロシージャについても説明しています。
- **『SQL Anywhere サーバ - プログラミング』** このマニュアルでは、C、C++、Java プログラミング言語、Visual Studio .NET を使用してデータベース・アプリケーションを構築、配備する方法について説明します。Visual Basic や PowerBuilder などのツールのユーザは、それらのツールのプログラミング・インタフェースを使用できます。
- **『SQL Anywhere 10 - エラー・メッセージ』** このマニュアルでは、SQL Anywhere エラー・メッセージの完全なリストを、その診断情報とともに説明します。
- **『Mobile Link - クイック・スタート』** このマニュアルでは、セッションベースのリレーショナル・データベース同期システムである Mobile Link について説明します。Mobile Link テクノロジは、双方向レプリケーションを可能にし、モバイル・コンピューティング環境に非常に適しています。
- **『Mobile Link - サーバ管理』** このマニュアルでは、Mobile Link アプリケーションを設定して管理する方法について説明します。

- **『Mobile Link - クライアント管理』** 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。
- **『Mobile Link - サーバ起動同期』** このマニュアルでは、Mobile Link のサーバによって開始される同期について説明します。サーバによって開始される同期とは、統合データベースから同期またはその他のリモート・アクションの開始を可能にする Mobile Link の機能です。
- **『QAnywhere』** このマニュアルでは QAnywhere について説明します。QAnywhere は、従来のデスクトップ・クライアントやラップトップ・クライアント用のメッセージング・プラットフォームであるほか、モバイル・クライアントや無線クライアント用のメッセージング・プラットフォームでもあります。
- **『SQL Remote』** このマニュアルでは、モバイル・コンピューティング用の SQL Remote データ・レプリケーション・システムについて説明します。このシステムによって、SQL Anywhere の統合データベースと複数の SQL Anywhere リモート・データベースの間で、電子メールやファイル転送などの間接的リンクを使用したデータ共有が可能になります。
- **『SQL Anywhere 10 - コンテキスト別ヘルプ』** このマニュアルには、[接続] ダイアログ、クエリ・エディタ、Mobile Link モニタ、SQL Anywhere コンソール・ユーティリティ、インデックス・コンサルタント、Interactive SQL のコンテキスト別のヘルプが収録されています。
- **『Ultra Light - データベース管理とリファレンス』** このマニュアルでは、小型デバイス用 Ultra Light データベース・システムの概要を説明します。
- **『Ultra Light - AppForge プログラミング』** このマニュアルでは、Ultra Light for AppForge について説明します。Ultra Light for AppForge を使用すると、Palm OS、Symbian OS、または Windows CE を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - .NET プログラミング』** このマニュアルでは、Ultra Light.NET について説明します。Ultra Light.NET を使用すると、PC、ハンドヘルド、モバイル、埋め込みデバイスのデータベース・アプリケーションを開発し、これらのデバイスに配備できます。
- **『Ultra Light - M-Business Anywhere プログラミング』** このマニュアルは、Ultra Light for M-Business Anywhere について説明します。Ultra Light for M-Business Anywhere を使用すると、Palm OS、Windows CE、または Windows XP を搭載しているハンドヘルド、モバイル、または埋め込みデバイスに対して Web ベースのデータベース・アプリケーションを開発、配備できます。
- **『Ultra Light - C/C++ プログラミング』** このマニュアルでは、Ultra Light C および Ultra Light C++ のプログラミング・インタフェースについて説明します。Ultra Light を使用すると、ハンドヘルド、モバイル、埋め込みデバイスに対してデータベース・アプリケーションを開発、配備できます。

マニュアルの形式

SQL Anywhere のマニュアルは、次の形式で提供されています。

- **オンライン・マニュアル** オンライン・マニュアルには、SQL Anywhere の完全なマニュアルがあり、SQL Anywhere ツールに関する印刷マニュアルとコンテキスト別のヘルプの両方が含

まれています。オンライン・マニュアルは、製品のメンテナンス・リリースごとに更新されます。これは、最新の情報を含む最も完全なマニュアルです。

Windows オペレーティング・システムでオンライン・マニュアルにアクセスするには、[スタート] − [プログラム] − [SQL Anywhere 10] − [オンライン・マニュアル] を選択します。オンライン・マニュアルをナビゲートするには、左ウィンドウ枠で HTML ヘルプの目次、索引、検索機能を使用し、右ウィンドウ枠でリンク情報とメニューを使用します。

UNIX オペレーティング・システムでオンライン・マニュアルにアクセスするには、SQL Anywhere のインストール・ディレクトリまたはインストール CD に保存されている HTML マニュアルを参照してください。

- **PDF ファイル** SQL Anywhere の完全なマニュアル・セットは、Adobe Reader で表示できる Adobe Portable Document Format (pdf) 形式のファイルとして提供されています。

  Windows では、PDF 形式のマニュアルはオンライン・マニュアルの各ページ上部にある PDF のリンクから、または Windows の [スタート] メニュー ([スタート] − [プログラム] − [SQL Anywhere 10] − [オンライン・マニュアル - PDF フォーマット]) からアクセスできます。

  UNIX では、PDF 形式のマニュアルはインストール CD にあります。

表記の規則

この項では、このマニュアルで使用されている書体およびグラフィック表現の規則について説明します。

SQL 構文の表記規則

SQL 構文の表記には、次の規則が適用されます。

- **キーワード**　SQL キーワードはすべて次の例に示す ALTER TABLE のように大文字で表記します。

  ALTER TABLE [*owner.*]*table-name*

- **プレースホルダ**　適切な識別子または式で置き換えられる項目は、次の例に示す *owner* や *table-name* のように表記します。

  ALTER TABLE [*owner.*]*table-name*

- **繰り返し項目**　繰り返し項目のリストは、次の例に示す *column-constraint* のように、リストの要素の後ろに省略記号 (ピリオド 3 つ …) を付けて表します。

  ADD *column-definition* [*column-constraint*, …]

  複数の要素を指定できます。複数の要素を指定する場合は、各要素間をカンマで区切る必要があります。

- **オプション部分**　文のオプション部分は角カッコで囲みます。

  RELEASE SAVEPOINT [*savepoint-name*]

  この例では、角カッコで囲まれた *savepoint-name* がオプション部分です。角カッコは入力しないでください。

- **オプション**　項目リストから 1 つだけ選択する場合や、何も選択しなくてもよい場合は、項目間を縦線で区切り、リスト全体を角カッコで囲みます。

  [**ASC** | **DESC**]

  この例では、ASC と DESC のどちらか 1 つを選択しても、選択しなくてもかまいません。角カッコは入力しないでください。

- **選択肢**　オプションの中の 1 つを必ず選択しなければならない場合は、選択肢を中カッコで囲み、縦棒で区切ります。

  [**QUOTES** { **ON** | **OFF** }]

  QUOTES オプションを使用する場合は、ON または OFF のどちらかを選択する必要があります。角カッコと中カッコは入力しないでください。

オペレーティング・システムの表記規則

- **Windows**　デスクトップおよびラップトップ・コンピュータ用の Microsoft Windows オペレーティング・システムのファミリのことです。Windows ファミリには Windows Vista や Windows XP も含まれます。
- **Windows CE**　Microsoft Windows CE モジュラ・オペレーティング・システムに基づいて構築されたプラットフォームです。Windows Mobile や Windows Embedded CE などのプラットフォームが含まれます。

  Windows Mobile は Windows CE 上に構築されています。これにより、Windows のユーザ・インタフェースや、Word や Excel といったアプリケーションの小規模バージョンなどの追加機能が実現されています。Windows Mobile は、モバイル・デバイスで最も広く使用されています。

  SQL Anywhere の制限事項や相違点は、基盤となっているオペレーティング・システム (Windows CE) に由来しており、使用しているプラットフォーム (Windows Mobile など) に依存していることはほとんどありません。
- **UNIX**　特に記述がないかぎり、UNIX は Linux プラットフォームと UNIX プラットフォームの両方のことです。

ファイルの命名規則

マニュアルでは、パス名やファイル名などのオペレーティング・システムに依存するタスクと機能を表すときは、通常 Windows の表記規則が使用されます。ほとんどの場合、他のオペレーティング・システムで使用される構文に簡単に変換できます。

- **ディレクトリ名とパス名**　マニュアルでは、ドライブを示すコロンや、ディレクトリの区切り文字として使用する円記号など、Windows の表記規則を使用して、ディレクトリ・パスのリストを示します。次に例を示します。

  ```
  MobiLink¥redirector
  ```

  UNIX、Linux、Mac OS X では、代わりにスラッシュを使用してください。次に例を示します。

  ```
  MobiLink/redirector
  ```

  SQL Anywhere がマルチプラットフォーム環境で使用されている場合、プラットフォーム間でのパス名の違いに注意する必要があります。
- **実行ファイル**　マニュアルでは、実行ファイルの名前は、Windows の表記規則が使用され、拡張子 *.exe* が付きます。UNIX、Linux、Mac OS X では、実行ファイルの名前には拡張子は付きません。NetWare では、実行ファイルの名前には、拡張子 *.nlm* が付きます。

  たとえば、Windows では、ネットワーク・データベース・サーバは *dbsrv10.exe* です。UNIX、Linux、Mac OS X では、*dbsrv10* になります。NetWare では、*dbsrv10.nlm* になります。
- **install-dir**　インストール・プロセスでは、SQL Anywhere をインストールするロケーションを選択できます。マニュアルでは、このロケーションは *install-dir* という表記で示されます。

インストールが完了すると、環境変数 SQLANY10 によって SQL Anywhere コンポーネントがあるインストール・ディレクトリのロケーション (*install-dir*) が指定されます。SQLANYSH10 は、SQL Anywhere が他の Sybase アプリケーションと共有しているコンポーネントがあるディレクトリのロケーションを指定します。

オペレーティング・システム別の *install-dir* のデフォルト・ロケーションの詳細については、「SQLANY10 環境変数」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **samples-dir** インストール・プロセスでは、SQL Anywhere に含まれるサンプルをインストールするロケーションを選択できます。マニュアルでは、このロケーションは *samples-dir* という表記で示されます。

  インストールが完了すると、環境変数 SQLANYSAMP10 によってサンプルがあるディレクトリのロケーション (*samples-dir*) が指定されます。Windows の [スタート] メニューから、[プログラム] – [SQL Anywhere 10] – [サンプル・アプリケーションおよびプロジェクト] を選択すると、このディレクトリで [Windows エクスプローラ] ウィンドウが表示されます。

  オペレーティング・システム別の *samples-dir* のデフォルト・ロケーションの詳細については、「サンプル・ディレクトリ」『SQL Anywhere サーバ - データベース管理』を参照してください。

- **環境変数** マニュアルでは、環境変数設定が引用されます。Windows では、環境変数を参照するのに、構文 *%envvar%* が使用されます。UNIX、Linux、Mac OS X では、環境変数を参照するのに、構文 *$envvar* または *${envvar}* が使用されます。

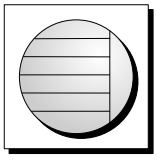
  UNIX、Linux、Mac OS X 環境変数は、*.cshrc* や *.tcshrc* などのシェルとログイン・スタートアップ・ファイルに格納されます。

グラフィック・アイコン

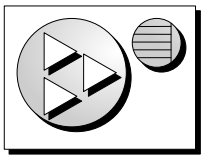
このマニュアルでは、次のアイコンを使用します。

- クライアント・アプリケーション

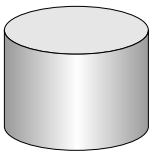
- SQL Anywhere などのデータベース・サーバ



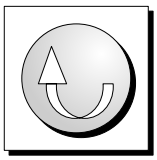
- Ultra Light アプリケーション



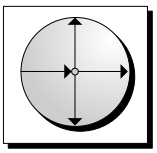
- データベース。高度な図では、データベースとデータベースを管理するデータ・サーバの両方をこのアイコンで表します。



- レプリケーションまたは同期のミドルウェア。ソフトウェアのこれらの部分は、データベース間のデータ共有を支援します。たとえば、Mobile Link サーバ、SQL Remote Message Agent などが挙げられます。



- Sybase Replication Server



- プログラミング・インタフェース

詳細情報の検索／フィードバックの提供

詳細情報の検索

詳しい情報やリソース (コード交換など) については、iAnywhere Developer Network (http://www.ianywhere.com/developer/) を参照してください。

ご質問がある場合や支援が必要な場合は、次に示す Sybase iAnywhere ニュースグループのいずれかにメッセージをお寄せください。

ニュースグループにメッセージをお送りいただく際には、ご使用の SQL Anywhere バージョンのビルド番号を明記し、現在発生している問題について詳しくお知らせくださいますようお願いいたします。バージョン情報は、コマンド・プロンプトで **dbeng10 -v** と入力して確認できます。

ニュースグループは、ニュース・サーバ *forums.sybase.com* にあります (ニュースグループにおけるサービスは英語でのみの提供となります)。以下のニュースグループがあります。

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

ニュースグループに関するお断り
iAnywhere Solutions は、ニュースグループ上に解決策、情報、または意見を提供する義務を負うものではありません。また、システム・オペレータ以外のスタッフにこのサービスを監視させて、操作状況や可用性を保証する義務もありません。
iAnywhere のテクニカル・アドバイザとその他のスタッフは、時間のある場合にかぎりニュースグループでの支援を行います。こうした支援は基本的にボランティアで行われるため、解決策や情報を定期的に提供できるとはかぎりません。支援できるかどうかは、スタッフの仕事量に左右されます。

フィードバック

このマニュアルに関するご意見、ご提案、フィードバックをお寄せください。

マニュアルに関するご意見、ご提案は、SQL Anywhere ドキュメンテーション・チームの iasdoc@ianywhere.com 宛てに電子メールでお寄せください。このアドレスに送信された電子メールに返信はいたしませんが、お寄せいただいたご意見、ご提案は必ず読ませていただきます。

マニュアルまたはソフトウェアについてのフィードバックは、上記のニュースグループを通してお寄せいただいてもかまいません。

パート I. Mobile Link クライアントの紹介

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

第 1 章

Mobile Link クライアントの紹介

目次

SQL Anywhere クライアント

dbmlsync というコマンド・ライン・ユーティリティを実行すると、同期を開始します。このユーティリティは、リモート・データベースに接続し、通常はリモート・データベースのトランザクション・ログに含まれる情報を使用してアップロードを準備します(または、トランザクション・ログを使用しないで、スクリプト化されたアップロードを起動できます)。dbmlsync ユーティリティは、同期パブリケーションと同期サブスクリプションに保管された情報を使用して Mobile Link サーバに接続し、データを交換します。

SQL Anywhere クライアントの詳細については、「SQL Anywhere クライアント」 83 ページを参照してください。

dbmlsync コマンド・ライン・オプションの詳細については、「Mobile Link SQL Anywhere クライアント・ユーティリティ [dbmlsync]」 117 ページを参照してください。

同期のカスタマイズ

dbmlsync プログラミング・インタフェースを使用することで、独自のアプリケーションに同期を統合できます。次の項を参照してください。

- 「dbmlsync 統合コンポーネント」 316 ページ
- 「dbmlsync の DBTools インタフェース」 315 ページ

dbmlsync のカスタマイズに使用できるクライアント・イベント・フックもあります。次の項を参照してください。

- 「SQL Anywhere クライアントのイベント・フック」 217 ページ

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。次の項を参照してください。

- 「スクリプト化されたアップロード」 323 ページ

Ultra Light クライアント

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

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

TCP/IP、HTTP、HTTPS、または ActiveSync を使用している場合には、同期関数を 1 回呼び出すと、アプリケーションから同期が開始されます。HotSync 用のインタフェースは、若干異なります。同期がアプリケーションまたは HotSync から開始されると、Mobile Link サーバと Ultra Light ランタイムが同期中の動作を制御します。

参照

- Ultra Light - データベース管理とリファレンス 『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light クライアント」 349 ページ
- 「Ultra Light クライアント用 ActiveSync と HotSync の配備」 369 ページ
- 「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページ

クライアントのネットワーク・プロトコルの指定

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。

mlsrv10 コマンド・ライン・オプションの構文は次のとおりです。

mlsrv10 -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。

```
mlsrv10 -c "dsn=SQL Anywhere 10 Demo" -x tcpip
```

プロトコルは、次の形式のオプションを使用して設定できます。

(*keyword***=***value***;⋯)**

次に例を示します。

```
mlsrv10 -c "dsn=SQL Anywhere 10 Demo" -x tcpip(
  host=localhost;port=2439)
```

参照

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

項目	参照場所
Mobile Link サーバのネットワーク・オプションの設定方法	「-x オプション」 『Mobile Link - サーバ管理』
Mobile Link クライアント・アプリケーションで使用可能なすべてのネットワーク・プロトコル・オプション	「Mobile Link クライアントのネットワーク・プロトコル・オプション」 35 ページ
SQL Anywhere クライアントのオプションの設定方法	「CommunicationAddress (adr) 拡張オプション」 173 ページ 「CommunicationType (ctp) 拡張オプション」 175 ページ
Ultra Light クライアントのオプションの設定方法	「Stream Parameters 同期パラメータ」 400 ページ 「Stream Type 同期パラメータ」 399 ページ

Mobile Link のシステム・テーブル

Mobile Link サーバのシステム・テーブル

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

「Mobile Link サーバ・システム・テーブル」『Mobile Link - サーバ管理』を参照してください。

クライアントのシステム・テーブル

SQL Anywhere と Ultra Light のデータベースにはシステム・テーブルがあります。

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

SQL Anywhere システム・テーブル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。「Sybase Central のシステム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

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」『SQL Anywhere サーバ - SQL リファレンス』
- 「SYSPUBLICATION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- 「SYSSUBSCRIPTION システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- 「SYSSYNCSCRIPT システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』

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

第 2 章

Mobile Link ユーザ

目次

Mobile Link ユーザの概要

「**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。同期システムに含まれるユーザの場合は、リモート・データベースに Mobile Link ユーザ名を作成し、Mobile Link サーバを使用して Mobile Link ユーザ名を登録してください。

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。

Mobile Link ユーザ名を使用して、同期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。

「スクリプトでのリモート ID と Mobile Link ユーザ名の使用」 16 ページを参照してください。

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 の名前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 ユーザ名と共有に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。

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

Mobile Link ユーザの作成と登録

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、Sybase Central または CREATE SYNCHRONIZATION USER 文を使用します。

  「SQL Anywhere リモート・データベースでの Mobile Link ユーザの作成」 97 ページを参照してください。

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

  「User Name 同期パラメータ」 405 ページと「Password 同期パラメータ」 391 ページを参照してください。

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する。

  「Mobile Link ユーザ認証ユーティリティ [mluser]」『Mobile Link - サーバ管理』を参照してください。

- Sybase Central を使用する。
- authenticate_user イベント用または authenticate_user_hashed イベント用のスクリプトを実装する。これらのスクリプトのどちらかを起動すると、Mobile Link サーバによって、認証が正常に行われるユーザが自動的に追加されます。
- mlsrv10 で -zu+ コマンド・ライン・オプションを指定する。この場合、最初に同期するときに、統合データベースに追加されていない既存の Mobile Link ユーザが追加されます。このオプションは、開発時には便利ですが、配備されたアプリケーションへの使用はおすすめできません。

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

ユーザの最初のパスワードを設定する

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。最初のパスワードは、Sybase Central から指定するか、または mluser コマンド・ライン・ユーティリティを使用して指定できます。

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

パスワードを指定しないでユーザを作成すると、Mobile Link はそのユーザに対してユーザ認証を実行しません。つまり、そのユーザは、パスワードを入力せずに接続や同期ができます。

♦ 1 ユーザに最初の Mobile Link パスワードを設定するには、次の手順に従います (Sybase Central の Admin モードの場合)。

1. Sybase Central から、Mobile Link プラグインを使用して統合データベースに接続します。
2. 管理モードを選択します。

   [モード] – [管理] を選択します。
3. [ユーザ] フォルダを開きます。
4. [ファイル] – [新規] – [ユーザ] を選択します。

   [ユーザの作成] ウィザードが表示されます。
5. ウィザードの指示に従います。

♦ **最初の Mobile Link パスワードを設定するには、次の手順に従います (コマンド・ラインの場合)。**

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・プロンプトを開き、mluser コマンド・ライン・ユーティリティを実行します。次に例を示します。

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

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

   「Mobile Link ユーザ認証ユーティリティ [mluser]」 『Mobile Link - サーバ管理』を参照してください。

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、デフォルトでは、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。これ以降、このユーザはパスワードなしで同期できるようになります。
- **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、[パスワード] フィールドの代わりに [新しいパスワード] フィールドに情報を入力するか、両方のフィールドに入力します。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。

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

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。

エンド・ユーザに対するパスワード入力の要求

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。

♦ **エンド・ユーザに Mobile Link パスワードの入力を要求するには、次の手順に従います。**

1. 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。

     「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページを参照してください。

   - **SQL Anywhere** ユーザは、dbmlsync コマンド・ラインで有効なパスワードを入力できます。コマンド・ラインに入力しないと、dbmlsync 接続ダイアログに入力するように要求されます。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ため、ダイアログで入力する方が安全です。

     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。

     「-c オプション」 130 ページを参照してください。

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。

♦ **エンド・ユーザに Mobile Link パスワードを入力させるには、次の手順に従います。**

1. 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 接続ダイアログ (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。

  「-mp オプション」 145 ページと「-mn オプション」 144 ページを参照してください。

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

  「Password 同期パラメータ」 391 ページと「New Password 同期パラメータ」 389 ページを参照してください。

最初のパスワードは、統合サーバで、または最初の同期で設定できます。「ユーザの最初のパスワードを設定する」 11 ページと「新しいユーザからの同期」 12 ページを参照してください。

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

リモート ID

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

SQL Anywhere または Ultra Light データベースを作成すると、NULL のリモート ID を指定できます。データベースを Mobile Link と同期する場合、Mobile Link は NULL のリモート ID をチェックし、該当するリモート ID が見つかると、GUID をリモート ID として割り当てます。リモート ID を一度設定すると、手動で変更されるまで、データベースでは同じリモート ID を保持します。

Mobile Link サーバは、SQL Anywhere リモート・データベースのリモート ID とサブスクリプション、および Ultra Light リモート・データベースのリモート 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 サーバにも記録されます。

複数の同時同期で同じリモート ID が使用されると、Mobile Link サーバはエラーを発行します。

Mobile Link リモート ID の設定

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

SQL Anywhere クライアントについては、「リモート ID の設定」 86 ページを参照してください。

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

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。この方法により、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 が割り当てられてるので、リモート ID はユニークになります。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。

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';
```

スクリプトでのリモート ID と Mobile Link ユーザ名の使用

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 ユーザが 1 対 1 の関係になるように配備し、Mobile Link ユーザ名がリモート・データベースをユニークに識別する場合は、リモート ID を無視できます。この場合、Mobile Link イベント・スクリプトでは、username パラメータを参照できます。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。「スクリプト・バージョン」 『Mobile Link - サーバ管理』を参照してください。

ユーザ認証メカニズムの選択

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。

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

  「カスタム・ユーザ認証」 21 ページを参照してください。

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

- 「Mobile Link クライアント／サーバ通信の暗号化」『SQL Anywhere サーバ - データベース管理』
- Ultra Light クライアント：「Ultra Light でのセキュリティの考慮事項」『Ultra Light - データベース管理とリファレンス』
- SQL Anywhere クライアント：「安全なデータの管理」『SQL Anywhere サーバ - データベース管理』

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、Sybase 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 ユーザの作成と登録」 10 ページを参照してください。

Mobile Link クライアントは、Mobile Link サーバに接続するときに、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。この値はオプションであり、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。

「カスタム・ユーザ認証」 21 ページを参照してください。

認証処理

次に、認証中に発生するイベントの順序を説明します。

1. リモート・アプリケーションは、リモート ID、Mobile Link ユーザ名を使用し、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。
   a. authenticate_user スクリプトを SQL で作成した場合、このスクリプトは authentication_status が 4000 に事前に設定され、Mobile Link ユーザ名が指定されて、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 で作成した場合、SQL 文が返されてから、この SQL 文は 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。
   b. authenticate_user スクリプトで例外が発生したり、スクリプトを実行したときにエラーが発生した場合、同期処理は停止します。

   authenticate_user スクリプトまたは返された SQL 文は、2 ー 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。
   a. パスワードが指定されている場合、ハッシュされた値が計算されます。新しいパスワードが指定されている場合、ハッシュされた値が計算されます。
   b. authenticate_user_hashed スクリプトが、authentication_status の現在値 (authenticate_user が存在しない場合は事前に設定された authentication_status、または authenticate_user スクリプトから返された authentication_status) とハッシュされたパスワードで呼び出されます。動作は、手順 3 と同じです。最初のパラメータで返された値は、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。
   a. カスタム・スクリプト authenticate_user または authenticate_user_hashed のいずれかが呼び出されますが、指定された Mobile Link ユーザ名が 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。

b. カスタム・スクリプトが呼び出されず、指定された Mobile Link ユーザ名も 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。

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

d. カスタム・スクリプトが*呼び出されず*、指定された Mobile Link ユーザ名が 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。

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

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

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

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。

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として、既存の DBMS 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があり、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といったカスタム機能の提供もあげられます。

次の 3 つのカスタム認証ツールがあります。

- mlsrv10 -zu+ オプション
- authenticate_user スクリプト
- authenticate_parameters スクリプト

mlsrv10 -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 スクリプトと authenticate_parameters スクリプトは、いずれもデフォルトの 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。

Mobile Link は、authenticate_user イベント用に事前に定義されたスクリプトをいくつかインストールします。これらのスクリプトは、LDAP、POP3、IMAP サーバを使用した認証を簡単に行えるようにします。「外部サーバに対する認証」 22 ページを参照してください。

ユーザ ID とパスワードのカスタム認証を作成するには、authenticate_user を使用します。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。

次の項を参照してください。

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

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 の同期論理で本来使用されているものです。

samples-dir¥MobiLink¥JavaAuthentication ディレクトリに簡単な例があります。*samples-dir¥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。*samples-dir* の詳細については、「サンプル・ディレクトリ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

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

SQL ユーザ認証

一般的な authenticate_user SQL スクリプトは、パラメータ authentication_status、ml_username、user_password、user_new_password を使用するストアド・プロシージャの呼び出しです。呼び出しでのパラメータの順序は、これと一致する必要があります。たとえば、SQL Anywhere 統合データベースでは、フォーマットは次のようになります。

```
call my_authentication( ?, ?, ?, ? )
```

ここで、最初の引数は認証コードです。認証コードは integer 型で、その他のパラメータは VARCHAR(128) です。

Transact-SQL のフォーマットは次のとおりです。

```
execute ? = my_authentication( ?, ?, ? )
```

ここで、認証コードは左側のパラメータです。

「authenticate_user 接続イベント」 『Mobile Link - サーバ管理』を参照してください。

外部サーバに対する認証

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

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

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

注意
Sybase Central モデル・モードで [認証] タブを使用して外部サーバへの認証を設定することもできます。「Mobile Link のモデル」 『Mobile Link - クイック・スタート』を参照してください。

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

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

♦ 事前に定義されたスクリプトを authenticate_user で直接使用するには、次の手順に従います。

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

   - ml_add_java_connection_script ストアド・プロシージャを使用するには、次のコマンド・プロンプトを入力します。

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

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

   - Sybase Central の [接続スクリプト追加] ウィザードを使用するには、スクリプト・タイプとして authenticate_user を選択し、コード・エディタで次のコードを入力します。

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

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

   「ml_add_java_connection_script」 『Mobile Link - サーバ管理』を参照してください。

2. この認証サーバのプロパティを追加します。

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

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

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

   「外部認証識別符号プロパティ」 25 ページと「ml_add_property」 『Mobile Link - サーバ管理』を参照してください。

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

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

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

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

```
package com.mycompany.mycode;

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

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

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


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

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

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

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

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

外部認証識別符号プロパティ

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

POP3 認証識別符号

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

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

IMAP 認証識別符号

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

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

LDAP 認証識別符号

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

詳細については、JNDI のマニュアルを参照してください。

第 3 章

Mobile Link クライアント・ユーティリティ

目次

Mobile Link クライアント・ユーティリティの概要

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

- 「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」 29 ページ
- 「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」 32 ページ

次の各項も参照してください。

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

ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]

ActiveSync 用 Mobile Link プロバイダをインストールするか、Windows CE デバイス上で Ultra Light アプリケーションを登録してインストールします。

構文

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

オプション	説明
-k *path*	デスクトップ・プロバイダ *mlasdesk.dll* のロケーションを指定する。デフォルトでは、このファイルは SQL Anywhere のインストール・ディレクトリの *win32* サブディレクトリ内で検索されます。 エンド・ユーザ (通常は SQL Anywhere 全体がインストールされていないユーザ) は、Mobile Link ActiveSync プロバイダのインストール時に、-k の指定が必要になる場合があります。
-n	アプリケーションを登録するが、デバイスにはコピーしない このオプションを指定すると、Mobile Link ActiveSync プロバイダのインストールに加えてアプリケーションが登録されますが、アプリケーションはデバイスにコピーはされません。アプリケーションに複数のファイルがある場合 (静的ライブラリではなく Ultra Light ランタイム・ライブラリ DLL を使用するようにコンパイルされている場合など)、または他の方法でアプリケーションをデバイスにコピーする場合は、このオプションを指定します。
-u	ActiveSync 用 Mobile Link プロバイダをアンインストールする。 このオプションを指定すると、Mobile Link ActiveSync プロバイダ用に登録されたアプリケーションの登録がすべて解除され、Mobile Link ActiveSync プロバイダがアンインストールされます。この操作によってデスクトップ・マシンやデバイスからファイルが削除されることはありません。デバイスがデスクトップに接続されていない場合は、エラーが返されます。
-v *path*	デバイス・プロバイダ *mlasdev.dll* のロケーションを指定する。デフォルトでは、このファイルは SQL Anywhere ディレクトリの *CE* サブディレクトリにある、プラットフォーム固有のディレクトリ内で検索されます。 エンド・ユーザ (通常は SQL Anywhere 全体がインストールされていないユーザ) は、Mobile Link ActiveSync プロバイダのインストール時に、-v の指定が必要になる場合があります。

その他のパラメータ	説明
src	アプリケーションをデバイスにコピーするためのソース・ファイル名とパスを指定する。このパラメータを指定するのは、アプリケーションを登録してデバイスにコピーする場合のみです。-n オプションを使用する場合は、このパラメータを指定しないでください。

その他のパラメータ	説明
dst	デバイス上でアプリケーションに使用するコピー先ファイル名とパスを指定する。
name	アプリケーション名を指定する。これは、ActiveSync がアプリケーションを参照するときに使用する名前です。
class	アプリケーションの登録済み Windows クラス名を指定する。
args	ActiveSync によってアプリケーションが起動されるときにアプリケーションに渡すコマンド・ライン引数を指定する。

説明

このユーティリティによって、ActiveSync 用の Mobile Link プロバイダがインストールされます。プロバイダには、デスクトップ上で実行されるコンポーネント (*mlasdesk.dll*) と、Windows CE デバイスに展開されるコンポーネント (*mlasdev.dll*) の両方が含まれています。mlasinst ユーティリティは、デスクトップ・プロバイダの現在のロケーションを示すレジストリ・エントリを作成し、デバイス・プロバイダをデバイスにコピーします。

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

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

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

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

- SQL Anywhere：「ActiveSync 用 Mobile Link プロバイダのインストール」 108 ページ
- Ultra Light：「Windows CE の ActiveSync」 376 ページ

例

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

```
mlasinst
```

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

```
mlasinst -u
```

次のコマンドは、ActiveSync 用の Mobile Link プロバイダがまだインストールされていない場合はインストールし、アプリケーション *myapp.exe* を登録します。また、*c:¥My Files¥myapp.exe* ファイルをデバイス上の *¥Program Files¥myapp.exe* にコピーします。-p -x 引数は、ActiveSync によっ

て起動されるときの myapp.exe に対するコマンド・ライン・オプションです。このコマンドは、1 行に入力してください。

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

参照

- 「ActiveSync 同期の使用」 107 ページ
- 「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページ

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ダウンロードします。

構文

mlfiletransfer [*options*] [*transfer-file*]

オプション	説明
-ap *param1*, ...	Mobile Link 認証パラメータ。「認証パラメータ」 『Mobile Link - サーバ管理』を参照してください。
-dp *path*	ダウンロードしたファイルを保存するローカル・パス。デフォルトでは、ダウンロードしたファイルは Windows CE ではルート・ディレクトリに、その他のプラットフォームでは現在のディレクトリに保存されます。
-df *filename*	ダウンロードしたファイルのローカル名。サーバで使用していたファイル名とは異なるファイル名をクライアントで使用する場合はこのオプションを使用します。デフォルトでは、サーバでのファイル名が使用されます。
-f	強制的にダウンロードし、ローカル・ファイルも最新の状態にします。前の部分的なダウンロードは破棄されます。
-g	ダウンロードの進行状況を表示します。
-p *password*	Mobile Link ユーザ名のパスワード。
-r	ダウンロードの再開を有効にします。このオプションを設定すると、ユーティリティは、通信エラーまたはユーザのキャンセルによって中断した前の部分的なダウンロードを再開します。サーバのファイルが部分的なファイルよりも新しい場合、部分的なファイルは破棄されます。このオプションよりも、-f オプションが優先されます。
-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**、**https** のいずれか 1 つを指定します。このオプションは必須です。 使用できる protocol-options は、プロトコルによって異なります。各プロトコルのオプションのリストについては、「Mobile Link クライアント・ネットワーク・プロトコル・オプション」 37 ページを参照してください。

オプション	説明
transfer-file	転送するファイルのサーバでのファイル名。パスを含めないでください。ファイルのロケーションは、mlsrv10 -ftr オプション (Mobile Link サーバを起動するときに使用) によって決まります。Mobile Link は、-ftr ディレクトリの username サブディレクトリでファイルを検索します。このサブディレクトリにファイルが見つからない場合は、-ftr ディレクトリを検索します。ファイルがどちらのディレクトリでも見つからない場合は、エラーが生成されます。 「-ftr オプション」 『Mobile Link - サーバ管理』を参照してください。

説明

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 ユーザごとにサブディレクトリを作成します。

Ultra Light ユーザは、Ultra Light ランタイムで MLFileTransfer メソッドも使用できます。「Mobile Link ファイル転送の使用」 367 ページを参照してください。

参照

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

第 4 章

Mobile Link クライアントのネットワーク・プロトコル・オプション

目次

Mobile Link クライアント・ネットワーク・プロトコル・オプション

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

クライアント・ネットワーク・プロトコル・オプションを使用するユーティリティ	参照場所
dbmlsync	「CommunicationAddress (adr) 拡張オプション」 173 ページ
Mobile Link ファイル転送ユーティリティ	「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」 32 ページ
Mobile Link Listener	「Listener 構文」 『Mobile Link - サーバ起動同期』の -x
Mobile Link モニタ	「Mobile Link モニタの起動」 『Mobile Link - サーバ管理』
QAnywhere Agent	「-x オプション」 『QAnywhere』
リダイレクタ	「リダイレクタのプロパティの設定 (サーバ・グループをサポートするリダイレクタの場合)」 『Mobile Link - サーバ管理』または「リダイレクタのプロパティの設定 (サーバ・グループをサポートしないリダイレクタの場合)」 『Mobile Link - サーバ管理』の Mobile Link ディレクティブ
Ultra Light	「Stream Parameters 同期パラメータ」 400 ページ

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

プロトコル・オプション

- **TCP/IP プロトコル・オプション** tcpip プロトコルを指定する場合は、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
buffer_size=*bytes*	「buffer_size」 42 ページ
client_port=*nnnnn*[-*mmmmm*]	「client_port」 48 ページ
compression={**zlib**|**none**}	「compression」 49 ページ

TCP/IP プロトコル・オプション	詳細の参照先
host=*hostname*	「host」 53 ページ
network_leave_open={**off**|**on**}	「network_leave_open」 58 ページ
network_name=*name*	「network_name」 59 ページ
port=*portnumber*	「port」 61 ページ
timeout=*seconds*	「timeout」 65 ページ
zlib_download_window_size=*window-bits*	「zlib_download_window_size」 73 ページ
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」 74 ページ

- **TLS プロトコル** TLS プロトコル (TLS セキュリティ付きの TCP/IP) を指定すると、必要に応じて次のプロトコル・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
buffer_size=*bytes*	「buffer_size」 42 ページ
certificate_company=*company_name*	「certificate_company」 43 ページ
certificate_name=*name*	「certificate_name」 45 ページ
certificate_unit=*company_unit*	「certificate_unit」 47 ページ
client_port=*nnnnn*[*-mmmmm*]	「client_port」 48 ページ
compression={**zlib**|**none**}	「compression」 49 ページ
host=*hostname*	「host」 53 ページ
fips={**y**|**n**}	「fips」 51 ページ
network_leave_open={**off**|**on**}	「network_leave_open」 58 ページ
network_name=*name*	「network_name」 59 ページ
port=*portnumber*	「port」 61 ページ
timeout=*seconds*	「timeout」 65 ページ
tls_type={**rsa**|**ecc**}	「tls_type」 66 ページ
zlib_download_window_size=*window-bits*	「zlib_download_window_size」 73 ページ
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」 74 ページ

- **HTTP プロトコル** http プロトコルを指定する場合は、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」 42 ページ
client_port=*nnnnn*[**-***mmmmm*]	「client_port」 48 ページ
compression={**zlib**|**none**}	「compression」 49 ページ
custom_header=*header*	「custom_header」 50 ページ
http_password=*password*	「http_password」 54 ページ
http_proxy_password=*password*	「http_proxy_password」 55 ページ
http_proxy_userid=*userid*	「http_proxy_userid」 56 ページ
http_userid=*userid*	「http_userid」 57 ページ
host=*hostname*	「host」 53 ページ
network_leave_open={**off**|**on**}	「network_leave_open」 58 ページ
network_name=*name*	「network_name」 59 ページ
persistent={**off**|**on**}	「persistent」 60 ページ
port=*portnumber*	「port」 61 ページ
proxy_host=*proxy-hostname-or-ip*	「proxy_host」 62 ページ
proxy_port=*proxy-portnumber*	「proxy_port」 63 ページ
set_cookie=*cookie-name***=***cookie-value*	「set_cookie」 64 ページ
timeout=*seconds*	「timeout」 65 ページ
trusted_certificates=*filename*	「trusted_certificates」 68 ページ
url_suffix=*suffix*	「url_suffix」 70 ページ
version=*HTTP-version-number*	「version」 72 ページ
zlib_download_window_size=*window-bits*	「zlib_download_window_size」 73 ページ
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」 74 ページ

- **HTTPS プロトコル** HTTPS プロトコルでは RSA 暗号化が使用されます。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

HTTPS プロトコルを指定する場合は、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」 42 ページ
certificate_company=*company_name*	「certificate_company」 43 ページ
certificate_name=*name*	「certificate_name」 45 ページ
certificate_unit=*company_unit*	「certificate_unit」 47 ページ
client_port=*nnnnn*[**-***mmmmm*]	「client_port」 48 ページ
compression={**zlib**|**none**}	「compression」 49 ページ
custom_header=*header*	「custom_header」 50 ページ
fips={**y**|**n**}	「fips」 51 ページ
host=*hostname*	「host」 53 ページ
http_password=*password*	「http_password」 54 ページ
http_proxy_password=*password*	「http_proxy_password」 55 ページ
http_proxy_userid=*userid*	「http_proxy_userid」 56 ページ
http_userid=*userid*	「http_userid」 57 ページ
network_leave_open={**off**|**on**}	「network_leave_open」 58 ページ
network_name=*name*	「network_name」 59 ページ
persistent={**off**|**on**}	「persistent」 60 ページ
port=*portnumber*	「port」 61 ページ
proxy_host=*proxy-hostname-or-ip*	「proxy_host」 62 ページ
proxy_port=*proxy-portnumber*	「proxy_port」 63 ページ
set_cookie=*cookie-name***=***cookie-value*	「set_cookie」 64 ページ

HTTPS プロトコル・オプション	詳細の参照先
timeout=*seconds*	「timeout」 65 ページ
tls_type={**rsa**|**ecc**}	「tls_type」 66 ページ
trusted_certificates=*filename*	「trusted_certificates」 68 ページ
url_suffix=*suffix*	「url_suffix」 70 ページ
version=*HTTP-version-number*	「version」 72 ページ
zlib_download_window_size=*window-size*	「zlib_download_window_size」 73 ページ
zlib_upload_window_size=*window-bits*	「zlib_upload_window_size」 74 ページ

buffer_size

バッファの最大バイト数を指定してから、ネットワークに書き込みます。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*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

- CE、Palm、Symbian - 2 K
- その他のプラットフォーム - 16 K

説明

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の数は減りますが、必要なメモリ量は増えます。

TCPIP と TLS でも、バッファ サイズが大きくなるほどパフォーマンスは向上しますが、必要なメモリ量が増えます。ただし、パフォーマンスの差は、HTTP ほど大きくありません。

単位はバイトです。キロバイトには K を使用してください。

最大値は 64 K です。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

certificate_company

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

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

構文

certificate_company=*organization*

プロトコル

- TLS、HTTPS

デフォルト

なし

説明

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用するため、同じ認証局が他の会社用に発行した証明書も信用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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「Mobile Link クライアント／サーバ通信の暗号化」『SQL Anywhere サーバ - データベース管理』
- 「証明書フィールドの確認」『SQL Anywhere サーバ - データベース管理』
- 「-x オプション」『Mobile Link - サーバ管理』
- 「trusted_certificates」 68 ページ
- 「certificate_name」 45 ページ
- 「certificate_unit」 47 ページ

例

3 つの識別情報フィールドすべてを検査し、指定した値だけを受け入れるように SQL Anywhere クライアントに指示する例を示します。この例は 3 つのフィールドすべてを確認します。フィールドを 1 つまたは 2 つだけ確認するように選択することもできます。

たとえば、SQL Anywhere クライアントが存在する場合、証明書の検証をサブスクリプションで次のように設定できます。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
FOR 'user01'
TO test_pub
ADDRESS 'port=3333;
  trusted_certificates=certicom.crt;
  certificate_company=Sybase, Inc.;
  certificate_unit=iAnywhere;certificate_name=sample'
```

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、次のように証明書の検証を設定できます。ここでは、データベースの作成時に、信用された証明書がデータベースにインストールされていることを前提としています。

```
ul_synch_info info;
info.stream = "tls";
info.stream_parms = UL_TEXT("port=9999;")
   UL_TEXT ( "certificate_company=Sybase, Inc.;" )
   UL_TEXT ( "certificate_unit=iAnywhere;" )
   UL_TEXT ( "certificate_name=sample;" );
. . .
ULSynchronize( &info );
```

certificate_name

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

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

構文

certificate_name=*common-name*

プロトコル

- TLS、HTTPS

デフォルト

なし

説明

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「Mobile Link クライアント／サーバ通信の暗号化」 『SQL Anywhere サーバ - データベース管理』
- 「証明書フィールドの確認」 『SQL Anywhere サーバ - データベース管理』
- 「-x オプション」 『Mobile Link - サーバ管理』
- 「trusted_certificates」 68 ページ
- 「certificate_company」 43 ページ
- 「certificate_unit」 47 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバ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。

```
mlsrv10
  -c "dsn=SQL Anywhere 10 Demo;uid=DBA;pwd=sql"
  -x https(
    port=9999;
    certificate=c:¥sa10¥win32¥rsaserver.crt;
    certificate_password=test)
```


SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "dsn=mydb;uid=DBA;pwd=sql"
  -e "ctp=https;
      adr='port=9999;
        trusted_certificates=c:¥sa10¥win32¥rsaroot.crt;
        certificate_name=RSA Server'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
info.stream = "https";
info.stream_parms = TEXT(
  "port=9999;
   trusted_certificates=¥sa10¥win32¥rsaroot.crt;
   certificate_name=RSA Server");
```

certificate_unit

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

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

構文

certificate_unit=*organization-unit*

プロトコル

- TLS、HTTPS

デフォルト

なし

説明

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「Mobile Link クライアント／サーバ通信の暗号化」 『SQL Anywhere サーバ - データベース管理』
- 「証明書フィールドの確認」 『SQL Anywhere サーバ - データベース管理』
- 「-x オプション」 『Mobile Link - サーバ管理』
- 「trusted_certificates」 68 ページ
- 「certificate_company」 43 ページ
- 「certificate_name」 45 ページ

例

セキュリティの例については、「certificate_name」 45 ページと「trusted_certificates」 68 ページを参照してください。

client_port

通信に使用するクライアント・ポートの範囲を指定します。

構文

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

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

compression

Mobile Link サーバと Mobile Link クライアント間の同期ストリームの圧縮のオンとオフを切り替えます。

構文

compression= { zlib | none }

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- Palm OS または Symbian ではサポートされません。
- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

Ultra Light では、compression はデフォルトでオフになっています。

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

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

説明

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。

Ultra Light で zlib compression を使用するには、アプリケーションで ULEnableZlibSyncCompression(sqlca) を呼び出し、*mlczlib10.dll* を配備する必要があります。

参照

- 「zlib_download_window_size」 73 ページ
- 「zlib_upload_window_size」 74 ページ

例

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

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


custom_header

カスタム HTTP ヘッダを指定します。

構文

custom_header=*header*

HTTP ヘッダの形式は、*header_name*: *header_value* です。

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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、custom_header を複数回使用してください。

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

例

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

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


fips

通信の暗号化に、FIPA によって認可された暗号化の実装を使用します。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

構文

fips={ y | n }

プロトコル

HTTPS、TLS

デフォルト

N

説明

FIPS は、RSA 暗号化でのみでサポートされています。

非 FIPS クライアントは、FIPS サーバに接続することはできません。逆についても同様です。

参照

- 「tls_type」 66 ページ

例

TCP/IP プロトコルの FIPS 承認の 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。

```
mlsrv10
  -c "dsn=SQL Anywhere 10 Demo;uid=DBA;pwd=sql"
  -x tls(
    port=9999;
    tls_type=rsa;
    fips=y;
    certificate=c:¥sa10¥win32¥rsaserver.crt;
    certificate_password=test )
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e
   "CommunicationType=tls;
    CommunicationAddress=
       'tls_type=rsa;
        fips=y;
        trusted_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 = TEXT(
  "tls_type=rsa;
   fips=y;
   trusted_certificates=¥rsaroot.crt;
   certificate_name=RSA Server");
```

host

Mobile Link サーバを実行中のマシン、または、Web サーバを介して同期する場合は Web サーバを実行中のマシンのホスト名または IP 番号を指定します。

構文

host=*hostname-or-ip*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

- Windows CE - デフォルト値は、デバイスに ActiveSync パートナーシップが設定されているデスクトップ・マシンの 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。

説明

Windows CE 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 CE デバイスに ActiveSync パートナーシップが設定されているデスクトップ・マシン上の Mobile Link サーバに Windows CE デバイスを接続できます。

Palm Computing Platform の場合は、localhost のデフォルト値がデバイスを指します。デスクトップ・マシンに接続するには、明示的なホスト名または IP アドレスを指定してください。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

http_password

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

構文

http_password=*password*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「http_userid」 57 ページ
- 「http_proxy_password」 55 ページ
- 「http_proxy_userid」 56 ページ

例

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

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


http_proxy_password

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

構文

http_proxy_password=*password*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「http_password」 54 ページ
- 「http_userid」 57 ページ
- 「http_proxy_userid」 56 ページ

例

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

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


http_proxy_userid

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

構文

http_proxy_userid=*userid*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「http_password」 54 ページ
- 「http_userid」 57 ページ
- 「http_proxy_password」 55 ページ

例

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

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


http_userid

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

構文

http_userid=*userid*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「http_password」 54 ページ
- 「http_proxy_password」 55 ページ
- 「http_proxy_userid」 56 ページ

例

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

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


network_leave_open

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

構文

network_leave_open={ off | on }

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

Palm 以外のデバイスでは、デフォルトは **off** です。

Palm では、デフォルトは **on** です。

説明

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

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

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「network_name」 59 ページ

network_name

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

構文

network_name=*name*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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 CE 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・データベースを無人で同期できます。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 CE では、*name* は、[設定] – [接続] – [接続] のドロップダウン・リスト内のネットワーク・プロファイルである必要があります。インターネット・ネットワークまたは勤務先ネットワークのデフォルト設定を使用するには、名前をそれぞれキーワード **default_internet** または **default_work** に設定します。

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

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「同期のスケジュール」 111 ページ
- 「network_leave_open」 58 ページ

persistent

同期のすべての HTTP 要求に単一の TCP/IP 接続を使用します。

構文

persistent={ off | on }

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

Off

説明

on は、クライアントが同期のすべての HTTP 要求に対して同じ TCP/IP 接続を使用することを意味しています。設定を off にすると、通常、中間エージェントとの互換性が高まります。

Palm デバイスを除き、Mobile Link に直接接続している場合は、persistent を on に設定してください。プロキシやリダイレクタなどの中間エージェントを通じて接続している場合は、永続的な接続によって問題が発生することがあります。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

port

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*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

TCP/IP のデフォルトは **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。

HTTP のデフォルト値は **443** です。

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

proxy_host

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*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

HTTP プロキシを通す場合だけ使用します。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

proxy_port

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*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

HTTP プロキシを通す場合だけ使用します。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

set_cookie

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

構文

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

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

なし

説明

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。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

例

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

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

timeout

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

構文

timeout=*seconds*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

デフォルト

240 秒

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

説明

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。

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 dbmlsync が、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。

値 0 はタイムアウトがないことを意味します。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

tls_type

同期に使用する暗号を解く鍵を指定します。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

構文

tls_type=*cipher*

プロトコル

- TLS、HTTPS

デフォルト

RSA

説明

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

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

参照

- 「トランスポート・レイヤ・セキュリティを使用する Mobile Link クライアントの設定」 『SQL Anywhere サーバ - データベース管理』
- 「fips」 51 ページ
- 「Mobile Link クライアント／サーバ通信の暗号化」 『SQL Anywhere サーバ - データベース管理』
- 「-x オプション」 『Mobile Link - サーバ管理』
- 「certificate_company」 43 ページ
- 「certificate_name」 45 ページ
- 「certificate_unit」 47 ページ
- 「trusted_certificates」 68 ページ

例

TCP/IP プロトコルの RSA 暗号化を設定する例を示します。これは、サーバ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。

```
mlsrv10
  -c "dsn=SQL Anywhere 10 Demo;uid=DBA;pwd=sql"
  -x tls(
```

```
port=9999;
tls_type=rsa;
certificate=c:¥sa10¥win32¥rsaserver.crt;
certificate_password=test )
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync -e
  "CommunicationType=tls;
  CommunicationAddress=
    'tls_type=rsa;
     trusted_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 = TEXT(
 "tls_type=rsa;
  trusted_certificates=¥rsaroot.crt;
  certificate_name=RSA Server");
```


trusted_certificates

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

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

構文

trusted_certificates=*filename*

構文 2 (Palm OS)

trusted_certificates=vfs:[*volume-label*:| *volume-ordinal*:]*filename*

プロトコル

- TLS、HTTPS

デフォルト

なし

説明

Certicom 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、証明書をクライアントに送信します。送信する証明書はエンティティの署名付き証明書だけでなく、自己署名ルート証明書にまで及びます。

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用できることを確認します。この機能を使用すると、信用するルート証明書を指定できます。

Ultra Light クライアントでは、データベースを作成するときに、信用されたルートを ulinit、ulcreate、ulload に指定できます。trusted_certificates パラメータを指定すると、ファイル内にある信用された証明書が、データベース内に保存されている信用された証明書に置き換わります。

32 ビットの Windows と Windows CE では、信用された証明書が指定されないと、クライアントは、オペレーティング・システムの信用された証明書ストアから証明書をロードします。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。

信用された証明書は、Palm OS ファイル・システムでサポートされています (レコードベースのデータ・ストアではサポートされていません)。Palm OS では、*volume_label* には、**INTERNAL** (組み込みのドライブ)、**CARD** (拡張カード)、またはボリュームのラベル名を指定できます。また、*volume-ordinal* を使用して、ボリュームを識別することもできます (デフォルトは 0 で、プラットフォームで列挙されている最初のボリュームです)。Palm プラットフォームのファイル名とパスの命名規則に従って、*filename* には、ファイルへのフル・パスを指定してください。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「Ultra Light 接続パラメータでのファイル・パスの指定」 『Ultra Light - データベース管理とリファレンス』
- 「Mobile Link クライアント／サーバ通信の暗号化」 『SQL Anywhere サーバ - データベース管理』
- 「-x オプション」 『Mobile Link - サーバ管理』
- 「tls_type」 66 ページ
- 「certificate_company」 43 ページ
- 「certificate_name」 45 ページ
- 「certificate_unit」 47 ページ

例

HTTPS プロトコルの RSA 暗号化を設定する例を示します。これは、サーバ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。

```
mlsrv10
  -c "dsn=SQL Anywhere 10 Demo;uid=DBA;pwd=sql"
  -x https(
    port=9999;
    certificate=c:¥sa10¥win32¥rsaserver.crt;
    certificate_password=test)
```

SQL Anywhere クライアントでは、実装は次のようになります。

```
dbmlsync
  -c "dsn=mydb;uid=DBA;pwd=sql"
  -e "ctp=https;
      adr='port=9999;
        trusted_certificates=c:¥sa10¥win32¥rsaroot.crt;
        certificate_name=RSA Server'"
```

Ultra Light クライアントでは、実装は次のようになります。

```
    info.stream = "https";
    info.stream_parms = TEXT(
      "port=9999;
       trusted_certificates=¥rsaroot.crt;
       certificate_name=RSA Server");
    info.security_stream = NULL;
    info.security_parms = NULL;
```

Palm OS が動作している Ultra Light クライアントの場合、stream と stream_parms の設定は次のようになります。

```
info.security_stream = "tls";
info.security_parms = "tls_type=rsa;trusted_certificates=vfs:/rsaroot.crt;port=9376";
```


url_suffix

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

構文

url_suffix=*suffix*

suffix の構文は、使用するリダイレクタのタイプによって異なります。

リダイレクタ	***suffix* の構文**
ISAPI	*exe_dir*/iaredirect.dll/ml/[*server-group*] *exe-dir* は *iaredirect.dll* のロケーションです。
NSAPI	*mlredirect*/ml/[*server-group*] *mlredirect* は *obj.conf* ファイル内でマップされている名前です。
Apache	*httpd.conf* ファイルでリダイレクタの <location> タグに指定した内容
Servlet	iaredirect/ml/
M-Business Anywhere	*sync.conf* ファイルでリダイレクタの <location> タグに指定した内容

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できませんが、クライアントでリダイレクタ用に設定されます。

デフォルト

デフォルトは Mobile Link です。

説明

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。

サーバ・グループは、一部のリダイレクタでのみサポートされます。詳細については、「Mobile Link リダイレクタがサポートする Web サーバ」を参照してください。

リダイレクタを使用する場合にこのオプションを設定する方法については、「Mobile Link クライアントとサーバのリダイレクタ設定」『Mobile Link - サーバ管理』を参照してください。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「Mobile Link クライアントとサーバのリダイレクタ設定」 『Mobile Link - サーバ管理』
- 「Mobile Link サーバ・グループ」 『Mobile Link - サーバ管理』

例

次の SQL 文は、HTTP との同期に使用される、sales5322 という同期ユーザを作成します。企業のファイアウォールの外側で Mobile Link サーバを実行し、リダイレクタ (NSAPI Web サーバのリバース・プロキシ) を使用して同期要求をリダイレクトすることが前提です。Mobile Link ユーザは URL *https://www.mycompany.com:80/mlredirect/ml/* に同期します。

```
CREATE SYNCHRONIZATION USER sales5322
TYPE https
ADDRESS 'host=www.mycompany.com;port=80;url_suffix=mlredirect/ml/'
```


version

同期に使用する HTTP のバージョンを指定します。

構文

version=*HTTP-version-number*

プロトコル

- HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。

デフォルト

デフォルト値は **1.1** です。

説明

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** です。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

zlib_download_window_size

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

構文

zlib_download_window_size=*window-bits*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。
- Palm OS または Symbian ではサポートされません。

デフォルト

Windows CE では 12、それ以外では 15 です。

説明

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

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

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

Ultra Light で zlib compression をサポートするには、アプリケーションで ULEnableZlibSyncCompression(sqlca) を呼び出し、*mlczlib10.dll* を配備する必要があります。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「compression」 49 ページ
- 「zlib_upload_window_size」 74 ページ

例

次のオプションでは、アップロードの圧縮のみを設定できます。

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


zlib_upload_window_size

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

構文

zlib_upload_window_size=*window-bits*

プロトコル

- TCPIP、TLS、HTTP、HTTPS

サポートに関する注意

- リダイレクタの ML ディレクティブでは使用できません。
- Palm OS または Symbian ではサポートされません。

デフォルト

Windows CE では 12、それ以外では 15 です。

説明

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

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

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

Ultra Light で zlib compression をサポートするには、アプリケーションで ULEnableZlibSyncCompression(sqlca) を呼び出し、*mlczlib10.dll* を配備する必要があります。

dbmlsync を使用してネットワーク・プロトコル・オプションを設定する方法については、「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

Ultra Light を使用してネットワーク・プロトコル・オプションを設定する方法については、「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページを参照してください。

参照

- 「compression」 49 ページ
- 「zlib_download_window_size」 73 ページ

例

次のオプションでは、ダウンロードの圧縮のみを設定できます。

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

第 5 章

リモート・クライアントでのスキーマの変更

目次

Mobile Link クライアントでのスキーマの変更の概要

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。

警告
スキーマを変更する直前に、正常な同期を実行します。スキーマをアップグレードするときに、開いているトランザクションがないことを確認してください。

SQL Anywhere リモート・データベースのスキーマのアップグレード

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

リモート・データベースに他の接続がないことが確実である場合は、ALTER PUBLICATION 文を手動で使用して、新規または変更したテーブルをパブリケーションに追加できます。それ以外の場合は、sp_hook_dbmlsync_schema_upgrade フックを使用して、スキーマをアップグレードしてください。
「sp_hook_dbmlsync_schema_upgrade」 270 ページを参照してください。

♦ **SQL Anywhere リモート・データベースにテーブルを追加するには、次の手順に従います。**

1. 関連するテーブル・スクリプトを統合データベースに追加します。

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

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

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

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

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。
   「sp_hook_dbmlsync_schema_upgrade」 270 ページを参照してください。

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

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

リモート・データベースのテーブル定義の変更

既存テーブルのカラムの数か型を変更する場合は、注意してください。Mobile Link クライアントが新しいスキーマで同期する場合、upload_update や download_cursor などのスクリプトが必要です。これらのスクリプトには、リモート・テーブルの全カラム用のパラメータがあります。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。

♦ 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を変更するには、次の手順に従います。

1. 統合データベースで、新しいスクリプト・バージョンを作成します。

   詳細については、「スクリプト・バージョン」 『Mobile Link - サーバ管理』を参照してください。

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
   ```

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

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。「sp_hook_dbmlsync_schema_upgrade」 270 ページを参照してください。

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。

   詳細については、「ALTER TABLE 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

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。

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

   この文は、sp_hook_dbmlsync_schema_upgrade フックの内部で使用できます。「sp_hook_dbmlsync_schema_upgrade」 270 ページを参照してください。

7. 新しいスクリプト・バージョンを使用して同期します。

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

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

- 新しいデータベースで新しいアプリケーションを配備する場合は、Mobile Link サーバとの同期を行って 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できるようにして、単一の統合データベースとこれらを同期する必要があります。2 種類以上の同期スクリプトを作成し、それらを統合データベースに格納して、Mobile Link サーバ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 文のリファレンス」 『Ultra Light - データベース管理とリファレンス』を参照してください。

パート II. SQL Anywhere クライアント

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

第 6 章

SQL Anywhere クライアント

目次

リモート・データベースの作成

SQL Anywhere データベースは、Mobile Link システムでリモート・データベースとして使用できます。必要な作業は、パブリケーションを作成し、Mobile Link ユーザを作成して統合データベースに登録し、Mobile Link ユーザをパブリケーションにサブスクライブすることだけです。

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

♦ SQL Anywhere データベースをリモート・データベースとして使用するには、次の手順に従います。

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。

   「データのパブリッシュ」 89 ページを参照してください。
3. リモート・データベースに Mobile Link ユーザを作成します。

   「SQL Anywhere リモート・データベースでの Mobile Link ユーザの作成」 97 ページを参照してください。
4. ユーザを統合データベースに登録します。

   「統合データベースへの Mobile Link ユーザ名の追加」 11 ページを参照してください。
5. 1 つまたは複数のパブリケーションに対する Mobile Link ユーザのサブスクリプションを作成します。

   「同期サブスクリプションの作成」 100 ページを参照してください。

リモート・データベースの配備

SQL Anywhere リモート・データベースを配備するには、データベースを作成し、適切なパブリケーションとサブスクリプ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。

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。この方法により、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 が割り当てられてるので、リモート ID はユニークになります。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。

「リモート ID の設定」 86 ページを参照してください。

♦ **プロトタイプをカスタマイズして Mobile Link リモート・データベースを配備するには、次の手順に従います。**

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

2. リモート・データベースごとに、次の操作を実行します。

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

     トランザクション・ログがリモート・データベースと同じディレクトリに保持されている場合、ログ・ファイル名を変更する必要はありません。

   - 個々の情報をデータベースに追加する SQL スクリプトを実行します。

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

[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。「モデルの配備」 『Mobile Link - クイック・スタート』を参照してください。

参照

- 「SQL Anywhere Mobile Link クライアントの配備」 『Mobile Link - サーバ管理』
- 「最初の同期は常に行われる」 88 ページ

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 *samples-dir¥MobiLink¥Contact¥customize.sql* にあります。*samples-dir* の詳細については、「サンプル・ディレクトリ」 『SQL Anywhere サーバ - データベース管理』を参照してください。

```
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'
        OPTION MEM=''
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・ラインは、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]
```

リモート ID の設定

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

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

データベース・オプションの変更の詳細については、次の項を参照してください。

- 「SET OPTION 文」 『SQL Anywhere サーバ - SQL リファレンス』
- 「データベース・オプションの設定」 『SQL Anywhere サーバ - SQL の使用法』
- 「SYSOPTION システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』

警告
リモート ID は最初の同期を実行する前に変更すると安全です。リモート ID を後で変更する場合は、変更する直前に同期処理の実行を完了しておくようにしてください。同期処理を行わないでリモート ID を変更すると、一部のデータが失われ、データベースの整合性が保たれなくなる可能性があります。

参照

- 「リモート ID」 15 ページ

例

次の SQL 文では、リモート ID を HR001 に設定します。

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

リモート・データベースのアップグレード

新しい SQL Anywhere リモート・データベースをインストールして古いバージョンを上書きすると、統合データベース内の同期進行状況情報が正しくなくなります。この問題を解決するには、このユーザの ml_user テーブルの progress カラムを 0 (ゼロ) に設定します。

Mobile Link システム・テーブル ml_user の詳細については、「ml_user」 『Mobile Link - サーバ管理』を参照してください。

アップグレードの詳細については、「SQL Anywhere Mobile Link クライアントのアップグレード」 『SQL Anywhere 10 - 変更点とアップグレード』を参照してください。

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。ほとんどの場合、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。

詳細については、「-r オプション」 157 ページを参照してください。

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 つのパブリケーション (-n pub1,pub2) に関して dbmlsync -n オプションを指定する場合で、pub1 は以前に同期され、pub2 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。

詳細については、次の項を参照してください。

- 「ISYSSYNC システム・テーブル」『SQL Anywhere サーバ - SQL リファレンス』
- 「ml_subscription」『Mobile Link - サーバ管理』
- 「トランザクション・ログ・ファイル」 104 ページ

データのパブリッシュ

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、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。

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

Sybase Central では、[パブリケーション] フォルダ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。

パブリケーションについての注意

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

参照

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

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、一組のアーティクルで構成されます。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。これらのテーブルをあらかじめ用意してください。

♦ 1 つ以上のテーブル全体をパブリッシュするには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインを使用して、DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. [ファイル] メニューから、[新規] – [パブリケーション] を選択します。

   [パブリケーション作成] ウィザードが表示されます。
4. 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. [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが、右側の [選択したテーブル] リストに表示されます。
6. オプションで、さらにテーブルを追加することもできます。テーブルの順序は重要ではありません。
7. [完了] をクリックします。

♦ 1 つまたは複数のテーブル全体をパブリッシュするには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. CREATE PUBLICATION 文を実行して、新しく作成するパブリケーション名とパブリッシュするテーブルを指定します。

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

例

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

```
CREATE PUBLICATION pub_customer (
   TABLE customer
)
```

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

```
CREATE PUBLICATION sales (
   TABLE customer,
   TABLE sales_order,
   TABLE sales_order_items,
   TABLE product
)
```

テーブル内の一部のカラムだけをパブリッシュする

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

注意

- 異なるカラムのサブセットを持つ同じテーブルが含まれたパブリケーションを 2 つ作成すると、両方のパブリケーションをサブスクライブするユーザは同期することができません。
- 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。

♦ テーブル内の一部のカラムだけをパブリッシュするには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインを使用して、DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. [ファイル] メニューから、[新規] – [パブリケーション] を選択します。

   [パブリケーション作成] ウィザードが表示されます。
4. 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. [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが右側の [選択したテーブル] のリストに追加されます。
6. [カラム] タブで、テーブルのアイコンをダブルクリックし、[使用できるカラム] のリストを展開します。パブリッシュするカラムを 1 つずつ選択し、[追加] をクリックします。右側に選択したカラムが表示されます。
7. [終了] をクリックします。

♦ テーブル内の一部のカラムだけをパブリッシュするには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. CREATE PUBLICATION 文を実行して、パブリケーション名とテーブル名を指定します。テーブル名の後ろにあるカッコの中に、パブリッシュするカラムをリストします。

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

例

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


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

テーブル内の一部のローだけをパブリッシュする

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

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

- サブクエリ
- 変数
- 非決定的関数

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

♦ WHERE 句を使用してパブリケーションを作成するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインを使用して、DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. [ファイル] メニューから、[新規] – [パブリケーション] を選択します。

   [パブリケーション作成] ウィザードが表示されます。
4. 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. [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが右側の [選択したテーブル] のリストに追加されます。
6. [WHERE 句] タブでテーブルを選択し、下側のボックスに探索条件を入力します。※削除オプションで、[挿入] ダイアログを使用して探索条件をフォーマットできます。※
7. [終了] をクリックします。

♦ WHERE 句を使用してパブリケーションを作成するには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてリモート・データベースに接続します。

2. パブリケーション対象のテーブルと WHERE 条件を含む CREATE PUBLICATION 文を実行します。

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

例

次の例は、単一のアーティクルで構成され、営業担当者 ID 番号 856 のすべての受注情報が含まれるパブリケーションを作成します。

```
CREATE PUBLICATION pub_orders_samuel_singer
( TABLE SalesOrders
    WHERE SalesRepresentative = 856 )
```

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。
ログ・ファイルが必要です。	ログ・ファイルは必要ありません。

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、ログ・ファイルはトランケートされず、大量のディスク領域が使用される場合があります。	ログ・ファイルが存在すると、同期処理によって同期トランケーション・ポイントは影響されません。このため、パブリケーションが長期間にわたって同期されなくても、ログ・ファイルはトランケートされます。ダウンロード専用のパブリケーションは、ログ・ファイルのトランケーションに影響しません。
ダウンロード専用同期によってスキャンされるログの量を減らすために、時々アップロードを行う必要があります。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。	アップロードを行う必要はありません。

参照

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

既存のパブリケーションの変更

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

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

注意

- 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。
- 変更には十分注意してください。Mobile Link 設定の実行中にパブリケーションを変更すると、エラーが発生し、データが失われることがあります。変更するパブリケーションにサブスクリプションが含まれている場合は、スキーマのアップグレードとして変更処理を行ってください。「リモート・クライアントでのスキーマの変更」 75 ページを参照してください。

♦ 既存のパブリケーションまたはアーティクルのプロパティを修正するには、次の手順に従います (Sybase Central の管理モードの場合)。

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

♦ **アーティクルを追加するには、次の手順に従います (Sybase Central の管理モードの場合)。**

1. SQL Anywhere プラグインを使用して、パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてリモート・データベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. パブリケーションを選択します。
4. [ファイル] メニューから [新規] – [アーティクル] を選択します。[新しいアーティクルの作成] ウィザードが表示されます。
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。
   - 最初のページでテーブルを選択する。
   - 次のページでカラム数を選択する。
   - 最後のページで WHERE 句を入力する (必要な場合)。
6. [完了] をクリックすると、アーティクルが作成されます。

♦ **アーティクルを削除するには、次の手順に従います (Sybase Central の管理モードの場合)。**

1. SQL Anywhere プラグインを使用して、パブリケーションを所有するユーザ、または DBA 権限を持つユーザとしてデータベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. パブリケーションをクリックします。
4. 右ウィンドウ枠で、削除するアーティクルを右クリックし、ポップアップ・メニューから [削除] を選択します。

♦ **既存のパブリケーションを修正するには、次の手順に従います (SQL の場合)。**

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

   「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

♦ 次の文は、customer 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 customer
)
```

「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」 『SQL Anywhere サーバ - SQL リファレンス』も参照してください。

パブリケーションの削除

Sybase Central または DROP PUBLICATION 文のいずれかを使用して、パブリケーションを削除できます。パブリケーションに接続されたサブスクリプションをすべて削除してから、パブリケーションを削除してください。

パブリケーションを削除するには、DBA 権限が必要です。

♦ パブリケーションを削除するには、次の手順に従います (Sybase Central の管理モードの場合)。

1. SQL Anywhere プラグインを使用して、DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. [パブリケーション] フォルダを開きます。
3. 対象のパブリケーションを右クリックし、ポップアップ・メニューから [削除] を選択します。

♦ パブリケーションを削除するには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてリモート・データベースに接続します。
2. DROP PUBLICATION 文を実行します。

   「DROP PUBLICATION 文 [Mobile Link] [SQL Remote]」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

例

次の文は、パブリケーション pub_orders を削除します。

```
DROP PUBLICATION pub_orders
```

SQL Anywhere リモート・データベースでの Mobile Link ユーザの作成

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。

♦ Mobile Link ユーザをリモート・データベースに追加するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインから、DBA 権限のあるユーザとしてデータベースに接続します。
2. [Mobile Link ユーザ] フォルダをクリックします。
3. [ファイル] メニューから [新規] – [Mobile Link ユーザ] を選択します。

   [ユーザの作成] ウィザードが表示されます。
4. Mobile Link ユーザ名を入力します。
5. [終了] をクリックします。

♦ Mobile Link ユーザをリモート・データベースに追加するには、次の手順に従います (SQL の場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. CREATE SYNCHRONIZATION USER 文を実行します。Mobile Link ユーザ名はリモート・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。

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

   ```
   CREATE SYNCHRONIZATION USER SSinger
   ```

   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。

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

   パスワードなど、Mobile Link ユーザのプロパティを設定する方法については、「Mobile Link ユーザの拡張オプションの格納」 98 ページを参照してください。

Mobile Link ユーザの登録については、「統合データベースへの Mobile Link ユーザ名の追加」 11 ページを参照してください。

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。

拡張オプションのリストについては、「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。

♦ データベースに Mobile Link 拡張オプションを保存するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインから、DBA 権限のあるユーザとしてデータベースに接続します。
2. [Mobile Link ユーザ] フォルダを開きます。
3. Mobile Link ユーザ名を右クリックし、ポップアップ・メニューで [プロパティ] を選択します。
4. 必要に応じてプロパティを変更します。

♦ データベースに Mobile Link 拡張オプションを保存するには、次の手順に従います (SQL の場合)。

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

   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
   ```

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

   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。

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

♦ クライアント・イベント・フックを使用して Mobile Link ユーザ・プロパティを指定するには、次の手順に従います。

- 次の同期の動作はプログラムを使用してカスタマイズできます。

  詳細については、「sp_hook_dbmlsync_set_extended_options」 272 ページを参照してください。

参照

- 「dbmlsync 拡張オプションの使用」 103 ページ

Mobile Link ユーザの削除

Mobile Link ユーザは、そのユーザ用のサブスクリプションをすべて削除した後で、リモート・データベースから削除してください。

♦ Mobile Link ユーザをリモート・データベースから削除するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. SQL Anywhere プラグインから、DBA 権限のあるユーザとしてデータベースに接続します。
2. [Mobile Link ユーザ] フォルダで、対象の Mobile Link ユーザを探します。
3. Mobile Link ユーザを右クリックし、ポップアップ・メニューから [削除] を選択します。

♦ Mobile Link ユーザをリモート・データベースから削除するには、次の手順に従います (SQL の場合)。

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

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

   ```
   DROP SYNCHRONIZATION USER SSinger
   ```

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

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。

パブリケーションの作成については、「データのパブリッシュ」 89 ページを参照してください。Mobile Link ユーザの作成については、「SQL Anywhere リモート・データベースでの Mobile Link ユーザの作成」 97 ページを参照してください。

注意
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も指定できます。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必要な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。

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

例

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

1. 最初に、customer テーブルと sales_order テーブルをパブリッシュします。パブリケーション名として testpub を指定します。

```
CREATE PUBLICATION testpub
   (TABLE customer, TABLE sales_order)
```

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サブスクリプションを作成します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO testpub
   FOR demo_ml_user
   TYPE tcpip
```

```
ADDRESS 'host=localhost;port=2439;'
OPTION sv='version1'
```

Mobile Link サブスクリプションの変更

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

♦ 同期サブスクリプションを変更するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. [Mobile Link ユーザ] フォルダを開きます。
3. 対象のユーザをクリックします。プロパティが右ウィンドウ枠に表示されます。
4. 右ウィンドウ枠で、[同期サブスクリプション] タブをクリックします。変更するサブスクリプションを右クリックし、ポップアップ・メニューから [プロパティ] を選択します。
5. 必要に応じてプロパティを変更します。

♦ 同期サブスクリプションを変更するには、次の手順に従います (SQL の場合)。

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

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

Mobile Link サブスクリプションの削除

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

同期サブスクリプションを削除するには、DBA 権限が必要です。

♦ 同期サブスクリプションを削除するには、次の手順に従います (Sybase Central 管理モードの場合)。

1. DBA 権限のあるユーザとしてデータベースに接続します。
2. [Mobile Link ユーザ] フォルダを開きます。
3. Mobile Link ユーザを選択します。

4. 対象のサブスクリプションを右クリックし、ポップアップ・メニューから [削除] を選択します。

♦ **同期サブスクリプションを削除するには、次の手順に従います (SQL の場合)。**

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

例

次の文は、パブリケーション pub_orders に対する Mobile Link ユーザ jsmith の同期サブスクリプションを削除します。

```
DROP SYNCHRONIZATION SUBSCRIPTION
FOR jsmith TO pub_orders
```

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

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 ユーティリティを実行することで同期処理が開始されます。このユーティリティは SQL Anywhere リモート・データベースへの接続と同期を行います。

dbmlsync コマンド・ラインで -c オプションを使用して接続パラメータを指定できます。これらのパラメータは、リモート・データベース用です。接続パラメータを指定しないと、[接続] ダイアログが表示され、必要な接続パラメータと起動オプションを指定するように要求されます。

「-c オプション」 130 ページを参照してください。

クライアントのネットワーク・プロトコル・オプションは、リモート・データベースの同期サブスクリプション、パブリケーション、ユーザに保存するか、dbmlsync コマンド・ラインで指定できます。これらのオプションは、適切な Mobile Link サーバの検索に使用されます。

「CommunicationAddress (adr) 拡張オプション」 173 ページを参照してください。

dbmlsync のパーミッション

dbmlsync でデータベースに接続するときは、処理中のすべての変更を適用するパーミッションが必要です。dbmlsync コマンド・ラインには、この接続用のパスワードが含まれます。このため、セキュリティ上の問題が発生する可能性があります。

セキュリティの問題を回避するには、ユーザ (DBA 以外) に REMOTE DBA 権限を付与し、このユーザ ID を dbmlsync 接続文字列に使用します。REMOTE DBA 権限を付与されたユーザ ID が DBA 権限を持つのは、dbmlsync ユーティリティから接続が確立された場合のみです。同じユーザ ID を使用する他の接続には、特別な権限は付与されません。

「GRANT REMOTE DBA 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

dbmlsync 拡張オプションの使用

Mobile Link には、同期処理をカスタマイズするための拡張オプションがいくつか用意されています。拡張オプションは、パブリケーション、ユーザ、またはサブスクリプションに設定できます。また、拡張オプションの値は、dbmlsync コマンド・ラインでオプションを使用して上書きできます。

拡張オプションの完全なリストは、「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。

♦ **dbmlsync コマンド・ラインで拡張オプションを上書きするには、次の手順に従います。**

- -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を設定するには、次の手順に従います。**

1. 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ことができますが、オプション名は引用符で囲まないでください。

dbmlsync ネットワーク・プロトコル・オプション

dbmlsync 接続情報には、サーバ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。

詳細については、次の項を参照してください。

- 「CommunicationType (ctp) 拡張オプション」 175 ページ
- 「CommunicationAddress (adr) 拡張オプション」 173 ページ

トランザクション・ログ・ファイル

多くの場合、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。オフセットは、サブスクリプションに対するすべての操作がアップロードされ確認された点を示します。

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。データベースを作成するときに、トランザクション・ログの格納場所と、トランザクション・ログを保持するかどうかを指定できます。

スクリプト化されたアップロードを実装したり、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・ログが必要ない場合があります。

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。

参照

- 「バックアップとデータ・リカバリ」 『SQL Anywhere サーバ - データベース管理』
- 「進行オフセット」 87 ページ
- 「トランザクション・ログ」 『SQL Anywhere サーバ - データベース管理』
- 「初期化ユーティリティ (dbinit)」 『SQL Anywhere サーバ - データベース管理』
- 「スクリプト化されたアップロード」 323 ページ

例

古いログ・ファイルがディレクトリ *c:¥oldlogs* に格納されているとします。次のコマンドを使用してリモート・データベースを同期することができます。

```
dbmlsync -c "dbn=remote;uid=syncuser" c:¥oldlogs
```

古いログ・ディレクトリへのパスは、コマンド・ラインに最後の引数として指定してください。

同期中の同時実行性

同期の整合性を確保するために、dbmlsync ではアップロードが構築されてからダウンロードが適用されるまでの間に、ダウンロードのローが修正されないようにする必要があります。

Windows CE 以外のプラットフォームでは、デフォルトの動作として dbmlsync は同期中のパブリケーションで指定されているすべてのテーブルの共有ロックを取得します。Windows CE では、デフォルトの動作として dbmlsync は排他ロックを取得します。dbmlsync は、アップロードの構築を開始する前にロックを取得し、ダウンロードが適用されるまでそのロックを保持します。

ロックの詳細については、「ロー・ロック」 『SQL Anywhere サーバ - SQL の使用法』を参照してください。

このロック動作は、次のオプションを使用してカスタマイズできます。

- -d オプション
- LockTables オプション

-d オプション

このロック・メカニズムを使用しているときに、データベースに別の接続が存在し、その接続に同期テーブルに対するロックがある場合は、同期が失敗します。別のロックが存在しても、同期がすぐに行われるようにする場合は、dbmlsync で -d オプションを使用します。このオプションを指定すると、同期に影響するロックのある接続はデータベースによって削除されるため、同期を進行できます。削除された接続のコミットされていない変更は、ロールバックされます。

詳細については、「-d オプション」 131 ページを参照してください。

LockTables オプション

データの整合性を保持する別の方法は、LockTables 拡張オプションを OFF に設定することです。OFF にすると、アーティクルのテーブルがロックされるのを防ぎます。これにより、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。ダウンロードを受信しても、そのローが修正されている場合はダウンロードは適用されません。この場合、dbmlsync は同期のリトライを行います。リトライは、新しいダウンロードの競合が検出されないかぎり正常に実行されます。

詳細については、「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

詳細については、「ConflictRetries (cr) 拡張オプション」 176 ページを参照してください。

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 2 つの方法があります。

- dbmlsync 統合コンポーネントを使用します。

  詳細については、「dbmlsync 統合コンポーネント」 287 ページを参照してください。

- DLL を呼び出すことのできる言語で開発している場合は、DBTools インタ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 Anywhere のインストール・ディレクトリの *h* 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。

  詳細については、次の項を参照してください。

  - 「dbmlsync の DBTools インタフェース」 315 ページ
  - 「DBSynchronizeLog 関数」 『SQL Anywhere サーバ - プログラミング』
  - 「a_sync_db 構造体」 『SQL Anywhere サーバ - プログラミング』

ActiveSync 同期の使用

ActiveSync は、Microsoft Windows CE ハンドヘルド・デバイス用の同期ソフトウェアです。ActiveSync は、Windows CE デバイスとデスクトップ・コンピュータ間の同期を制御します。ActiveSync 用の Mobile Link プロバイダは、Mobile Link サーバとの同期を制御します。

SQL Anywhere クライアント用の ActiveSync 同期を設定するには、次の手順に従います。

- SQL Anywhere リモート・データベースを ActiveSync 同期用に設定する。

  「ActiveSync 用の SQL Anywhere リモート・データベースの設定」 107 ページを参照してください。

- ActiveSync 用 Mobile Link プロバイダをインストールする。

  「ActiveSync 用 Mobile Link プロバイダのインストール」 108 ページを参照してください。

- SQL Anywhere クライアントを、ActiveSync で使用できるように登録する。

  「ActiveSync 用 SQL Anywhere クライアントの登録」 109 ページを参照してください。

ActiveSync 同期を使用する場合は、ActiveSync ソフトウェアから同期を開始します。ActiveSync 用の Mobile Link プロバイダは、dbmlsync を起動するか、スケジュール文字列でのスケジュールに従ってスリープ中の dbmlsync をウェイクアップできます。

リモート・データベース内で遅延フックを使用して dbmlsync をスリープ・モードに設定することもできますが、ActiveSync 用の Mobile Link プロバイダは、このステータスからは同期を開始できません。

同期スケジュールの詳細については、「同期のスケジュール」 111 ページを参照してください。

ActiveSync 用の SQL Anywhere リモート・データベースの設定

♦ SQL Anywhere リモート・データベースを ActiveSync 用に設定するには、次の手順に従います。

1. 同期タイプ (TCP/IP、TLS、HTTP、または HTTPS) を選択します。

   同期タイプは、同期パブリケーション、同期ユーザ、または同期サブスクリプション用に設定できます。それぞれの設定方法は似ています。ここでは、典型的な CREATE SYNCHRONIZATION USER 文の一部を示します。

   ```
   CREATE SYNCHRONIZATION USER SSinger
   TYPE tcpip
   ...
   ```

2. ADDRESS 句を使用して、ActiveSync 用 Mobile Link プロバイダと Mobile Link サーバ間の通信を指定します。

HTTP または TCP/IP 同期の場合は、CREATE SYNCHRONIZATION USER または CREATE SYNCHRONIZATION SUBSCRIPTION 文の ADDRESS 句によって、Mobile Link クライアントとサーバ間の通信を指定します。ActiveSync の場合、通信は 2 段階で発生します。つまり、デバイス上の dbmlsync ユーティリティからデスクトップ・マシン上の ActiveSync 用 Mobile Link プロバイダへの通信が発生してから、デスクトップ・マシンから Mobile Link サーバへの通信が発生します。ADDRESS 句では、ActiveSync 用 Mobile Link プロバイダと Mobile Link サーバ間の通信を指定します。

次の文は、マシン kangaroo 上の Mobile Link サーバへの TCP/IP 通信を指定します。

```
CREATE SYNCHRONIZATION USER SSinger
TYPE tcpip
ADDRESS 'host=kangaroo;port=2439'
```

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

ActiveSync 用 Mobile Link プロバイダのインストール

インストール・ユーティリティ (*mlasinst.exe*) を使用して、ActiveSync 用 Mobile Link プロバイダをインストールしてから、SQL Anywhere Mobile Link クライアントを ActiveSync 用に登録します。

SQL Anywhere for Windows CE のインストーラによって、ActiveSync 用 Mobile Link プロバイダがインストールされます。SQL Anywhere for Windows CE をインストールする場合、この項で説明する手順を実行する必要はありません。

ActiveSync 用 Mobile Link プロバイダをインストールしたら、各アプリケーションを個別に登録してください。手順については、「ActiveSync 用 SQL Anywhere クライアントの登録」 109 ページを参照してください。

♦ ActiveSync 用 Mobile Link プロバイダをインストールするには、次の手順に従います。

1. 使用中のマシンに ActiveSync ソフトウェアがインストールしてあり、Windows CE デバイスが接続されていることを確認します。
2. 次のコマンドを入力して、Mobile Link プロバイダをインストールします。

   ```
   dbasinst -k desk-path -v dev-path
   ```

   desk-path はプロバイダのデスクトップ・コンポーネント (*mlasdesk.dll*) のロケーション、*dev-path* はデバイス・コンポーネント (*mlasdev.dll*) のロケーションです。

   コンピュータに SQL Anywhere がインストールされている場合、*mlasdesk.dll* は SQL Anywhere のインストール・ディレクトリの *win32* または *win64* サブディレクトリにあり、*mlasdev.dll* はプラットフォーム固有のディレクトリ内の *CE* サブディレクトリにあります。-v または -k を省略すると、デフォルトでこれらのディレクトリが検索されます。

   リモート・プロバイダが開けないというメッセージが表示された場合は、デバイスのソフト・リセットを実行し、コマンドを繰り返します。

詳細については、「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」 29 ページを参照してください。

3. マシンを再起動します。

   マシンを再起動すると、ActiveSync で新しいプロバイダが認識されます。

4. Mobile Link プロバイダを有効にします。

   - [ActiveSync] ウィンドウで [オプション] をクリックします。
   - リストにある [Mobile Link] 項目を有効にして [OK] をクリックし、Mobile Link プロバイダをアクティブにします。
   - 登録されたアプリケーションのリストを表示するには、もう一度 [オプション] をクリックし、Mobile Link プロバイダを選択して [設定] をクリックします。

     アプリケーションの登録についての詳細は、「ActiveSync 用 SQL Anywhere クライアントの登録」 109 ページを参照してください。

ActiveSync 用 SQL Anywhere クライアントの登録

ActiveSync で使用するアプリケーションを登録するには、ActiveSync プロバイダのインストール・ユーティリティを使用する方法と、ActiveSync ソフトウェア自体を使用する方法があります。この項では、ActiveSync ソフトウェアを使用する方法について説明します。

もう 1 つの方法については、「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」 29 ページを参照してください。

♦ SQL Anywhere クライアントを ActiveSync 用に登録するには、次の手順に従います。

1. ActiveSync 用 Mobile Link プロバイダがインストールされていることを確認します。

   詳細は、「ActiveSync 用 Mobile Link プロバイダのインストール」 108 ページを参照してください。

2. デスクトップ・マシンで、ActiveSync ソフトウェアを起動します。
3. [ActiveSync] ウィンドウで [オプション] を選択します。
4. 情報タイプのリストから、[Mobile Link] を選択し、[設定] をクリックします。
5. [Mobile Link 同期] ダイアログで [新規] をクリックします。[プロパティ] ダイアログが表示されます。
6. アプリケーションについて次の情報を入力します。

   - **[アプリケーション名]** ActiveSync ユーザ・インタフェースに表示されるアプリケーションを識別する名前。
   - **[クラス名]** -wc オプションを使用して設定した、dbmlsync クライアントのクラス名。

詳細については、「-wc オプション」 166 ページを参照してください。

- **[パス]** デバイス上の dbmlsync アプリケーションのロケーション。
- **[引数]** ActiveSync が dbmlsync の起動時に使用するコマンド・ライン引数。

  dbmlsync は、2 つのモードのうちの 1 つを使用して開始します。

  - スケジューリング・オプションを指定すると、dbmlsync は停止モードに入ります。この場合、クラス名の設定と一致する値を指定した dbmlsync -wc オプションを使用します。

    詳細については、「-wc オプション」 166 ページと「同期のスケジュール」 111 ページを参照してください。

  - このように指定しないと、dbmlsync は停止モードに入りません。この場合、-k を使用して dbmlsync を停止します。

    詳細については、「-k オプション (旧式)」 142 ページを参照してください。

7. [OK] をクリックしてアプリケーションを登録します。

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。

  「dbmlsync オプションを使用したスケジュールの設定」 111 ページを参照してください。

- 定義した論理に基づいて同期を開始するには、dbmlsync イベント・フックを使用します。この方法は、不定期またはイベントの応答として同期を起動する場合に適しています。この場合、指定したフック・コードによって dbmlsync を自動的に停止できます。

  「イベント・フックを使用した同期の開始」 112 ページを参照してください。

停止

スケジュール・オプションやフックを指定すると、dbmlsync は停止モードになります。停止とは、ログのスキャンに使用される時間を少なくする機能です。停止を実行してパフォーマンスを向上させるには、dbmlsync 拡張オプション HoverRescanThreshold を設定するか、dbmlsync ストアド・プロシージャ sp_hook_dbmlsync_log_rescan を使用します。

詳細については、次の項を参照してください。

- 「HoverRescanThreshold (hrt) 拡張オプション」 186 ページ
- 「sp_hook_dbmlsync_log_rescan」 257 ページ

dbmlsync オプションを使用したスケジュールの設定

dbmlsync をバッチ形式で実行して、同期終了後に停止するように設定する代わりに、dbmlsync を継続的に実行してあらかじめ決められた時間に同期を行うように、SQL Anywhere クライアントを設定することもできます。

同期スケジュールは、拡張オプションとして指定します。同期スケジュールを dbmlsync コマンド・ライン上で指定するか、同期ユーザ、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。

スケジュール構文については、「Schedule (sch) 拡張オプション」 198 ページを参照してください。

拡張オプションの詳細については、次の項を参照してください。

- 「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページ
- 「-eu オプション」 140 ページ

♦ **同期サブスクリプションにスケジュールを追加するには、次の手順に従います。**

1. 同期サブスクリプション内で Schedule 拡張オプションを設定します。次に例を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
FOR mluser
ADDRESS 'host=localhost'
OPTION schedule='weekday@11:30am-12:30pm'
```

dbmlsync -is オプションを使用すると、スケジュールの設定を無効にし、直ちに同期を行うことができます。-is オプションは、スケジュール拡張オプションで指定したスケジュールを無視するよう dbmlsync に指示します。詳細については、「-is オプション」 141 ページを参照してください。

♦ **dbmlsync コマンド・ラインでスケジュールを追加するには、次の手順に従います。**

1. スケジュール拡張オプションを設定します。拡張オプションは -e または -eu で設定します。次に例を示します。

```
dbmlsync -e "sch=weekday@11:30am-12:30pm" ...
```

同期スケジュールがこのどちらかで指定された場合、dbmlsync は同期終了後も停止せず、継続して実行します。

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。

詳細については、「sp_hook_dbmlsync_end」 254 ページを参照してください。

sp_hook_dbmlsync_delay 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。

詳細については、「sp_hook_dbmlsync_delay」 234 ページを参照してください。

dbmlsync の同期のカスタマイズ

dbmlsync クライアント・イベント・フック

イベント・フックでは、SQL ストアド・プロシージャを使用して、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。

クライアント・イベント・フックの詳細については、「SQL Anywhere クライアントのイベント・フック」 217 ページを参照してください。

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 統合コンポーネント** dbmlsync 統合コンポーネントには、ビジュアルと非ビジュアルのプログラミング・インタフェースがあり、同期の開始、同期の進行状況に関する情報の受信、同期イベントの特別な処理の実装を行います。たとえば、dbmlsync 統合コンポーネントを使用して、同期処理を .NET アプリケーションに統合できます。

  詳細については、「dbmlsync 統合コンポーネント」 287 ページを参照してください。

- **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。

  詳細については、「dbmlsync の DBTools インタフェース」 315 ページを参照してください。

SQLAnywhere クライアントのロギング

SQL Anywhere リモート・データベースを使用する Mobile Link アプリケーションを作成する場合、注意が必要なクライアント・ログ・ファイルは 2 種類あります。

- dbmlsync コンソール・ログ
- SQL Anywhere トランザクション・ログ

dbmlsync コンソール・ログ

デフォルトでは、dbmlsync メッセージは dbmlsync コンソールに送信されます。また、-o または -ot オプションを使用して結果をコンソール・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.log* という名前のログ・ファイルに送ります。

```
dbmlsync -o dbmlsync.log ...
```

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。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。

ログファイルのサイズを制御したり、ファイルが最大サイズに達したときの処理を指定したりできます。

- ログ・ファイルを指定して、結果をログ・ファイルに追加する場合は、-o オプションを使用します。
- ログ・ファイルを指定して、結果をログ・ファイルに追加する前にファイルの内容を削除する場合は、-ot オプションを使用します。
- -o または -ot のほかに -os オプションを使用してサイズを指定すると、そのサイズに達したときに、ログ・ファイルの名前が変更され、元の名前の新しいファイルが使用されます。

詳細については、次の項を参照してください。

- 「-o オプション」 147 ページ
- 「-ot オプション」 149 ページ
- 「-os オプション」 148 ページ

−v オプションを使用すると、コンソール・ログ・ファイルに記録され、dbmlsync のウィンドウに表示される情報を制御することができます。

詳細については、「-v オプション」 165 ページを参照してください。

delete_old_logs オプションを使用すると、ログ・ファイルを管理できます。

詳細については、「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」 『SQL Anywhere サーバ - データベース管理』を参照してください。

コンソール・ログを指定しないと、すべての結果がコンソールに表示されます。コンソール・ログを指定すると、コンソールに送信される結果の件数が少なくなります。

SQL Anywhere トランザクション・ログ

「トランザクション・ログ・ファイル」 104 ページを参照してください。

第 7 章

Mobile Link SQL Anywhere クライアント・ユーティリティ [dbmlsync]

目次

dbmlsync 構文

dbmlsync ユーティリティを使用して、SQL Anywhere リモート・データベースと統合データベースの同期を行います。

構文

dbmlsync [*options*] [*transaction-logs-directory*]

オプション	説明
@*data*	指定された環境変数または設定ファイルからオプションを読み込む。「@data オプション」 123 ページを参照してください。
-a	エラー時に再入力のプロンプトを表示しない。「-a オプション」 124 ページを参照してください。
-ap	認証パラメータを指定する。「-ap オプション」 125 ページを参照してください。
-ba *file-name*	ダウンロード・ファイルを適用する。「-ba オプション」 126 ページを参照してください。
-bc *file-name*	ダウンロード・ファイルを作成する。「-bc オプション」 127 ページを参照してください。
-be *string*	ダウンロード・ファイルを作成するときに文字列を追加する。「-be オプション」 128 ページを参照してください。
-bg	ダウンロード・ファイルを作成するときに、そのファイルを新しいリモートに適合するようにする。「-bg オプション」 129 ページを参照してください。
-c *connection-string*	*parm1=value1*; *parm2=value2,…* の形式でデータベース接続パラメータを指定する。このオプションを指定しないと、ダイアログが表示されます。このダイアログで接続情報を指定します。「-c オプション」 130 ページを参照してください。
-d	ロックされているデータベースへの接続のうち、同期されたアーティクルと競合しているものをすべて削除する。「-d オプション」 131 ページを参照してください。
-dc	以前に失敗したダウンロードを続行する。「-dc オプション」 132 ページを参照してください。
-dl	コンソールにログ・メッセージを表示する。「-dl オプション」 133 ページを参照してください。
-drs *bytes*	再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定する。「-drs オプション」 134 ページを参照してください。
-ds	ダウンロード専用同期を実行する。「-ds オプション」 135 ページを参照してください。

オプション	説明
-e "*option=value*"···	拡張オプションを指定する。「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。
-eh	フック関数で発生したエラーを無視する。
-ek *key*	暗号化キーを指定する。「-ek オプション」 138 ページを参照してください。
-ep	暗号化キーを入力するよう要求する。「-ep オプション」 139 ページを参照してください。
-eu	最新の -n オプションで定義されたアップロードに対して拡張オプションを指定する。「-eu オプション」 140 ページを参照してください。
-is	スケジュールを無視する。「-is オプション」 141 ページを参照してください。
-k	完了時にウィンドウを閉じる。「-k オプション (旧式)」 142 ページを参照してください。
-l	使用可能な拡張オプションをリストする。「-l オプション」 143 ページを参照してください。
-mn *password*	新しい Mobile Link パスワードを指定する。「-mn オプション」 144 ページを参照してください。
-mp *password*	Mobile Link パスワードを指定する。「-mp オプション」 145 ページを参照してください。
-n *name*	同期パブリケーション名を指定する。「-n オプション」 146 ページを参照してください。
-o *logfile*	このファイルに出力メッセージのログを取る。「-o オプション」 147 ページを参照してください。
-os *size*	出力ログの最大サイズを指定する (このサイズに達するとログの名前が変更される)。「-os オプション」 148 ページを参照してください。
-ot *logfile*	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る。「-ot オプション」 149 ページを参照してください。
-p	ログスキャンのポーリングを無効にする。「-p オプション」 150 ページを参照してください。
-pc+	同期と同期の間で、Mobile Link サーバへのオープン接続を維持する。「-pc オプション」 151 ページを参照してください。
-pd *dllname*;...	Windows CE 用の指定された dll をプリロードする。「-pd オプション」 152 ページを参照してください。

オプション	説明
-pi	Mobile Link に接続できるかどうかをテストする。「-pi オプション」 153 ページを参照してください。
-pp *number*	ログスキャンのポーリング期間を設定する。「-pp オプション」 154 ページを参照してください。
-q	最小化ウィンドウで実行する。「-q オプション」 155 ページを参照してください。
-r[**a** | **b**]	アップロードのリトライにクライアントの進行状況値を使用する。「-r オプション」 157 ページを参照してください。
-sc	各同期の前にスキーマ情報を再ロードする。「-sc オプション」 158 ページを参照してください。
-tu	トランザクション単位のアップロードを実行する。「-tu オプション」 159 ページを参照してください。
-u *ml_username*	同期する Mobile Link ユーザを指定する。「-u オプション」 161 ページを参照してください。
-uo	アップロード専用同期を実行する。「-uo オプション」 162 ページを参照してください。
-urc *row-estimate*	アップロードされるロー数の推定値を指定する。「-urc オプション」 163 ページを参照してください。
-ux	Solaris と Linux で、コンソールを開く。「-ux オプション」 164 ページを参照してください。
-v[*levels*]	冗長オペレーション。「-v オプション」 165 ページを参照してください。
-wc *classname*	ウィンドウ・クラス名を指定する。「-wc オプション」 166 ページを参照してください。
-x	トランザクション・ログの名前を変更して再起動する。「-x オプション」 167 ページを参照してください。
transaction-logs-directory	トランザクション・ログのロケーションを指定する。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。

説明

コマンド・ラインで 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、またはコマンド・ラインの情報を使用します。

トランザクション・ログ・ファイル　*transaction-logs-directory* には、SQL Anywhere リモート・データベースのトランザクション・ログが格納されているディレクトリを指定します。アクティブな

トランザクション・ログ・ファイルとトランザクション・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、この両方が必要です。次のすべての条件を満たす場合、このパラメータを指定してください。

- 前回の同期の後で、作業ログ・ファイルの内容が削除され、ファイルの名前が変更されている場合
- 名前が変更されたログ・ファイルが格納されているディレクトリ以外のディレクトリから、dbmlsync ユーティリティを実行する場合

詳細については、「トランザクション・ログ・ファイル」 104 ページを参照してください。

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。詳細については、「dbmlsync のフックの概要」 219 ページと「SQL Anywhere クライアントのイベント・フック」 217 ページを参照してください。

dbmlsync の使用　dbmlsync の使用についての詳細は、「同期の開始」 103 ページを参照してください。

参照

- 「同期の開始」 103 ページ
- 「SQL Anywhere クライアントのイベント・フック」 217 ページ
- 「dbmlsync 統合コンポーネント」 316 ページ
- 「dbmlsync の DBTools インタフェース」 315 ページ

@data オプション

指定された環境変数または設定ファイルからオプションを読み込みます。

構文

dbmlsync @*data* …

説明

このオプションを指定すると、環境変数または設定ファイル内にコマンド・ライン・オプションを記述できます。指定された名前に環境変数と設定ファイルの両方が存在する場合、環境変数が使用されます。

設定ファイルの詳細については、「設定ファイルの使用」 『SQL Anywhere サーバ - データベース管理』を参照してください。

設定ファイル内のパスワードなどの情報を保護する場合は、ファイル非表示ユーティリティを使用して、設定ファイルの内容を難読化できます。

「ファイル非表示ユーティリティ (dbfhide)」 『SQL Anywhere サーバ - データベース管理』を参照してください。

-a オプション

dbmlsync がエラー時に再入力のダイアログ・プロンプトを表示しないように指定します。

構文

dbmlsync -a …

-ap 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

dbmlsync -ap "*parameters,...***" …**

説明

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。

参照

- 「認証パラメータ」 『Mobile Link - サーバ管理』
- 「authenticate_parameters 接続イベント」 『Mobile Link - サーバ管理』

-ba 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 "*file-name***" …**

説明

リモート・データベースに適用する既存のダウンロード・ファイルの名前を指定します。オプションでパスを指定できます。パスを指定しない場合、デフォルト・ロケーションは dbmlsync が起動されたディレクトリです。

参照

- 「Mobile Link ファイルベースのダウンロード」 『Mobile Link - サーバ管理』
- 「-bc オプション」 127 ページ
- 「-be オプション」 128 ページ
- 「-bg オプション」 129 ページ

-bc 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 "*file-name*" …

説明

指定された名前でダウンロード・ファイルを作成します。ダウンロード・ファイルにはファイル拡張子 .df を使用してください。

オプションでパスを指定できます。パスを指定しない場合、デフォルト・ロケーションは dbmlsync の現在の作業ディレクトリ (dbmlsync が起動されたディレクトリ) です。

オプションで、ダウンロード・ファイルを作成したときと同じ dbmlsync コマンド・ラインで、-be オプションを使用してリモート・データベースで検証できる文字列を指定したり、-bg オプションを使用して新しいリモート・データベースのダウンロード・ファイルを作成したりできます。

参照

- 「Mobile Link ファイルベースのダウンロード」 『Mobile Link - サーバ管理』
- 「-ba オプション」 126 ページ
- 「-be オプション」 128 ページ
- 「-bg オプション」 129 ページ

-be 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 "*file-name***" -be "***string***" …**

説明

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。

参照

- 「sp_hook_dbmlsync_validate_download_file」 284 ページ
- 「Mobile Link ファイルベースのダウンロード」 『Mobile Link - サーバ管理』
- 「-bc オプション」 127 ページ
- 「-ba オプション」 126 ページ

-bg 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 "*file-name*" -bg …

説明

-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 -bg ダウンロード・ファイルを適用しないでください。

参照

- 「Mobile Link ファイルベースのダウンロード」 『Mobile Link - サーバ管理』
- 「-ba オプション」 126 ページ
- 「-bc オプション」 127 ページ
- 「Mobile Link の世代番号」 『Mobile Link - サーバ管理』
- 「新しいリモートの同期」 『Mobile Link - サーバ管理』

-c 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***" …**

説明

接続文字列では、DBA や REMOTE DBA 権限を使用して SQL Anywhere リモート・データベースに接続するための dbmlsync パーミッションを指定します。この場合は、REMOTE DBA 権限を持つユーザ ID を使用することをおすすめします。

keyword=value の形式で、複数のパラメータ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] ダイアログが表示されます。この接続ダイアログのフィールドで、残りのコマンド・ライン・オプションを指定できます。

SQL Anywhere データベースに接続するための接続パラメータの完全なリストについては、「接続パラメータ」『SQL Anywhere サーバ - データベース管理』を参照してください。

-d 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 …

説明

同期中は、LockTables 拡張オプションを OFF に設定しないかぎり、同期するパブリケーションに関するすべてのテーブルがロックされて他のプロセスによる変更が加えられません。別の接続でこれらのいずれかのテーブルにロックがあると、同期は失敗したり遅延したりする可能性があります。このオプションを指定すると、SQL Anywhere は競合ロックを保持するリモート・データベースへの他の接続をすべて強制的に削除するため、同期はただちに実行されます。

参照

- 「同期中の同時実行性」 105 ページ

-dc 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 …

説明

デフォルトでは、ダウンロード中に Mobile Link で障害が発生すると、ダウンロード・データはリモート・データベースに適用されません。ただし、受信したダウンロードの一部がリモート・デバイスのテンポラリ・ファイルに保存されているため、次に dbmlsync を起動するときに -dc オプションを指定すると、短時間でダウンロードを完了できます。-dc オプションを指定すると、dbmlsync は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ときに、アップロードされる新しいデータがある場合は、再起動可能なダウンロードは失敗します。

また、ContinueDownload 拡張オプションや sp_hook_dbmlsync_end フックを使用して、失敗したダウンロードを再起動することもできます。

参照

- 「失敗したダウンロードの再開」『Mobile Link - サーバ管理』
- 「ContinueDownload (cd) 拡張オプション」 177 ページ
- 「sp_hook_dbmlsync_end」 254 ページ
- 「DownloadReadSize (drs) 拡張オプション」 181 ページ

-dl オプション

ログ・ファイル内のメッセージを表示します。

構文

dbmlsync -dl …

説明

通常、ファイルに出力のログを取るときは、dbmlsync ウィンドウに書き込まれるよりも多くのメッセージがログ・ファイルに書き込まれます。このオプションを使用すると、dbmlsync は通常はファイルにのみ書き込まれる情報をウィンドウにも出力します。このオプションを使用すると、同期の速度に影響する場合があります。

-drs 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

dbmlsync -drs *bytes ...*

説明

-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 ー ダウンロード読み込みサイズ -1 の間です。たとえば、DownloadReadSize が 100 バイトで、497 バイトを読み込んだ後でエラーが発生した場合は、読み込んだ最後の 97 バイトが失われます。このようにして失われたバイト数は、ダウンロードが再起動されたときに再送信されます。

通常は、ダウンロード読み込みサイズの値を大きくすると、正常な同期でのパフォーマンスが向上しますが、エラーが発生したときに再送信されるデータが多くなります。

このオプションの一般的な用途は、通信の信頼性が低いときにデフォルトのサイズを減らすことです。

デフォルトは **32767** です。このオプションを 32767 より大きな値に設定すると、32767 が使用されます。

また、DownloadReadSize 拡張オプションを使用して、ダウンロードの読み込みサイズを指定することもできます。

参照

- 「DownloadReadSize (drs) 拡張オプション」 181 ページ
- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「ContinueDownload (cd) 拡張オプション」 177 ページ
- 「sp_hook_dbmlsync_end」 254 ページ
- 「-dc オプション」 132 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -drs 100
```


-ds 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 …

説明

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、リモート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が失敗した場合は、通常の同期が行われるまで、ダウンロード専用同期は継続してエラーとなります。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」 『Mobile Link - サーバ管理』を参照してください。

参照

- 「アップロード専用の同期とダウンロード専用の同期」 『Mobile Link - サーバ管理』
- 「DownloadOnly (ds) 拡張オプション」 180 ページ
- 「ダウンロード専用のパブリケーション」 93 ページ

-e 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*; ⋯

extended-option:
adr cd cr ctp dbs dir drs ds eh el ft hrt inc isc lt mem mn mp p pp sa sc sch scn st sv toc tor uo v vn vm vo vr vs vu

パラメータ

拡張オプションは、長形式または省略形で指定できます。

「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。

説明

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 sv=test は pub1 と pub2 の両方の同期に適用されます。

```
dbmlsync -e "sv=test" -n pub1 -n pub2
```

拡張オプションは、dbmlsync コンソール・ログと SYSSYNC システム・ビューで確認できます。

単一のアップロードに拡張オプションを指定するには、-eu オプションを使用します。

参照

- 「-eu オプション」 140 ページ
- 「SYSSYNC システム・ビュー」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sp_hook_dbmlsync_set_extended_options」 272 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を起動するときの拡張オプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost;dir=c:¥db¥logs"...
```

-eh 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 …

-ek オプション

高度に暗号化された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* …

説明

高度に暗号化された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。高度に暗号化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。高度に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

-ep オプション

暗号化キーを入力するよう要求します。

構文

dbmlsync -ep …

説明

このオプションでは、暗号化キーを入力するためのダイアログ・ボックスが表示されます。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。高度に暗号化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。高度に暗号化されたデータベースでは、キーを指定しないとコマンドが失敗します。

-eu 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 -n *publication-name* **-eu** *keyword=value*;…

説明

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 sv=test は pub2 の同期のみに適用されます。

```
dbmlsync -n pub1 -n pub2 -eu "sv=test"
```

拡張オプションが複数設定された場合の処理方法については、「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。

拡張オプションの完全なリストは、「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページを参照してください。

-is オプション

スケジュールの指示を無視して、直ちに同期を行います。

構文

dbmlsync -is …

説明

同期をスケジュールする拡張オプションを無視します。

スケジュールの詳細については、「同期のスケジュール」 111 ページを参照してください。

-k オプション (旧式)

同期が終了したときに dbmlsync を停止します。このオプションは推奨されなくなりました。代わりに -qc を使用してください。

構文

dbmlsync -k …

参照

- 「-qc オプション」 156 ページ

-l 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 …

説明

dbmlsync コマンド・ラインと一緒に使用すると、使用可能な拡張オプションがすべて表示されます。

-mn 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* …

説明

Mobile Link ユーザのパスワードを変更します。

詳細については、「Mobile Link ユーザ」 9 ページを参照してください。

参照

- 「MobiLinkPwd (mp) 拡張オプション」 193 ページ
- 「NewMobiLinkPwd (mn) 拡張オプション」 194 ページ
- 「-mp オプション」 145 ページ

-mp 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* ⋯

説明

Mobile Link ユーザ認証用のパスワードを指定します。

詳細については、「Mobile Link ユーザ」 9 ページを参照してください。

参照

- 「MobiLinkPwd (mp) 拡張オプション」 193 ページ
- 「NewMobiLinkPwd (mn) 拡張オプション」 194 ページ
- 「-mn オプション」 144 ページ

-n オプション

同期させるパブリケーションを指定します。

構文

dbmlsync -n *pubname* …

説明

同期パブリケーションの名前です。-n オプションを複数指定すると、複数の同期パブリケーションを同期できます。

-n を使用して複数のパブリケーションを同期するには、次の 2 つの方法があります。

- -n pub1,pub2,pub3 と指定して、1 つのアップロードで pub1、pub2、pub3 をアップロードし、その後に 1 つのダウンロードを行う。

  この方法では、各パブリケーション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で設定した拡張オプションは無視されます。

- -n pub1 -n pub2 -n pub3 と指定して、pub1、pub2、pub3 の 3 つを別個の逐次同期で同期する。

  -n pub1 -n pub2 を指定するなど、連続した同期が非常に速く行われると、サーバ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。

  詳細については、「sp_hook_dbmlsync_delay」 234 ページを参照してください。

-o オプション

出力を dbmlsync コンソール・ログに送信します。

構文

dbmlsync -o *file-name* …

説明

出力をログ・ファイルに追加します。デフォルトでは、画面に出力が表示されます。

参照

- 「-os オプション」 148 ページ
- 「-ot オプション」 149 ページ

-os オプション

dbmlsync コンソール・ログの最大サイズを指定します (このサイズに達するとログの名前が変更されます)。

構文

dbmlsync -os *size* [**K** | **M** | **G**]…

説明

size には、出力メッセージのログを取る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.dbr* に変更します。*yymmdd* は年月日を表し、*xx* は AA ー ZZ のアルファベット順の英字です。

このオプションを使用すると、古いログ・ファイルを手動で削除してディスク領域を解放できます。

参照

- 「-o オプション」 147 ページ
- 「-ot オプション」 149 ページ

-ot オプション

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。

構文

dbmlsync -ot *logfile* …

説明

機能は -o オプションと同じですが、dbmlsync の起動時にログ・ファイルの内容が削除されてからメッセージが書き込まれます。

参照

- 「-o オプション」 147 ページ
- 「-os オプション」 148 ページ

-p 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 …

説明

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、スケジュール・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効果がありません。ポーリングが有効な場合は、設定された間隔で実行されます。これはデフォルトでは 1 分ですが、dbmlsync -pp オプションで変更できます。

このオプションは、拡張オプションの DisablePolling=on とまったく同じです。

参照

- 「DisablePolling (p) 拡張オプション」 178 ページ
- 「PollingPeriod (pp) 拡張オプション」 197 ページ
- 「-pp オプション」 154 ページ

-pc 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+ ...

説明

このオプションを指定すると、dbmlsync は通常どおり Mobile Link サーバに接続しますが、以降の同期で使用できるようにその接続を開いたままにします。永続的な接続は、次のいずれかが発生すると閉じます。

- エラーが発生し、同期に失敗した場合
- 活性チェックのタイムアウトが発生した場合

  「timeout」 65 ページを参照してください。
- 同期は、通信タイプまたはアドレスが異なる場合に開始されます。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) に開始されます。

永続的な接続が閉じている場合は、新しい接続 (永続的) が開きます。

このオプションが最も役に立つのは、クライアントが高い頻度で同期するために、サーバへの接続確立コストが高くなっている場合です。

デフォルトでは、永続的な接続は維持されません。

-pd オプション

Windows CE 用の指定された DLL をプリロードします。

構文

dbmlsync -pd *dllname*;...

説明

dbmlsync を Windows CE で実行するときに暗号化された通信ストリームを使用している場合は、-pd オプションを使用して起動時に適切な DLL がロードされるようにしてください。それ以外の場合、dbmlsync では必要になるまで DLL をロードしません。Windows CE ではリソースが制限されるため、後で DLL をロードするとエラーが発生しやすくなります。

次に、各通信プロトコル用にロードする必要がある DLL を示します。

プロトコル	DLL
ECC	mlcecc10.dll
RSA	mlcrsa10.dll
FIPS	mlcrsafips10.dll

複数の DLL は、セミコロンで区切ったリストで指定します。次に例を示します。

```
-pd mlcrsafips10.dll;mlcrsa10.dll
```

-pi オプション

Mobile Link サーバを ping します。

構文

dbmlsync -pi -c *connection_string* ...

説明

-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
- end_connection

-pp 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**]…

説明

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 はデフォルトでは同期の前に発生する一時停止でログをスキャンします。同期が開始されるとき、ログはすでに少なくとも一部がスキャンされているため、この動作はより効率的です。このデフォルトの動作は、ログスキャンのポーリングと呼ばれます。

このオプションは、ログスキャンの間隔を指定します。サフィックス s、m、h、d を使用して、それぞれ秒、分、時間、日を指定します。デフォルトは **1** 分です。サフィックスを指定しない場合、デフォルトの時間単位は分です。

参照

- 「PollingPeriod (pp) 拡張オプション」 197 ページ
- 「DisablePolling (p) 拡張オプション」 178 ページ
- 「-p オプション」 150 ページ

-q オプション

Mobile Link 同期クライアントを最小化ウィンドウで起動します。

構文

dbmlsync -q …

-qc 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 …

説明

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出力ログ・ファイルが指定されている場合に、同期の完了後に dbmlsync を終了します。

参照

- 「-o オプション」 147 ページ
- 「-ot オプション」 149 ページ

-r 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 } …

説明

進行オフセットの詳細については、「進行オフセット」 87 ページを参照してください。

-rb　リモート・データベースがバックアップからリストアされると、デフォルトの動作がデータ損失の原因となることがあります。この場合、リモート・データベースをリストアした後、初めて dbmlsync を実行するときに -rb を指定します。-rb を使用すると、リモート・データベースに記録されたオフセットが統合データベースから取得したオフセットよりも小さい場合、リモート・データベースに記録されているオフセットからアップロードが継続されます。-rb を使用し、リモート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小さい場合、エラーが報告され、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大きくない場合、エラーが報告され、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。

-sc 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…

説明

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、ほとんどの場合、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。

-tu 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 …

説明

-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、子と親のテーブルでデータが変更されると、Mobile Link はデータの挿入を親のテーブル、子のテーブルの順に行いますが、データの削除は子のテーブル、親の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ることになります。
- 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。ほとんどの場合、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、各トランザクションは正常にアップロードされたときに適用されるため、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が失われることがありません。-tu オプションを使用している場合にアップロード・エラーが発生しても、正常にアップロードされたすべてのトランザクションが適用されます。

-tu オプションを指定すると、Mobile Link は SQL Remote とほぼ同じように動作します。主な相違点は、SQL Remote がすべての変更を発生順にリモート・データベースにレプリケートし、結合を行わないことです。このように動作させるには、リモート・データベースで各データベース操作の後にコミットしてください。

Increment 拡張オプションやスクリプト化されたアップロードに -tu を使用することはできません。

参照

- 「自己参照テーブルからのデータのアップロード」 『Mobile Link - サーバ管理』

-u オプション

Mobile Link ユーザ名を指定します。

構文

dbmlsync -u *ml_username* …

説明

dbmlsync コマンド・ラインでユーザを 1 人指定できます。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。

-uo オプション

同期にはアップロードのみが含まれ、ダウンロードは発生されないよう指定します。

構文

dbmlsync -uo…

説明

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。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」『Mobile Link - サーバ管理』を参照してください。

参照

- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link - サーバ管理』
- 「DownloadOnly (ds) 拡張オプション」 180 ページ
- 「UploadOnly (uo) 拡張オプション」 207 ページ

-urc 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* ⋯

説明

パフォーマンスを改善するために、同期でアップロードされ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。

指定する推定値に関係なく、同期は正しく実行されます。

参照

- 「Memory (mem) 拡張オプション」 191 ページ
- 「大規模なアップロードのロー数の推定」 『Mobile Link - サーバ管理』

-ux オプション

Linux で、メッセージを表示するコンソールを開きます。

構文

dbmlsync -ux…

説明

-ux が指定されている場合、dbmlsync は使用可能な表示を見つけます。たとえば、DISPLAY 環境変数が設定されていなかったり、X-Window サーバが実行されていなかったりしたために、使用可能な表示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・モードでコンソールを実行するには、-q を使用します。

Windows では、コンソールは自動的に表示されます。

参照

- 「-q オプション」 155 ページ

-v 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。高度な冗長性を指定するとパフォーマンスに影響するので、通常は高度な冗長性を使用するのは開発段階だけにしてください。

構文

dbmlsync -v [*levels*] ⋯

説明

-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。

levels の値は次のとおりです。たとえば -vnrsu や -v+cp などのように、次のオプションを一度に 1 つまたは複数指定できます。

- **+** c と p 以外のすべてのログ・オプションをオンにする
- **c** ログ内の接続文字列を公開する
- **p** ログ内のパスワードを公開する
- **n** アップロードとダウンロードされたロー数のログを取る
- **o** 指定したコマンド・ライン・オプションと拡張オプションに関する情報のログを取る
- **r** アップロードとダウンロードされたローの値のログを取る
- **s** フック・スクリプトに関連するメッセージのログを取る
- **u** アップロード関する情報のログを取る

-v オプションと同様の機能を持つ拡張オプションがあります。-v オプションと拡張オプションの両方を指定して、競合が発生した場合は、-v オプションが拡張オプションよりも優先されます。競合が発生しない場合は、冗長ログ・オプションが追加され、指定したすべてのオプションが使用されます。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。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseHooks (vs) 拡張オプション」 209 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「-o オプション」 147 ページ
- 「-ot オプション」 149 ページ

-wc 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* ⋯

説明

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、サーバによって開始される同期を使用しているときなどに、停止モードの dbmlsync をウェイクアップするのに使用可能なウィンドウ・クラス名を指定します。

また、このウィンドウ・クラス名によって、ActiveSync 同期用のアプリケーションが識別されます。ActiveSync 同期に使用するアプリケーションの登録時に、クラス名を指定してください。

このオプションが適用されるのは、Windows だけです。

参照

- 「ActiveSync 用 SQL Anywhere クライアントの登録」 109 ページ
- 「ActiveSync 同期の使用」 107 ページ
- 「Schedule (sch) 拡張オプション」 198 ページに示されている INFINITE キーワード
- 「同期のスケジュール」 111 ページ

例

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

-x オプション

出力メッセージ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。

構文

dbmlsync -x [*size* [**K** | **M** | **G**]…

説明

オプションの *size* は、トランザクション・ログが指定されたサイズより大きい場合にのみ、名前が変更されることを意味します。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。デフォルトのサイズは 0 です。

場合によっては、リモート・データベースのバックアップが実行されたり、データベース・サーバを停止するときにトランザクション・ログの名前を変更する代わりに、統合データベースにデータが同期されます。

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。

参照

- 「スケジュールとイベントの使用によるタスクの自動化」『SQL Anywhere サーバ - データベース管理』
- 「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」『SQL Anywhere サーバ - データベース管理』
- 「CREATE EVENT 文」『SQL Anywhere サーバ - SQL リファレンス』

第 8 章

Mobile Link SQL Anywhere クライアントの拡張オプション

目次

dbmlsync 拡張オプションの概要

拡張オプションは、dbmlsync コマンド・ライン上で -e オプションまたは -eu オプションを使用して指定するか、データベースに格納できます。拡張オプションをデータベースに格納するには、Sybase Central を使用するか、sp_hook_dbmlsync_set_extended_options イベント・フックを使用するか、次のいずれかの文で OPTION 句を使用します。

- CREATE SYNCHRONIZATION SUBSCRIPTION
- ALTER SYNCHRONIZATION SUBSCRIPTION
- CREATE SYNCHRONIZATION USER
- ALTER SYNCHRONIZATION USER
- 同期ユーザを指定しない CREATE SYNCHRONIZATION SUBSCRIPTION (これによって、拡張オプションがパブリケーションに対応付けられる)

優先順位

dbmlsync は、データベースに格納されるオプションとコマンド・ラインで指定されるオプションを結合します。競合するオプションが指定されている場合、dbmlsync は次のようにして競合を解決します。メソッドで指定されるオプションは、次のリストの順番で先に出てくるものが後で出てくるものより優先されます。

1. sp_hook_dbmlsync_set_extended_options イベント・フックで指定されているオプション。
2. 拡張オプションではないコマンド・ラインで指定されたオプション(たとえば、-ds は -e "ds=off" を上書きします)。
3. コマンド・ラインで -eu オプションを使用して指定されているオプション
4. コマンド・ラインで -e オプションを使用して指定されているオプション
5. SQL 文または Sybase Central でサブスクリプションに対して指定されているオプション。[同期モデル展開] ウィザードを使用して Mobile Link モデルを展開すると、拡張オプションが自動的に設定され、サブスクリプションで指定されます。
6. SQL 文または Sybase Central で Mobile Link ユーザに対して指定されているオプション
7. SQL 文または Sybase Central でパブリケーションに対して指定されているオプション

> **注意**
> この優先順位は、前述の SQL 文の TYPE と ADDRESS の各オプションで指定しているような接続パラメータにも適用されます。

拡張オプションは、ログと SYSSYNC システム・ビューで確認できます。

拡張オプションを使用して同期をチューニングする方法については、「dbmlsync 拡張オプションの使用」 103 ページを参照してください。

参照

- 「-e オプション」 136 ページ
- 「-eu オプション」 140 ページ
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
- 「SYSSYNC システム・ビュー」『SQL Anywhere サーバ - SQL リファレンス』
- 「sp_hook_dbmlsync_set_extended_options」 272 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を起動するときの拡張オプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost;dir=c:¥db¥logs"...
```

次の SQL 文は、拡張オプションをデータベースに格納する方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION TO mypub
  FOR mluser
  ADDRESS 'host=localhost'
  OPTION schedule='weekday@11:30am-12:30pm', dir='c:¥db¥logs'
```

次の dbmlsync コマンド・ラインは、オプションとその構文をリストする使用画面を開きます。

```
dbmlsync -l
```

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*; ...

パラメータ

「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 ページを参照してください。

説明

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。

リダイレクタを使用している場合は、「Mobile Link クライアントとサーバのリダイレクタ設定」 『Mobile Link - サーバ管理』を参照してください。

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

このオプションも、パブリケーション、サブスクリプション、またはユーザを作成、変更する SQL 文を使用して、データベースに格納できます。詳細については、次の項を参照してください。

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

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。

「CommunicationType (ctp) 拡張オプション」 175 ページを参照してください。

参照

- 「Mobile Link クライアントのネットワーク・プロトコル・オプション」 35 ページ
- 「Mobile Link クライアントとサーバのリダイレクタ設定」 『Mobile Link - サーバ管理』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "adr=host=localhost"
```

コマンド・ラインで複数のネットワーク・プロトコル・オプションを指定するには、それらを一重引用符で囲みます。次に例を示します。

```
dbmlsync -e "adr='host=somehost;port=5001'"
```


データベースに Address または CommunicationType を格納するには、拡張オプションを使用するか、ADDRESS 句を使用できます。次に例を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  ADDRESS 'host=localhost;port=2439'
```

CommunicationType (ctp) 拡張オプション

Mobile Link サーバ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*; ...

説明

network-protocol には、**tcpip**、**tls**、**http**、**https** のいずれか 1 つを指定します。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。

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

参照

- 「Mobile Link クライアント／サーバ通信の暗号化」『SQL Anywhere サーバ - データベース管理』
- 「CommunicationAddress (adr) 拡張オプション」 173 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ctp=https"
```

データベースに CommunicationType を格納するには、拡張オプションを使用するか、TYPE 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
  TYPE 'tcpip'
```


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*; ...

説明

拡張オプション LockTables を OFF に設定すると (同期されているテーブルに対するロックを dbmlsync で取得しないようにする)、アップロードの構築からダウンロードの適用までにデータベースに適用されるオペレーションを実行できます。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。このオプションは、実行するリトライの回数を制御します。

このオプションは、LockTables オプションが OFF (デフォルトは ON) の場合にだけ役に立ちます。

デフォルトは **-1** です (リトライは無制限に続行されます)。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「競合の解決」 『Mobile Link - サーバ管理』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "cr=5"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION cr='5';
```

ContinueDownload (cd) 拡張オプション

以前に失敗したダウンロードを再起動します。

構文

cd={ ON | OFF }; ...

説明

ダウンロード中に Mobile Link で障害が発生すると、ダウンロード・データはリモート・データベースに適用されません。ただし、受信したダウンロードの一部はリモート・デバイスのテンポラリ・ファイルに格納されるため、後でダウンロードを再起動できます。拡張オプション cd=on を設定すると、dbmlsync はダウンロードを再起動し、前のダウンロードで受信しなかった部分をダウンロードします。残りのデータをダウンロードできる場合は、完全なダウンロードがリモート・データベースに適用されます。

-cd=on を設定した場合、アップロードされる新しいデータがあると、再起動可能なダウンロードは失敗します。

また、SQL Anywhere リモート・データベースの再起動可能なダウンロードは、-dc オプションまたは sp_hook_dbmlsync_end フックを使用して指定することもできます。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「sp_hook_dbmlsync_set_extended_options」 272 ページ
- 「-dc オプション」 132 ページ
- 「sp_hook_dbmlsync_end」 254 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "cd=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


DisablePolling (p) 拡張オプション

ログスキャンの自動ポーリングを無効にします。

構文

p={ ON | OFF }; ...

説明

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、デフォルトでは dbmlsync はスケジュールされた同期の間の時間にログをスキャンします。また、sp_hook_dbmlsync_delay フックが使用されている場合、デフォルトでは dbmlsync は同期直前に生じる一時停止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、ポーリング期間の間待機した後、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期間は 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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「PollingPeriod (pp) 拡張オプション」 197 ページ
- 「-p オプション」 150 ページ
- 「-pp オプション」 154 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "p=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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

DownloadBufferSize (dbs) 拡張オプション

ダウンロード・バッファのサイズを指定します。

構文

dbs=*number*[**K** | **M**]; ...

説明

バッファのサイズ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。

このオプションを 0 に設定すると、dbmlsync はダウンロードのバッファを行いません。このオプションに 0 より大きい値を設定すると、Mobile Link サーバによって通信ストリームからダウンロード・ストリーム全体が読み込まれてから、リモート・データベースに適用されます。オプションで指定された領域にダウンロード・ストリームが収まると、ストリーム全体がメモリ内に保持されます。それ以外の場合は、一部がテンポラリ・ファイルに出力されます。

設定が 0 より大きく 4 KB より小さい場合、dbmlsync は 4 KB のバッファ・サイズを使用し、警告を発します。デフォルトは、Windows CE では **32 K** で、他のすべてのオペレーティング・システムでは **1 M** です。

ダウンロード・バッファリングにより Mobile Link サーバはダウンロードをより早く送信できるので、ダウンロードの受信確認を省略するという利点があります。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "dbs=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 dbs='32k';
```


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 }; ...

説明

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、リモートでの変更が上書きされないようにします。これを実行するには、ログをスキャンし、アップロードされるのを待機している操作に関連するローを検出します。これらのローのいずれかがダウンロードによって修正されると、ダウンロードはロールバックされ、同期は失敗します。この理由で同期が失敗した場合は、問題を訂正するために完全な同期を行う必要があります。

ダウンロード専用同期で同期されたリモートがある場合、ダウンロード専用同期がスキャンするログの量を減らすために、定期的に完全な同期を行ってください。そうしないと、ダウンロード専用同期が完了するのに次第に時間がかかるようになります。これが問題になる場合は、ダウンロード専用パブリケーションを使用すると、同期中のログの問題を防ぐことができます。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」 『Mobile Link - サーバ管理』を参照してください。

デフォルトは **OFF** です (アップロードとダウンロード両方の完全な同期)。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「-ds オプション」 135 ページ
- 「ダウンロード専用のパブリケーション」 93 ページ
- 「アップロード専用の同期とダウンロード専用の同期」 『Mobile Link - サーバ管理』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ds=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ds='ON';
```


DownloadReadSize (drs) 拡張オプション

再起動可能なダウンロードについて、通信障害の後に再送する必要があるデータの最大値を指定します。

構文

drs=*number*[**K**]; ...

説明

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 ー DownloadReadSize -1 の間です。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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「-drs オプション」 134 ページ
- 「失敗したダウンロードの再開」『Mobile Link - サーバ管理』
- 「ContinueDownload (cd) 拡張オプション」 177 ページ
- 「sp_hook_dbmlsync_end」 254 ページ
- 「-dc オプション」 132 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "drs=100"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。


```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION drs='100';
```

ErrorLogSendLimit (el) 拡張オプション

同期エラーが発生した場合、dbmlsync からサーバに送信するリモート・ログ・ファイルのサイズを指定します。

構文

el=*number*[**K** | **M**]; ...

説明

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 サーバに送信する出力ログのバイト数を指定します。dbmlsync 出力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。

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 を十分に大きい値に設定すると、dbmlsync は現在のセッションの出力ログ・メッセージ全体を、Mobile Link サーバに送信します。たとえば、出力ログ・メッセージが古い出力ログ・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 は新しく生成されたエラー・メッセージとログ・メッセージの最後の部分だけを、指定されたサイズの範囲内で送信します。

注意：出力ログのサイズは、使用している冗長性の設定により影響を受けます。冗長性を調節するには、dbmlsync -v オプションを使用するか、"verbose" で始まる dbmlsync 拡張オプションを使用します。詳細については、「-v オプション」 165 ページと次に示す -e 冗長性オプションを参照してください。

- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseHooks (vs) 拡張オプション」 209 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

デフォルトは **32 K** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "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';
```

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 }; ...

説明

デフォルトは **ON** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "ft=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION ft='off';
```


HoverRescanThreshold (hrt) 拡張オプション

スケジュールを使用している場合、再スキャンの実行までに累積可能な廃棄メモリ量を制限します。

構文

hrt=*number*[**K** | **M**]; ...

説明

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 **1 M** です。

コマンド・ラインで複数の -n 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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「sp_hook_dbmlsync_log_rescan」 257 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "hrt=2m"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION hrt='2m';
```


IgnoreHookErrors (eh) 拡張オプション

フック関数内で発生したエラーを無視するように指定します。

構文

eh={ ON | OFF }; ...

説明

デフォルトは **OFF** です。

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

このオプションは、dbmlsync -eh オプションと同じです。

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "eh=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
   TO sales_publication
   FOR ml_user1
   OPTION eh='off';
```


IgnoreScheduling (isc) 拡張オプション

スケジュール設定を無視するように指定します。

構文

isc={ ON | OFF }; ...

説明

ON に設定すると、dbmlsync は拡張オプションで指定されたスケジュール情報を無視して、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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「同期のスケジュール」 111 ページ
- 「Schedule (sch) 拡張オプション」 198 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "isc=off"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION isc='off';
```

Increment (inc) 拡張オプション

インクリメンタル・アップロードを有効にし、インクリメンタル・アップロードのサイズを制御します。

構文

inc=*number*[**K** | **M**]; ...

説明

このオプションは、インクリメンタル・スキャンの最小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。

このオプションを指定すると、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。このオプションは、サイトが完全なアップロードを完了するだけの長い時間接続を維持できない場合に役立ちます。オプションが設定されていない場合、アップロードは 1 つにまとめて送信されます。

このオプションの値は、大まかに各アップロード部分のサイズを指定します。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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "inc=32000"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION inc='32k';
```


LockTables (lt) 拡張オプション

同期されるパブリケーション内のテーブルを同期する前にロックするよう指定します。

構文

lt={ ON | OFF | SHARE | EXCLUSIVE }; ...

説明

SHARE は、dbmlsync が共有モードのすべての同期テーブルをロックすることを意味します。EXCLUSIVE は、dbmlsync が排他モードのすべての同期テーブルをロックすることを意味します。Windows CE 以外のプラットフォームでは、ON は SHARE と同じです。Windows CE デバイスでは、ON は EXCLUSIVE と同じです。デフォルトは **ON** です。

同期中の修正を許可するには、OFF に設定します。OFF に設定すると、dbmlsync では別のメカニズムを使用して、ダウンロードによってローがアップロードされる変更内容で上書きされないようにします。

共有ロックと排他ロックの詳細については、「ロックの仕組み」『SQL Anywhere サーバ - SQL の使用法』と「LOCK TABLE 文」『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

Mobile Link アプリケーションでのテーブルのロックの詳細については、「同期中の同時実行性」 105 ページを参照してください。

同期テーブルが排他モードでロックされているとき (Windows CE デバイスではデフォルト)、他の接続はそのテーブルにアクセスできません。このため、個別の接続を実行する 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。

個別の接続を実行するフックについては、「SQL Anywhere クライアントのイベント・フック」 217 ページを参照してください。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "lt=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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

Memory (mem) 拡張オプション

アップロードを構築するときに dbmlsync で使用するキャッシュ・サイズを指定します。

構文

mem=*number*[**K** | **M**]; ...

説明

アップロードを構築するために使用されるキャッシュのサイズをバイト単位で指定します。キャッシュのサイズを大きくすると、メモリ内に保持できるデータのページが多くなるため、ディスクの読み込みと書き込みの回数が減少し、パフォーマンスが向上します。

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 **1 M** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「-urc オプション」 163 ページ
- 「パフォーマンスに関するヒント」 『Mobile Link - サーバ管理』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mem=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 mem='2m';
```


MirrorLogDirectory (mld) 拡張オプション

古いミラー・ログ・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=*filename*; ...

説明

このオプションにより、次の 2 つの状況のいずれかが発生する場合に、dbmlsync は古いミラー・ログ・ファイルを削除できます。

- ミラー・トランザクション・ログとは別のディレクトリにオフライン・ミラー・ログが置かれる

  または

- リモート・データベース・エンジン以外のマシンで dbmlsync が実行される

通常の設定では、アクティブなミラー・ログと名前の変更されたミラー・トランザクション・ログ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マシンで実行されるため、このオプションは不要であり、古いミラー・ログ・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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「delete_old_logs オプション [Mobile Link クライアント] [SQL Remote] [Replication Agent]」『SQL Anywhere サーバ - データベース管理』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mld=c:¥tmp¥file"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mld='c:¥tmp¥file';
```

MobiLinkPwd (mp) 拡張オプション

Mobile Link パスワードを指定します。

構文

mp=*password*; ...

説明

接続に使用する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。このユーザは、dbmlsync -u オプションで指定することもできます。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 **-e mn** で変更できます。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「NewMobiLinkPwd (mn) 拡張オプション」 194 ページ
- 「-mn オプション」 144 ページ
- 「-mp オプション」 145 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mp=password"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


NewMobiLinkPwd (mn) 拡張オプション

新しいパスワードを指定します。

構文

mn=*new-password*; ...

説明

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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「MobiLinkPwd (mp) 拡張オプション」 193 ページ
- 「-mn オプション」 144 ページ
- 「-mp オプション」 145 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "mp=oldpassword; mn=newpassword"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION mn='SQL';
```

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 }; ...

説明

このオプションが有効なのは、スケジュールが 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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Schedule (sch) 拡張オプション」 198 ページ
- 「同期のスケジュール」 111 ページ

例

次の dbmlsync コマンド・ラインの一部は、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "schedule=EVERY:01:00;nss=off"...
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION nss='off', schedule='EVERY:01:00';
```


OfflineDirectory (dir) 拡張オプション

オフライン・トランザクションのログを含むパスを指定します。

構文

dir=*path*; ...

説明

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "dir=c:¥db¥logs"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION dir='c:¥db¥logs';
```

PollingPeriod (pp) 拡張オプション

ログスキャンのポーリング期間を指定します。

構文

pp=*number*[**S** | **M** | **H** | **D**]; ...

説明

このオプションは、ログスキャンの間隔を指定します。サフィックス s、m、h、d を使用して、それぞれ秒、分、時間、日を指定します。デフォルトは **1** 分です。サフィックスを指定しない場合、デフォルトの時間単位は分です。

ログスキャンのポーリングは、同期をスケジュールしているとき、または sp_hook_dbmlsync_delay フックを使用しているときだけ実行されます。

ログスキャンのポーリングの説明は、「DisablePolling (p) 拡張オプション」 178 ページを参照してください。

このオプションは、**dbmlsync -pp** と同じです。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「DisablePolling (p) 拡張オプション」 178 ページ
- 「-pp オプション」 154 ページ
- 「-p オプション」 150 ページ
- 「sp_hook_dbmlsync_delay」 234 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "pp=5"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION pp='5';
```


Schedule (sch) 拡張オプション

同期のスケジュールを指定します。

構文

sch=*schedule*; ...

schedule : { **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**

パラメータ

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長時間かかる場合、同期はすぐに再開されます。

dbmlsync 起動後すぐ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。「NoSyncOnStartup (nss) 拡張オプション」 195 ページを参照してください。

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) と指定します。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、同期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。

曜日は、Mon、Tue のようになります。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・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。待機中、dbmlsync が実行され、ログが定期的にスキャンされます。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。

詳細については、「NoSyncOnStartup (nss) 拡張オプション」 195 ページを参照してください。

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。

詳細については、「-wc オプション」 166 ページを参照してください。

説明

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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

schedule オプションの構文は、同期 SQL 文と dbmlsync コマンド・ラインのどちらの場合も同じです。

IgnoreScheduling 拡張オプションと -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。詳細については、「IgnoreScheduling (isc) 拡張オプション」 188 ページを参照してください。

スケジュールの詳細については、「同期のスケジュール」 111 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sch=WEEKDAY@8:00am,SUN@9:00pm"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sch='WEEKDAY@8:00am,SUN@9:00pm';
```


ScriptVersion (sv) 拡張オプション

スクリプト・バージョンを指定します。

構文

sv=*version-name*; ...

説明

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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sv=SyaAd001"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION sv='SysAd001';
```


SendColumnNames (scn) 拡張オプション

ダイレクト・ロー・ハンドリングで使用するために、アップロード時にカラム名が送信されるように指定します。

構文

scn={ ON | OFF }; ...

説明

カラム名は、Mobile Link サーバによってダイレクト・ロー・ハンドリングで使用されます。ダイレクト・ロー・ハンドリング API を使用して、インデックスではなく、名前でカラムを参照する場合は、このオプションを設定してください。このオプションで送信されるカラム名は、このような場合にのみ使用されます。

「ダイレクト・ロー・ハンドリング」『Mobile Link - サーバ管理』を参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "scn=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


SendDownloadACK (sa) 拡張オプション

クライアントからサーバ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 }; ...

説明

確認をオフにすると (デフォルト)、統合データベースでの競合が減り、ダウンロード・トランザクションの時間が短縮されるためにスループットが向上します。ダウンロード・トランザクションの時間が短縮されるのは、リモート・クライアントがダウンロードを適用する間 Mobile Link がこれらのトランザクションを開いている必要がなくなり、できるかぎり早急にトランザクションがコミットまたはロールバックされるからです。クライアント側でダウンロード・バッファリングを有効にすると、ダウンロード肯定応答がなくなり最大限のパフォーマンスを実現できます。SendDownloadAck を OFF に設定することをおすすめします。ダウンロードに失敗しても、リモートが同じタイムスタンプを何度もアップロードするため、データが失われることはありません。

ダウンロード確認を OFF にすることによるパフォーマンスの向上については、「ダウンロード確認を有効にしない」 『Mobile Link - サーバ管理』を参照してください。

注意：SendDownloadAck が ON に設定され、冗長モードである場合、受信確認行はクライアント・ログに書き込まれます。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "sa=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 sa='off';
```

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 }; ...

説明

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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "st=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


TableOrder (tor) 拡張オプション

アップロードでのテーブルの順序を指定します。

構文

tor=*tables*; ...

tables = *table-name* [,*table-name*], ...

説明

このオプションを使用すると、アップロードされるリモート・データベースのテーブルの順序を指定できます。*tables* はカンマで区切ったリストで指定します。アップロードされるすべてのテーブルを指定する必要があります。同期に含まれていないテーブルを指定すると、そのテーブルは無視されます。

指定するテーブルの順序は、参照整合性を保つようにします。つまり、Table1 が Table2 を外部キー参照する場合、Table2 は Table1 の前にアップロードする必要があります。適切な順序でテーブルを指定しないと、次の 2 つの場合を除き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。

このオプションを使用しなければならないケースはありません。このパラメータは、特定の順序でテーブルをアップロードするユーザのために提供されています。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「TableOrderChecking (toc) 拡張オプション」 206 ページ
- 「アップロードの処理方法」 『Mobile Link - クイック・スタート』
- 「参照整合性と同期」 『Mobile Link - クイック・スタート』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "tor=admin,parent,child"
```

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,parent,child';
```


TableOrderChecking (toc) 拡張オプション

TableOrder を指定するときは、外部キーを持つテーブルの前に別のテーブルがアップロードされることがないように、dbmlsync でチェックする必要があるかどうかを決めます。

構文

tor={ OFF | ON }; ...

説明

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 により、外部キーを持つテーブルの前ではテーブルは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、外部キーを持つテーブルの前ではテーブルはアップロードされないという規則に従わない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。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「TableOrder (tor) 拡張オプション」 204 ページ
- 「アップロードの処理方法」 『Mobile Link - クイック・スタート』
- 「参照整合性と同期」 『Mobile Link - クイック・スタート』

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "toc=OFF"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

```
CREATE SYNCHRONIZATION SUBSCRIPTION
  TO sales_publication
  FOR ml_user1
  OPTION toc='Off';
```

UploadOnly (uo) 拡張オプション

同期がアップロードだけを含むように指定します。

構文

uo={ ON | OFF }; ...

説明

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 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。

ダウンロード専用同期に定義する必要があるスクリプトのリストについては、「必要なスクリプト」 『Mobile Link - サーバ管理』を参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「アップロード専用の同期とダウンロード専用の同期」 『Mobile Link - サーバ管理』
- 「DownloadOnly (ds) 拡張オプション」 180 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "uo=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


Verbose (v) 拡張オプション

完全冗長を指定します。

構文

v={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「VerboseHooks (vs) 拡張オプション」 209 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "v=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


VerboseHooks (vs) 拡張オプション

フック・スクリプトに関するメッセージのログを取るように指定します。

構文

vs={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「SQL Anywhere クライアントのイベント・フック」 217 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vs=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


VerboseMin (vm) 拡張オプション

少量の情報のログを取るように指定します。

構文

vm={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vm=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


VerboseOptions (vo) 拡張オプション

指定したコマンド・ライン・オプション (拡張オプションを含む) に関する情報のログを取るように指定します。

構文

vo={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vo=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


VerboseRowCounts (vn) 拡張オプション

アップロードおよびダウンロードされるローの数のログを取るように指定します。

構文

vn={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vn=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


VerboseRowValues (vr) 拡張オプション

アップロードおよびダウンロードされるローの値のログを取るように指定します。

構文

vr={ ON | OFF }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseUpload (vu) 拡張オプション」 214 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vr=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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


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 }; ...

説明

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。

詳細については、「-v オプション」 165 ページを参照してください。

デフォルトは **OFF** です。

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

拡張オプションのデータベースへの格納もできます。dbmlsync 拡張オプションの詳細については、「dbmlsync 拡張オプションの概要」 171 ページを参照してください。

参照

- 「Verbose (v) 拡張オプション」 208 ページ
- 「Verbose (v) 拡張オプション」 208 ページ
- 「VerboseMin (vm) 拡張オプション」 210 ページ
- 「VerboseOptions (vo) 拡張オプション」 211 ページ
- 「VerboseRowCounts (vn) 拡張オプション」 212 ページ
- 「VerboseRowValues (vr) 拡張オプション」 213 ページ

例

次の dbmlsync コマンド・ラインは、dbmlsync を使用するときのこのオプションの設定方法を示します。

```
dbmlsync -e "vu=on"
```

次の SQL 文は、このオプションのデータベースへの格納方法を示します。

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

第 9 章

Mobile Link SQL 文

目次

Mobile Link 文

次に、Mobile Link SQL Anywhere クライアントの構成と実行に使用する SQL 文を示します。

- 「ALTER PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「ALTER SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「CREATE SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「DROP PUBLICATION 文 [Mobile Link] [SQL Remote]」『SQL Anywhere サーバ - SQL リファレンス』
- 「DROP SYNCHRONIZATION SUBSCRIPTION 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「DROP SYNCHRONIZATION USER 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「START SYNCHRONIZATION DELETE 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』
- 「STOP SYNCHRONIZATION DELETE 文 [Mobile Link]」『SQL Anywhere サーバ - SQL リファレンス』

Ultra Light クライアント

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

第 10 章

SQL Anywhere クライアントのイベント・フック

目次

dbmlsync のフックの概要

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。したがって、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。
また、フック内の接続設定を変更すると、予期しない結果になる場合があります。フック内の接続設定の変更が必要な場合は、フックに古い値を復元してから、フックを完了してください。

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 統合コンポーネントと dbmlsync 用の DBTools インタフェースが含まれます。

「dbmlsync の同期のカスタマイズ」 113 ページを参照してください。

同期イベント・フックの順序

次の疑似コードは、使用可能なイベントと、同期処理中に各イベントが呼び出されるポイントを示します。たとえば、sp_hook_dbmlsync_abort は最初に呼び出されるイベント・フックです。

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。*_end フックが定義されていても、*_begin フックが定義されていない場合は、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。

```
sp_hook_dbmlsync_abort
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)
for each download segment
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
```

アップロード・オプションの詳細については、「-tu オプション」 159 ページと「Increment (inc) 拡張オプション」 189 ページを参照してください。

イベント・フック・プロシージャの使用

この項では、イベント・フック・プロシージャの設計と使用におけるいくつかの注意事項について説明します。

注意

- イベント・フック・プロシージャでは、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。
- 接続設定は変更しないでください。フック内の接続設定を変更すると、予期しない結果になる場合があります。フック内の接続設定の変更が必要な場合は、フックに古い値を復元してから、フックを完了してください。
- イベント・フック接続は、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 (通常は、REMOTE DBA 権限を持つユーザ)、または dbmlsync ユーザがメンバであるグループの ID のどちらかで所有されなければなりません。
- リモート・データベースは、各フックのインスタンスを 1 つだけ持ちます。所有者が異なるフックのインスタンスは、複数作成しないでください。
- フック・プロシージャは DBA 権限を持つユーザが作成してください。
- *_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。*_end フックが定義されていても、*_begin フックが定義されていない場合は、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。

#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、次の dbmlsync コマンド・ラインでは、sp_hook_dbmlsync_abort フックを呼び出す時点では、#hook_dict テーブルには次のようなローが含まれています。

```
dbmlsync -c 'dsn=MyDsn' -n pub1,pub2 -u MyUser
```

#hook_dict ロー	値
publication_0	pub1
publication_1	pub2
MobiLink user	MyUser
Abort synchronization	false

アボート・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、同期のアボートをフックから dbmlsync に命令することができます。

```
UPDATE #hook_dict
SET value='true'
WHERE name='abort synchronization'
```

各フックの記述には、#hook_dict テーブルのローがリストされます。

例

次のサンプル sp_hook_dbmlsync_delay プロシージャは、#hook_dict テーブルを使用して引数を受け渡す様子を示します。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 ー 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、コンソールに出力します。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin()
BEGIN
  DECLARE syncdef VARCHAR(150);
  SELECT '>>>syncdef = ' || value INTO syncdef
    FROM #hook_dict
    WHERE name ='MobiLink user name';
```

```
    MESSAGE syncdef TYPE INFO TO CONSOLE;
END
```

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、次のプロシージャは例外です。

- sp_hook_dbmlsync_all_error
- sp_hook_dbmlsync_communication_error
- sp_hook_dbmlsync_download_com_error
- sp_hook_dbmlsync_download_fatal_sql_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。

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。この項では、エラーや警告の処理に使用するイベント・フック・プロシージャについて説明します。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」 244 ページ
- 「sp_hook_dbmlsync_download_ri_violation」 246 ページ

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。

「sp_hook_dbmlsync_ml_connect_failed」 265 ページを参照してください。

dbmlsync エラーの処理

dbmlsync エラー・メッセージが生成されるたびに、次のフックが呼び出されます。

- まず、エラーのタイプに応じて、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error のいずれかのフックが呼び出されます。これらのフックには、エラーのタイプに固有の情報が含まれています。たとえば、SQL エラーでは sqlcode と sqlstate が返されます。
- 次に、sp_hook_dbmlsync_all_error が呼び出されます。このフックには、発生したすべてのエラーのログを取る場合に役立ちます。

次の項を参照してください。

- 「sp_hook_dbmlsync_communication_error」 232 ページ
- 「sp_hook_dbmlsync_sql_error」 277 ページ
- 「sp_hook_dbmlsync_misc_error」 262 ページ
- 「sp_hook_dbmlsync_all_error」 227 ページ

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。

「sp_hook_dbmlsync_end」 254 ページを参照してください。

エラーの無視

イベント・フック・プロシージャ内で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、イベント・フック・プロシージャ内でエラーが発生しても無視するように指定できます。

「IgnoreHookErrors (eh) 拡張オプション」 187 ページを参照してください。

sp_hook_dbmlsync_abort

このストアド・プロシージャは、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、dbmlsync はイベント終了後すぐに終了します。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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)	数値	abort synchronization を TRUE に設定すると、この値を使用して、アボートされた同期の終了コードを設定できます。0 は同期が成功したことを示します。他の数は同期が失敗したことを示します。
script version (in|out)	スクリプト・バージョン名	同期に使用される Mobile Link スクリプト・バージョン

説明

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。

abort synchronization の値を True に設定することによりフックがアボートを要求する場合、終了コードが sp_hook_dbmlsync_process_exit_code フックに渡されます。sp_hook_dbmlsync_process_exit_code フックが定義されていない場合、終了コードがプログラムの終了コードとして使用されます。

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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「sp_hook_dbmlsync_process_exit_code」 268 ページ

例

次のプロシージャは、毎日 19:00 ー 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;
```

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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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。 このフックを使用して、sp_hook_dbmlsync_end フックにステータス情報を渡す場合は、sp_hook_dbmlsync_end フックにより同期のリトライなどのアクションを実行することができます。

説明

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* ローはありません。

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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「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」 223 ページ
- 「sp_hook_dbmlsync_communication_error」 232 ページ
- 「sp_hook_dbmlsync_misc_error」 262 ページ
- 「sp_hook_dbmlsync_sql_error」 277 ページ

例

次のテーブルを使用して、リモート・データベース内のエラーのログを取るとします。

```
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;
```

可能性のあるエラーの ID 値を表示するには、dbmlsync のテストを実行します。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;
```

「sp_hook_dbmlsync_end」 254 ページを参照してください。

sp_hook_dbmlsync_begin

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

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"          integer NOT NULL DEFAULT autoincrement ,
  "event_name"       varchar(128) NOT NULL ,
  "ml_user"          varchar(128) NULL ,
  "event_time"       timestamp NULL,
  "table_name"       varchar(128) NULL ,
  "upsert_count"     varchar(128) NULL ,
  "delete_count"     varchar(128) NULL ,
  "exit_code"        integer NULL ,
  "status_retval"    varchar(128) NULL ,
  "pubs"             varchar(128) NULL ,
  "sync_descr "      varchar(128) NULL ,
   PRIMARY KEY ("event_id"),
)
```

次に、パブリケーションのリストをコンパイルする例を示します。同期処理の初めにパブリケーション・リストなどの同期情報のログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_begin ()
BEGIN

   DECLARE pubs_list VARCHAR(1024);
   DECLARE temp_str VARCHAR(128);
   DECLARE qry VARCHAR(128);
```



```
-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
     FROM #hook_dict
     WHERE name LIKE 'publication_%';


-- log publication and synchronization information
    INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
     SELECT 'dbmlsync_begin',#hook_dict.value,pubs_list,CURRENT TIMESTAMP
     FROM #hook_dict
     WHERE name='MobiLink user';
END
```


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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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)	数値	メッセージをユニークに識別する ID。このローを使用すると、エラー・メッセージ・テキストが変更されたときに、エラー・メッセージを識別できます。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。 このフックを使用して、sp_hook_dbmlsync_end フックにステータス情報を渡す場合は、_end フックにより同期のリトライなどのアクションを実行することができます。
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

説明

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。

エラー・コード値のリストについては、「Mobile Link 通信エラー・メッセージ」『SQL Anywhere 10 - エラー・メッセージ』を参照してください。

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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「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」 223 ページ
- 「sp_hook_dbmlsync_all_error」 227 ページ
- 「sp_hook_dbmlsync_misc_error」 262 ページ
- 「sp_hook_dbmlsync_sql_error」 277 ページ

例

次のテーブルを使用して、リモート・データベース内の通信エラーのログを取るとします。

```
CREATE TABLE communication_error_log
(
  error_msg VARCHAR(10240),
  error_code VARCHAR(128)
);
```

次の例では、sp_hook_dbmlsync_communication_error を設定して、通信エラーのログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_communication_error()
BEGIN
 DECLARE msg VARCHAR(255);
 DECLARE code INTEGER;

 // get the error message text
 SELECT value INTO msg
  FROM #hook_dict
  WHERE name ='error message';

 // get the error code
 SELECT value INTO code
  FROM #hook_dict
  WHERE name = 'stream error code';

 // log the error information
 INSERT INTO communication_error_log(error_code,error_msg)
  VALUES (code,msg);
END
```


sp_hook_dbmlsync_delay

このストアド・プロシージャを使用して、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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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同期処理の始めの **sp_hook_dbmlsync_begin** の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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「イベント・フックを使用した同期の開始」 112 ページ
- 「sp_hook_dbmlsync_download_end」 240 ページ

例

次のテーブルを使用して、リモート・データベース上の注文のログを取るとします。

```
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、遅延期間はゼロ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;
```

「sp_hook_dbmlsync_end」 254 ページを参照してください。

sp_hook_dbmlsync_download_begin

このストアド・プロシージャを使用して、同期処理のダウンロード処理開始時にカスタム・アクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"          INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"        VARCHAR(128) NOT NULL ,
  "ml_user"           VARCHAR(128) NULL ,
  "event_time"         TIMESTAMP NULL,
  "table_name"         VARCHAR(128) NULL ,
  "upsert_count"       VARCHAR(128) NULL ,
  "delete_count"       VARCHAR(128) NULL ,
  "exit_code"         INTEGER NULL ,
  "status_retval"      VARCHAR(128) NULL ,
  "pubs"              VARCHAR(128) NULL ,
  "sync_descr "         VARCHAR(128) NULL ,
   PRIMARY KEY ("event_id"),
);
```

次に、パブリケーションのリストをコンパイルする例を示します。同期のダウンロード処理の初めにパブリケーション・リストなどの同期情報のログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_begin ()
BEGIN
```

```
    DECLARE pubs_list VARCHAR(1024);
    DECLARE temp_str VARCHAR(128);
    DECLARE qry VARCHAR(128);


-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
     FROM #hook_dict
     WHERE name LIKE 'publication_%';


-- log publication and synchronization information
    INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
     SELECT 'dbmlsync_download_begin',#hook_dict.value,
      pubs_list,CURRENT TIMESTAMP
     FROM #hook_dict
     WHERE name='MobiLink user';
END;
```

sp_hook_dbmlsync_download_com_error (旧式)

Mobile Link サーバによって送信されたダウンロードの読み込み中に通信エラーが発生した場合は、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。

このフックは使用されなくなりました。「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」 223 ページを参照してください。

#hook_dict テーブルのロー

名前	値	説明
table name (in)	テーブル名	エラーの発生時に操作中だったテーブル。dbmlsync がテーブルを識別できない場合、値は空の文字列になります。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

この名前のプロシージャが存在する場合、同期のダウンロード・フェーズ中に通信エラーが検出されると呼び出されます。その場合、ダウンロードは終了します。

このプロシージャは別個の接続で実行されるため、障害のログを取ることができます。それ以外の場合、ログのアクションは同期アクションとともにロールバックされます。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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、通信エラーのログを取るとします。

```
CREATE TABLE SyncLogComErrorTable
(
```



```
    " user_name "  VARCHAR(255) NOT NULL ,
    " event_time "   TIMESTAMP NOT NULL ,
);
```

次に、Mobile Link サーバによって送信されたダウンロードを読み込み中に通信エラーが発生した場合に、Mobile Link ユーザと現在のタイム・スタンプ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 SyncLogComErrorTable テーブルに格納されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_com_error ()
BEGIN
 INSERT INTO SyncLogComErrorTable (user_name, event_time)
  SELECT #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user';
END;
```


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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「イベント・フックを使用した同期の開始」 112 ページ
- 「sp_hook_dbmlsync_delay」 234 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"         INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"        TIMESTAMP NULL,
  "table_name"        VARCHAR(128) NULL ,
  "upsert_count"      VARCHAR(128) NULL ,
  "delete_count"      VARCHAR(128) NULL ,
  "exit_code"         INTEGER NULL ,
  "status_retval"     VARCHAR(128) NULL ,
  "pubs"             VARCHAR(128) NULL ,
  "sync_descr "       VARCHAR(128) NULL ,
   PRIMARY KEY ("event_id"),
)
```

次に、パブリケーションのリストをコンパイルする例を示します。同期のダウンロード処理の終りにパブリケーション・リストなどの同期情報のログを取ります。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_end ()
BEGIN

    DECLARE pubs_list VARCHAR(1024);
    DECLARE temp_str VARCHAR(128);
    DECLARE qry VARCHAR(128);



-- insert publication list into pubs_list
    SELECT LIST(value) INTO pubs_list
     FROM #hook_dict
     WHERE name LIKE 'publication_%';



-- log publication and synchronization information
    INSERT INTO SyncLog(event_name,ml_user,pubs,event_time)
     SELECT 'dbmlsync_download_end',#hook_dict.value,
      pubs_list,CURRENT TIMESTAMP
     FROM #hook_dict
     WHERE name='MobiLink user';
END
```

sp_hook_dbmlsync_download_fatal_sql_error (旧式)

データベース・エラーによって同期ダウンロードのロールバックが発生すると動作します。

このフックは使用されなくなりました。「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」 223 ページを参照してください。

#hook_dict テーブルのロー

名前	値	説明
table name (in)	テーブル名	エラーの発生時に操作中だったテーブル。dbmlsync がテーブルを識別できない場合、値は空の文字列になります。
SQL error code (in)	SQL エラー・コード	操作が失敗した時にデータベースから返される SQL エラー・コードを識別します。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

この名前のプロシージャが存在する場合、同期のダウンロード・フェーズ中に通信エラーが検出されると呼び出されます。これが発生するのは、無視できない SQL エラーが発生した場合、または sp_hook_dbmlsync_download_SQL_error フックがすでに呼び出されていて、エラーを無視しないように選択されている場合です。

このプロシージャは別個の接続で実行されるため、障害のログを取ることができます。それ以外の場合、ログのアクションは同期アクションとともにロールバックされます。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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「sp_hook_dbmlsync_download_sql_error (旧式)」 248 ページ

例

次のテーブルを使用して、SQL エラーのログを取るとします。

```
CREATE TABLE "DBA"."SyncLogComErrorTable"
(
  " error_code "      VARCHAR(255) NOT NULL ,
  " event_time "      TIMESTAMP NOT NULL ,
);
```

次に、ダウンロードの読み込み中に SQL エラーが発生した場合に、SQL エラー・コードと現在のタイム・スタンプ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 SyncLogSQLErrorTable に格納されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_fatal_sql_error ()
BEGIN
 INSERT INTO SyncLogSQLErrorTable (error_code, event_time)
  SELECT #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'SQL error code';
END;
```


sp_hook_dbmlsync_download_log_ri_violation

ダウンロード・プロセスの参照整合性違反のログを取ります。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
foreign key table (in)	テーブル名	フック呼び出し対象の外部キー・カラムを含むテーブル
primary key table (in)	テーブル名	フック呼び出し対象の外部キーが参照するテーブル
role name (in)	ロール名	フック呼び出し対象の外部キーのロール名
script version (in)	スクリプト・バージョン名	同期に使用される Mobile Link スクリプト・バージョン

説明

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_conflict 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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、この

フックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

参照整合性違反の問題を解決するために、このフックを使用しないでください。このフックはロギングにのみ使用してください。参照整合性違反を解決するには、sp_hook_dbmlsync_download_ri_violation を使用します。

参照

- 「sp_hook_dbmlsync_download_ri_violation」 246 ページ
- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、参照整合性違反のログを取るとします。

```
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 LogRIErrorTable 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;
```


sp_hook_dbmlsync_download_ri_violation

ダウンロード・プロセスの参照整合性違反を解決できます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
foreign key table (in)	テーブル名	フック呼び出し対象の外部キー・カラムを含むテーブル
primary key table (in)	テーブル名	フック呼び出し対象の外部キーが参照するテーブル
role name (in)	ロール名	フック呼び出し対象の外部キーのロール名
script version (in)	スクリプト・バージョン名	同期に使用される Mobile Link スクリプト・バージョン

説明

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_conflict を呼び出します (実装されている場合)。それでも競合が解決されない場合は、dbmlsync がそのローを削除します。参照整合性違反を含む残りの外部キーに対して、このプロセスが繰り返されます。

このフックは、現在同期中のテーブルに関する参照整合性違反がある場合にのみ呼び出されます。同期中ではないテーブルに関する参照整合性違反がある場合、このフックは呼び出されず、同期が失敗します。

このフックは、dbmlsync がダウンロードに使用する接続と同じ接続で呼び出されます。データベースでデータの不整合が発生する可能性があるため、このフックに明示的または暗黙的なコミットが含まれないようにしてください。ダウンロードがコミットまたはロールバックされると、このフックのアクションがコミットまたはロールバックされます。

他のフック・アクションとは異なり、このフック中に実行される操作は次の同期中にアップロードされません。

参照

- 「sp_hook_dbmlsync_download_log_ri_violation」 244 ページ

例

この例は、次に示す Department テーブルと Employee テーブルを使用します。

```
CREATE TABLE Department(
 "department_id"  INTEGER primary key
);

CREATE TABLE Employee(
  "employee_id"      INTEGER PRIMARY KEY,
  "department_id" INTEGER,
  FOREIGN KEY EMPLOYEE_FK1 (department_id) REFERENCES Department
);
```

次の sp_hook_dbmlsync_download_ri_violation の定義は、Department テーブルと Employee テーブル間の参照整合性違反をクリーンアップします。この定義によって、外部キーの役割名が検証され、欠落している department_id の値が Department テーブルに挿入されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_ri_violation()
BEGIN

IF EXISTS (SELECT * FROM #hook_dict WHERE name = 'role name'
 AND value = 'EMPLOYEE_FK1') THEN

 -- update the Department table with missing department_id values
 INSERT INTO Department
  SELECT distinct department_id FROM Employee
  WHERE department_id NOT IN (SELECT department_id FROM Department)

END IF;
```


sp_hook_dbmlsync_download_sql_error (旧式)

Mobile Link サーバによって送信されたダウンロードの適用中に発生したデータベース・エラーを処理します。

このフックは使用されなくなりました。「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」 223 ページを参照してください。

#hook_dict テーブルのロー

名前	値	説明
table name (in)	テーブル名	エラーの発生時に操作中だったテーブル。dbmlsync がテーブルを識別できない場合、値は空の文字列になります。
continue (in|out)	**true** | **false**	エラーを無視して同期を継続するかどうかを示します。sp_hook_dbmlsync_download_fatal_sql_error フックを呼び出し、同期を停止するには、このパラメータを **False** に設定してください。このパラメータを **True** に設定すると、dbmlsync はエラーを無視し、同期を続行します。この結果、データが失われることがあります。
SQL error code (in)	SQL エラー・コード	操作が失敗した時にデータベースから返される SQL エラー・コードを識別します。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

この名前のプロシージャが存在する場合、同期のダウンロード・フェーズ中にデータベース・エラーが検出されたときに呼び出されます。このプロシージャは、エラーを無視して同期を継続できる場合にのみ呼び出されます。致命的なエラーの場合は、sp_hook_dbmlsync_download_fatal_SQL_error プロシージャが呼び出されます。

警告
continue を True に設定すると、dbmlsync はデータベース・エラーを無視し、同期を続行します。失敗した操作をもう一度実行することはありません。このため、ダウンロードの一部またはすべてが失われることがあります。失われるデータの量は、発生したエラーの種類、発生したタイミング、リカバリするためにフックが使用される段階によって異なります。どのデータが失われるかを予測するのは非常に困難であるため、この機能を使用するときは最大限の注意を払ってください。ほとんどの場合、SQL エラーが発生した後は、処理を続行し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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「sp_hook_dbmlsync_download_fatal_sql_error (旧式)」 242 ページ

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)	テーブル名	操作が適用される予定のテーブル
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"          INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"       TIMESTAMP NULL,
  "table_name"       VARCHAR(128) NULL ,
  "upsert_count"     VARCHAR(128) NULL ,
  "delete_count"     VARCHAR(128) NULL ,
  "exit_code"        INTEGER NULL ,
  "status_retval"    VARCHAR(128) NULL ,
  "pubs"             VARCHAR(128) NULL ,
  "sync_descr "      VARCHAR(128) NULL ,
   PRIMARY KEY ("event_id"),
);
```

次に、テーブルがダウンロードされる直前に Mobile Link ユーザ、テーブル名、現在のタイムスタンプのログを取る例を示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_begin()
BEGIN
```



```
    DECLARE tbl VARCHAR(255);

    -- load the table name from #hook_dict
   SELECT #hook_dict.value
    INTO tbl
    FROM #hook_dict
    WHERE #hook_dict.name = 'table name';

   INSERT INTO SyncLog (event_name, ml_user, table_name
     ,event_time)
    SELECT 'download_table_begin', #hook_dict.value, tbl
     ,CURRENT TIMESTAMP
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
END;
```


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)	テーブル名	操作が直前に適用されたテーブル
delete count (in)	ローの数	ダウンロードによってこのテーブルから削除されたローの数
upsert count (in)	ローの数	ダウンロードによってこのテーブルで更新または挿入されたローの数
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"         INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"        TIMESTAMP NULL,
  "table_name"        VARCHAR(128) NULL ,
  "upsert_count"      VARCHAR(128) NULL ,
  "delete_count"      VARCHAR(128) NULL ,
  "exit_code"        INTEGER NULL ,
  "status_retval"     VARCHAR(128) NULL ,
  "pubs"             VARCHAR(128) NULL ,
```



```
    "sync_descr "        VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

次に、テーブルがダウンロードされた直後に Mobile Link ユーザ、テーブル名、挿入または更新されたローの数のログを取る例を示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_download_table_end()
BEGIN
   -- declare variables
   DECLARE tbl VARCHAR(255);
   DECLARE upsertCnt VARCHAR(255);
   DECLARE deleteCnt VARCHAR(255);

   -- load the table name from #hook_dict
   SELECT #hook_dict.value
    INTO tbl
    FROM #hook_dict
    WHERE #hook_dict.name = 'table name';

   -- load the upsert count from #hook_dict
   SELECT #hook_dict.value
    INTO upsertCnt
    FROM #hook_dict
    WHERE #hook_dict.name = 'upsert count';

   -- load the delete count from #hook_dict
   SELECT #hook_dict.value
    INTO deleteCnt
    FROM #hook_dict
    WHERE #hook_dict.name = 'delete count';

   INSERT INTO SyncLog (event_name, ml_user, table_name,
     upsert_count, delete_count, event_time)
    SELECT 'download_table_end', #hook_dict.value, tbl,
     upsertCnt, deleteCnt, CURRENT TIMESTAMP
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
END;
```


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。
exit code (in)	数値	0 (デフォルト) 以外の値が設定された場合は、同期エラーを表します。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ

名前	値	説明
upload status (in)	**not sent** | **committed** | **failed**	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などで発生します。 ◆ **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。
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** の場合は、現在の同期のダウンロードが失敗しており、再起動することができます。**False** の場合は、ダウンロードが正常に行われたか、または再起動することができません。
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

説明

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。

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。

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 -n pub1, -n pub2, ... といった形式で複数のパブリ

ケーションを指定している場合、dbmlsync は最初のパブリケーションを繰り返し同期し、2 番目のパブリケーションを同期しません。

参照

- 「dbmlsync のフックの概要」 219 ページ
- 「同期イベント・フックの順序」 219 ページ
- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「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」 223 ページ

例

次の例では、現在の同期のダウンロードが失敗して再起動が可能な場合、ダウンロードは手動で再起動されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_end()
BEGIN
  -- Restart the download if the download for the current sync
  --  failed and can be restarted
  IF EXISTS (SELECT * FROM #hook_dict
    WHERE name = 'restartable download' AND value=’ true’ )
      THEN
   UPDATE #hook_dict SET value ='download' WHERE name='restart';
  END IF;
END;
```

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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
discarded storage (in)	数値	最後の同期の後で破棄されるメモリのバイト数
rescan (in|out)	**true** | **false**	フックによって True と設定されている場合、dbmlsync は次の同期の前に完全な再スキャンを行います。エントリ時には、この値は False に設定されます。
script version (in)	スクリプト・バージョン名	同期に使用される Mobile Link スクリプト・バージョン

説明

コマンド・ラインで複数の -n オプションを指定すると、メモリ破棄の原因となる断片化が dbmlsync で起きる可能性があります。破棄されたメモリは、データベース・トランザクション・ログの再スキャンによってリカバリできます。このフックにより、dbmlsync を使用してデータベース・トランザクション・ログの再スキャンを行いメモリを回復させるかどうかを決定できます。

再スキャンを強制する他の条件を満たさない場合、このフックは sp_hook_dbmlsync_process_exit_code フックの直後に呼び出されます。

参照

- 「HoverRescanThreshold (hrt) 拡張オプション」 186 ページ

例

次の例では、破棄された記憶領域が 1000 バイトを超える場合に #hook_dict テーブル内の再スキャン・フィールドを TRUE に設定します。

```
CREATE PROCEDURE sp_hook_dbmlsync_log_rescan ()
BEGIN
 IF EXISTS(SELECT * FROM #hook_dict
  WHERE name = 'Discarded storage' AND value>1000)
 THEN
  UPDATE #hook_dict SET value ='true' WHERE name='Rescan';
 END IF;
END;
```


sp_hook_dbmlsync_logscan_begin

このストアド・プロシージャを使用して、アップロード用にトランザクション・ログがスキャンされる直前にカスタム・アクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
starting log offset_*n* (in)	数値	スキャンの開始位置を示すログ・オフセット値。アップロードされるパブリケーションごとに 1 つの値があります。*n* の番号はゼロから始まります。この値は、Publication-*n* に一致します。たとえば、log offset_0 は publication_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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"         INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"       TIMESTAMP NULL,
```



```
    "table_name"         VARCHAR(128) NULL ,
    "upsert_count"       VARCHAR(128) NULL ,
    "delete_count"       VARCHAR(128) NULL ,
    "exit_code"          INTEGER NULL ,
    "status_retval"      VARCHAR(128) NULL ,
    "pubs"               VARCHAR(128) NULL ,
    "sync_descr "        VARCHAR(128) NULL ,
     PRIMARY KEY ("event_id"),
);
```

次に、アップロードのためにトランザクション・ログがスキャンされる直前に Mobile Link ユーザと現在のタイムスタンプのログを取る例を示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_begin ()
BEGIN
 -- log the synchronization event
 INSERT INTO SyncLog (event_name, ml_user,event_time)
  SELECT 'logscan_begin', #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user' ;
END;
```


sp_hook_dbmlsync_logscan_end

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た直後にカスタム・アクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
ending log offset (in)	数値	スキャンの終了位置を示すログ・オフセット値
starting log offset_*n* (in)	数値	同期する各サブスクリプションの初期進行値。*n* 値は、Publication_*n* の値に対応します。たとえば、Starting log offset_1 は Publication_1 のオフセットです。
log scan retry (in)	**true** | **false**	この同期でトランザクション・ログが初めてスキャンされる場合、この値は False、それ以外の場合は True。Mobile Link サーバと 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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

この名前のプロシージャが存在する場合、dbmlsync がアップロードをアセンブルするためにトランザクション・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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"         INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"       TIMESTAMP NULL,
```



```
    "table_name"        VARCHAR(128) NULL ,
    "upsert_count"      VARCHAR(128) NULL ,
    "delete_count"      VARCHAR(128) NULL ,
    "exit_code"         INTEGER NULL ,
    "status_retval"     VARCHAR(128) NULL ,
    "pubs"              VARCHAR(128) NULL ,
    "sync_descr "       VARCHAR(128) NULL ,
     PRIMARY KEY ("event_id"),
);
```

次に、アップロードのためにトランザクション・ログがスキャンされた直後に Mobile Link ユーザと現在のタイムスタンプのログを取る例を示します。#hook_dict の log scan retry パラメータは、トランザクション・ログが複数回スキャンされるかどうかを示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_logscan_end ()
BEGIN

 DECLARE scan_retry VARCHAR(128);

 -- load the scan retry parameter from #hook_dict
 SELECT #hook_dict.value
  INTO scan_retry
  FROM #hook_dict
  WHERE #hook_dict.name = 'log scan retry';

 -- Determine if the log is being rescanned
 -- and log the synchronization event

 IF scan_retry='true' THEN
   INSERT INTO SyncLog (event_name, ml_user,event_time,sync_descr)
    SELECT 'logscan_end', #hook_dict.value, CURRENT TIMESTAMP,
     'Transaction log rescanned'
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
 ELSE
   INSERT INTO SyncLog (event_name, ml_user,event_time,sync_descr)
    SELECT 'logscan_end', #hook_dict.value, CURRENT TIMESTAMP,
     'Transaction log scanned normally'
    FROM #hook_dict
    WHERE name = 'MobiLink user' ;
 END IF;
END;
```


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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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。 このフックを使用して、sp_hook_dbmlsync_end フックにステータス情報を渡す場合は、_end フックにより同期のリトライなどのアクションを実行することができます。

説明

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* ローはありません。

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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「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」 223 ページ
- 「sp_hook_dbmlsync_communication_error」 232 ページ
- 「sp_hook_dbmlsync_all_error」 227 ページ
- 「sp_hook_dbmlsync_sql_error」 277 ページ

例

次のテーブルを使用して、リモート・データベース内のエラーのログを取るとします。

```
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;
```

可能性のあるエラーの ID 値を表示するには、dbmlsync のテストを実行します。たとえば、次の dbmlsync コマンド・ラインは、無効なパブリケーションを参照します。

```
dbmlsync -c eng=custdb;uid=DBA;pwd=sql -n 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。

```
1,9931,
 'There is no synchronization subscription to publication "test"'
```

カスタム・エラー処理を行うには、sp_hook_dbmlsync_misc_error でエラー ID 9931 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 = 9931 THEN
  // handle invalid publication
 END IF;

END;
```

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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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
connection address (in|out)	接続アドレス	フックが呼び出されるとき、これは失敗した直前の通信の試行で使用されたアドレスです。この値を新しい接続アドレスに設定して通信を試行できます。retry を true に設定すると、次の通信の試行でこの値が使用されます。プロトコル・オプションのリストについては、「Mobile Link クライアント・ネットワーク・プロトコル・オプション」 37 ページを参照してください。
connection type (in|out)	ネットワーク・プロトコル	フックが呼び出されるとき、これは失敗した直前の通信の試行で使用されたネットワーク・プロトコル (TCPIP など) です。この値を新しいネットワーク・プロトコルに設定して通信を試行できます。retry を true に設定すると、次の通信の試行でこの値が使用されます。ネットワーク・プロトコルのリストについては、「CommunicationType (ctp) 拡張オプション」 175 ページを参照してください。
user data (in|out)	ユーザ定義のデータ	次の通信の試行が失敗した場合に使用されるステータス情報。たとえば、発生したリトライの回数を保存すると便利です。デフォルトは、空の文字列です。
allow remote ahead (in|out)	**true** | **false**	dbmlsync を -ra オプションで起動した場合だけ true です。このローは、現在の同期のために -ra オプションの読み込みまたは変更を行う場合だけ使用できます。「-r オプション」 157 ページを参照してください。

名前	値	説明
allow remote behind (in|out)	**true** | **false**	dbmlsync を -rb オプションを指定して起動した場合だけ true です。このローは、現在の同期のために -rb オプションの読み込みまたは変更を行う場合だけ使用できます。「-r オプション」 157 ページを参照してください。
retry (in|out)	**true** | **false**	失敗した接続の試行をリトライする場合にこの値を true に設定します。デフォルト値は FALSE です。

説明

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、最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。

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 value
     INTO buf
     FROM #hook_dict
     WHERE name = 'user data';

 IF idx <= 5 THEN
      UPDATE #hook_dict
      SET value = idx
      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 value
  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
       WHERE name = 'user data';

      UPDATE #hook_dict
       SET value = 'TRUE'
       WHERE name = 'retry';
 END IF;
END;
```


sp_hook_dbmlsync_process_exit_code

このストアド・プロシージャを使用して、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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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)	数値	直近の同期試行からの終了コード。0 は同期が成功したことを示します。他の値は同期が失敗したことを示します。この値は、そのフックを使用して同期をアボートするとき、sp_hook_dbmlsync_abort によって設定できます。
last exit code (in)	数値	最後にこのフックが呼び出されたときに #hook_dict テーブルの **new exit code** ローに格納される値、またはこれがフックへの最初の呼び出しの場合は 0
new exit code (in|out)	数値	そのプロセスに対して必要な終了コード。dbmlsync が終了するとき、dblmsync の **exit code** はそのフックへの最後の呼び出しによってこのローに格納される値です。この値は -32768 から 32767 になります。
script version (in)	スクリプト・バージョン名	同期に使用される Mobile Link スクリプト・バージョン

説明

コマンド・ラインで -n 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、あるいは 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* ローはありません。

例

dbmlsync を実行して 5 つの同期を行い、終了コードで失敗した同期の数を示すとします。たとえば、終了コード 0 は失敗がないことを示し、1 は 1 つの同期が失敗したことを示します。これを実現するには、sp_hook_dbmlsync_process_exit_code フックを次のように定義します。この場合、3 つの同期が失敗すると新しい終了コードは 3 になります。

```
CREATE PROCEDURE sp_hook_dbmlsync_process_exit_code()
BEGIN
  DECLARE  rc INTEGER;

  SELECT value INTO rc FROM #hook_dict WHERE name = 'exit code';
  IF rc <> 0 THEN
    SELECT value INTO rc FROM #hook_dict WHERE name = 'last exit code';
    UPDATE #hook_dict SET value = rc + 1 WHERE name = 'new exit code';
  END IF;
END;
```

参照

- 「同期イベント・フックの順序」 219 ページ
- 「sp_hook_dbmlsync_abort」 225 ページ

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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
スクリプト・バージョン	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
drop hook (out)	**never** | **always** | **on success**	次のいずれかの値を取ります。 **never** - (デフォルト) データベースから sp_hook_dbmlsync_schema_upgrade フックを削除しません。 **always** - フックの実行を試行した後、データベースから sp_hook_dbmlsync_schema_upgrade フックを削除します。 **on success** - フックの実行に成功した場合、データベースから sp_hook_dbmlsync_schema_upgrade フックを削除します。dbmlsync の -eh オプションが使用されている場合、または dbmlsync の拡張オプション IgnoreHookErrors が True に設定されている場合、on success は always と同じです。

説明

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。これは、データベースが同期可能である状態を維持するために必要です。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 (LockTables はデフォルトでは on です)。

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。

参照

- 「リモート・クライアントでのスキーマの変更」 75 ページ

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);

 -- If the schema upgrade is successful, drop this hook:
 UPDATE #hook_dict
  SET value = 'on success'
  WHERE name = 'drop hook';
END;
```


sp_hook_dbmlsync_set_extended_options

同期に適用される拡張オプションを指定して、次の同期の動作をプログラムによってカスタマイズするには、このストアド・プロシージャを使用します。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
dxtended options (out)	*opt*=*val*;...	次の同期のために追加される拡張オプション

説明

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、パブリケーションと 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、スケジュール・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。

参照

- 「同期イベント・フックの順序」 219 ページ
- 「Mobile Link SQL Anywhere クライアントの拡張オプション」 169 ページ
- 「優先順位」 171 ページ

例

次の例では、sp_hook_dbmlsync_set_extended_options を使用して、SendColumnNames 拡張オプションを指定します。この拡張オプションは、pub1 が同期される場合だけ適用されます。

```
CREATE PROCEDURE sp_hook_dbmlsync_set_extended_options ()
BEGIN
 IF exists(SELECT * FROM #hook_dict
  WHERE name LIKE 'publication_%' AND value='pub1')
 THEN
   -- specify the SendColumnNames=on extended option
   UPDATE #hook_dict
         SET value = 'SendColumnNames=on'
    WHERE name = 'extended options';
 END IF;
END;
```

sp_hook_dbmlsync_set_ml_connect_info

このストアド・プロシージャを使用して、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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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
connection type (in/out)	tcpip、tls、http、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
connection address (in/out)	プロトコル・オプション	Mobile Link サーバへの接続に使用する通信アドレス。「Mobile Link クライアント・ネットワーク・プロトコル・オプション」 37 ページを参照してください。

説明

このフックを使用すると、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。

参照

- 「dbmlsync のフックの概要」 219 ページ
- 「同期イベント・フックの順序」 219 ページ
- 「CommunicationType (ctp) 拡張オプション」 175 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」 37 ページ
- 「sp_hook_dbmlsync_set_extended_options」 272 ページ

例

```
CREATE PROCEDURE sp_hook_dbmlsync_set_ml_connect_info()
begin
```



```
    UPDATE #hook_dict
    SET VALUE = 'tcpip'
      WHERE name = 'connection type';

    UPDATE #hook_dict
    SET VALUE = 'host=localhost'
      WHERE name = 'connection address';
end
```

sp_hook_dbmlsync_set_upload_end_progress

このストアド・プロシージャは、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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
start progress as timestamp_*n*	進行状況 (タイムスタンプ)	同期中の各パブリケー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、パブリケーション名の識別に使用する整数と同じです。
start progress as bigint_*n*	進行状況 (bigint)	同期中の各パブリケーションの開始進行状況が bigint で示されます。ここで、*n* は、パブリケーション名の識別に使用する整数と同じです。
script version (n)	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 であるとみなされます。 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) として表されるタイムスタンプであるとみなされます。 デフォルト値は FALSE です。

名前	値	説明
end progress (in|out)	タイムスタンプ	フックはこのローを変更して、アップロード・スクリプトに渡される "end progress as bigint" と "end progress as timestamp" 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、"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。

説明

スクリプト化されたアップロードの場合は、アップロード手順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。

参照

- 「スクリプト化されたアップロードのカスタム進行状況値」 335 ページ
- 「同期イベント・フックの順序」 219 ページ
- 「スクリプト化されたアップロード」 323 ページ

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)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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)	数値	メッセージをユニークに識別する ID。このローを使用すると、エラー・メッセージ・テキストが変更されたときに、エラー・メッセージを識別できます。
error hook user state (in|out)	整数	この値はフックによって設定して、今後の呼び出しに対するステータス情報を、sp_hook_dbmlsync_all_error、sp_hook_dbmlsync_communication_error、sp_hook_dbmlsync_misc_error、sp_hook_dbmlsync_sql_error、または sp_hook_dbmlsync_end フックに渡すことができます。これらのフックの 1 つが最初に呼び出されるとき、ローの値は 0 です。フックがローの値を変更した場合は、次回のフックの呼び出しには新しい値が使用されます。 このフックを使用して、sp_hook_dbmlsync_end フックにステータス情報を渡す場合は、_end フックにより同期のリトライなどのアクションを実行することができます。
sql code (in)	SQL エラー・コード	操作が失敗した時にデータベースから返される SQL エラー・コード。これらの値は、SQL Anywhere インストール環境の *h* サブディレクトリにある *sqlerr.h* で定義されます。

名前	値	説明
sql state (in)	SQLSTATE 値	操作が失敗した時にデータベースから返される SQL ステータス

説明

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* ローはありません。

SQL エラーは、SQL Anywhere の SQLCODE または ANSI SQL 規格の SQLSTATE を使用して識別できます。SQLCODE または SQLSTATE 値のリストについては、「データベース・エラー・メッセージ」『SQL Anywhere 10 - エラー・メッセージ』を参照してください。

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。

Windows CE デバイスのデフォルトでは、同期テーブルは排他モードでロックされます。このため、同期テーブルへのアクセスが必要な場合、このフックは正常に実行されません。また、このフックは、同期テーブルへのアクセスが必要で、ユーザが dbmlsync 拡張オプション LockTables を EXCLUSIVE に設定している場合にも実行できません。「LockTables (lt) 拡張オプション」 190 ページを参照してください。

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。

参照

- 「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」 223 ページ
- 「sp_hook_dbmlsync_all_error」 227 ページ
- 「sp_hook_dbmlsync_communication_error」 232 ページ
- 「sp_hook_dbmlsync_misc_error」 262 ページ
- 「データベース・エラー・メッセージ」『SQL Anywhere 10 - エラー・メッセージ』

sp_hook_dbmlsync_upload_begin

このストアド・プロシージャを使用して、アップロード転送の直前にカスタム・アクションを追加します。

#hook_dict テーブルのロー

名前	値	説明
Publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

説明

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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog
(
 "event_id"         INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
  "event_name"       VARCHAR(128) NOT NULL ,
  "ml_user"          VARCHAR(128) NULL ,
  "event_time"        TIMESTAMP NULL,
  "table_name"        VARCHAR(128) NULL ,
  "upsert_count"      VARCHAR(128) NULL ,
  "delete_count"      VARCHAR(128) NULL ,
  "exit_code"         INTEGER NULL ,
  "status_retval"     VARCHAR(128) NULL ,
  "pubs"             VARCHAR(128) NULL ,
  "sync_descr "       VARCHAR(128) NULL ,
   PRIMARY KEY ("event_id"),
);
```

次に、アップロードの転送の直前に Mobile Link ユーザと現在のタイムスタンプのログを取る例を示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_begin ()
BEGIN
 INSERT INTO SyncLog (event_name, ml_user,event_time)
  SELECT 'upload_begin', #hook_dict.value, CURRENT TIMESTAMP
  FROM #hook_dict
```



```
  WHERE name = 'MobiLink user';
END;
```

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)	下の「説明」の値の範囲を参照	アップロード障害の原因。詳細については、下の「説明」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 Link サーバは、アップロードをコミットしませんでした。 **unknown** - dbmlsync が -tu オプションを使用して起動され、トランザクション・レベルのアップロードが発生しました。アップロードされる各トランザクションでは、sp_hook_dbmlsync_upload_begin フックと sp_hook_dbmlsync_upload_end フックが呼び出され、upload status の値は、最後のものを除き、常に **unknown** です。
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
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

名前	値	説明
authentication value (in)	値	この値は、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。

説明

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。

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_ABORTED_UPLOAD** リモートでエラーが発生したため、アップロードに失敗しました。この失敗の一般的な原因には、通信エラーとメモリ不足が挙げられます。
- **UPLD_ERR_COMMUNICATIONS_FAILURE** 通信エラーが発生しました。
- **UPLD_ERR_LOG_OFFSET_MISMATCH** 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・オフセット間に競合があるため、アップロードに失敗しました。
- **UPLD_ERR_GENERAL_FAILURE** アップロードに失敗しました。原因は不明です。
- **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_USERID_ALREADY_IN_USE** このユーザ ID はすでに使用されています。
- **UPLD_ERR_DOWNLOAD_NOT_AVAILABLE** 統合データベース側でアップロードがコミットされましたが、Mobile Link はダウンロードを生成できなかったため、エラーが発生しました。
- **UPLD_ERR_PROTOCOL_MISMATCH** dbmlsync が Mobile Link サーバから予期しないデータを受信しました。
- **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。

参照

- 「同期イベント・フックの順序」 219 ページ

例

次のテーブルを使用して、リモート・データベース上の同期イベントのログを取るとします。

```
CREATE TABLE SyncLog(
   "event_id"          INTEGER NOT NULL DEFAULT AUTOINCREMENT ,
   "event_name"        VARCHAR(128) NOT NULL ,
   "ml_user"           VARCHAR(128) NULL ,
   "event_time"        TIMESTAMP NULL,
   "table_name"        VARCHAR(128) NULL ,
   "upsert_count"      VARCHAR(128) NULL ,
   "delete_count"      VARCHAR(128) NULL ,
   "exit_code"         INTEGER NULL ,
   "status_retval"     VARCHAR(128) NULL ,
   "pubs"              VARCHAR(128) NULL ,
   "sync_descr "       VARCHAR(128) NULL ,
    PRIMARY KEY ("event_id"),
);
```

次に、Mobile Link サーバがアップロードを受信したかを dbmlsync が検証した後で Mobile Link ユーザと現在のタイムスタンプのログを取る例を示します。

```
CREATE PROCEDURE sp_hook_dbmlsync_upload_end ()
BEGIN

 DECLARE status_return_value VARCHAR(255);

 -- store status_return_value
  SELECT #hook_dict.value
  INTO status_return_value
  FROM #hook_dict
  WHERE #hook_dict.name = 'upload status';

   INSERT INTO SyncLog (event_name, ml_user,
     status_retval, event_time)
   SELECT 'upload_end', #hook_dict.value,
     status_return_value, CURRENT TIMESTAMP
  FROM #hook_dict
  WHERE name = 'MobiLink user';
END;
```

sp_hook_dbmlsync_validate_download_file

このフックを使用して、ダウンロード・ファイルがリモート・データベースに適用できるかどうかを決定するためのカスタム論理を実装します。このフックは、ファイルベースのダウンロードが適用される場合のみ呼び出されます。

#hook_dict テーブルのロー

名前	値	説明
publication_*n* (in)	パブリケーション名	同期されているパブリケーション (*n* は整数)。アップロード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* の番号はゼロから始まります。
MobiLink user (in)	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ
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間 (ダウンロード・ファイルには、最後のダウンロード時間とその前のダウンロード時間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間 (ダウンロード・ファイルには、最後のダウンロード時間とその前のダウンロード時間の間に変更されたすべてのローが含まれます)
file creation time (in)		ダウンロード・ファイルが作成された時間
file generation number_*n* (in)	数値	ダウンロード・ファイルからの世代番号。各 publication_*n* エントリに対して、1 つのファイル世代番号 number_*n* があります。publication_*n* と generation number_*n* の *n* は一致します。*n* の番号はゼロから始まります。
user data (in)	文字列	ダウンロード・ファイルが作成されたときに、dbmlsync -be オプションで指定した文字列
apply file (in|out)	**True**|**False**	True (デフォルト) の場合、ダウンロード・ファイルは dbmlsync の他の検証チェックを通過したときだけ適用されます。False の場合、ダウンロード・ファイルはリモート・データベースに適用されません。

名前	値	説明
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 オプションが使用された場合は True。-bg が使用された場合、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。

説明

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 と SYSPUBLICATION システム・ビューから問い合わせることができます。

-ba オプションが指定されていると、このフックが呼び出されます。ダウンロード・ファイルがリモート・データベースに適用される前に呼び出されます。

このフックのアクションは、フックが完了した直後にコミットされます。

参照

- 「-be オプション」 128 ページ
- 「-bg オプション」 129 ページ
- 「Mobile Link ファイルベースのダウンロード」 『Mobile Link - サーバ管理』

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;
```


第 11 章

dbmlsync 統合コンポーネント

目次

dbmlsync 統合コンポーネントの概要

dbmlsync 統合コンポーネントは、アプリケーションに同期を追加するのに使用できる ActiveX です。このコンポーネントでは、SQL Anywhere クライアントの動作を調整するための一連のプロパティ、イベント、メソッドを提供します。

dbmlsync 統合コンポーネントは、2 つの形式で使用できます。いずれも、同じプロパティ、イベント、メソッドを公開しています。

- ビジュアル・コンポーネント。標準の dbmlsync ユーザ・インタフェースをアプリケーションに簡単に統合できます。
- 非ビジュアル・コンポーネント。ユーザ・インタフェースを使用せずに、または独自に作成したカスタム・ユーザ・インタフェースを使用して、コンポーネントの機能にアクセスできます。

dbmlsync 統合コンポーネントを使用すると、アプリケーションで同期を開始し、同期の進行状況についての情報を受け取ってから、同期イベントに基づいた特別な処理を実装できます。

dbmlsync の DBTools インタフェース

dbmlsync 統合コンポーネントを使用する代わりに、dbmlsync 用の DBTools インタフェースを使用することもできます。

「データベース・ツール・インタフェース」『SQL Anywhere サーバ - プログラミング』を参照してください。

サポートされるプラットフォーム

dbmlsync 統合コンポーネントは、ActiveX をサポートする Windows CE バージョンを含め、Mobile Link でサポートされているすべての Windows オペレーティング・システムで使用できます。

サポートされている開発環境には、Microsoft Visual Basic 6.0、eMbedded Visual Basic、Visual Studio .NET があります。

dbmlsync 統合コンポーネントの設定

dbmlsync 統合コンポーネントは、さまざまなプログラミング環境で使用できる ActiveX です。このコンポーネントの設定方法については、ご使用のプログラミング環境のマニュアルを参照してください。

dbmlsync 統合コンポーネントのメソッド

次に、DbmlsyncCOM.Dbmlsync クラスによって実装されるメソッドを示します。

Run メソッド

dbmlsync コマンド・ライン・オプションを使用して、1 つまたは複数の同期を開始します。

構文

Run(ByVal *cmdLine* As String **)**
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

cmdLine　dbmlsync オプションを指定する文字列です。

説明

オプションのリストは、「dbmlsync 構文」 119 ページを参照してください。

Run メソッドはすぐに返され、同期が完了するのを待機しません。DoneExecution イベントを使用して、同期が完了するタイミングを指定することができます。

cmdLine パラメータには、dbmlsync コマンド・ライン・ユーティリティとの同期を実行する場合に使用するのと同じオプションを含める必要があります。たとえば、次のコマンド・ラインと Run メソッドの呼び出しは同じです。

```
dbmlsync -c uid=DBA;pwd=sql

dbmlsync1.Run "-c uid=DBA;pwd=sql"
```

例

次の例は、remote1 というリモート・データベースの同期を開始します。

```
dbmlsync1.Run "-c eng=remote1;uid=DBA;pwd=sql"
```

Stop メソッド

アクティブな同期に終了するように要求します。

構文

Stop( )
Member of **DbmlsyncCOM.Dbmlsync**

説明

Stop メソッドは、アクティブな同期を終了する要求を発行します。このメソッドはすぐに返されます。

ビジュアルな dbmlsync 統合コンポーネントに組み込まれた停止ボタンは、このメソッドを自動的に呼び出します。

例

次の例は、dbmlsync 統合コンポーネントのインスタンスである dbmlsync1 によって実行されている同期を停止します。

```
dbmlsync1.Stop
```

dbmlsync 統合コンポーネントのプロパティ

dbmlsync 統合コンポーネントのプロパティを使用すると、コンポーネントの動作をカスタマイズしたり、実行中の同期の状態を調べたりできます。

Path プロパティ

dbmlsync.exe のロケーションを指定します。

構文

Public Property **Path( )** As String
Member of **DbmlsyncCOM.Dbmlsync**

説明

Windows の PATH 環境変数で指定されているディレクトリに *dbmlsync.exe* が置かれている場合は、このプロパティを設定する必要はありません。

例

次の例は、dbmlsync 統合コンポーネントのインスタンスのパスを設定します。

```
dbmlsync1.Path = "c:¥program files¥sql anywhere 10¥win32"
```

UploadEventsEnabled プロパティ

UploadRow イベントを有効にします。

構文

Public Property **UploadEventsEnabled()** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

UploadRow イベントを処理する場合は、このプロパティを True に設定してください。デフォルトは False で、UploadRow イベントは無効になっています。プロパティを True に設定すると、パフォーマンスが低下します。

「UploadRow イベント」 309 ページを参照してください。

例

次の例は、UploadEventsEnabled を True に設定します。

```
dbmlsync1.UploadEventsEnabled = True
```

DownloadEventsEnabled プロパティ

DownloadRow イベントを有効にします。

構文

Public Property **DownloadEventsEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

DownloadRow イベントを処理する場合は、このプロパティを True に設定してください。デフォルトは False で、DownloadRow イベントは無効になっています。プロパティを True に設定すると、パフォーマンスが低下します。

「DownloadRow イベント」 301 ページを参照してください。

例

次の例は、DownloadEventsEnabled を True に設定します。

```
dbmlsync1.DownloadEventsEnabled = True
```

ErrorMessageEnabled プロパティ

MsgError 型のメッセージに対して Message イベントが呼び出されないようにします。

構文

Public Property **ErrorMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

エラー情報を Message イベントで処理しない場合は、パフォーマンスを向上させるため、このプロパティを False に設定してください。デフォルトは True で、Message イベントをトリガする MsgError 型のメッセージが有効になっています。

「Message イベント」 304 ページを参照してください。

例

次の例は、ErrorMessageEnabled を False に設定します。

```
dbmlsync1.ErrorMessageEnabled = False
```

WarningMessageEnabled プロパティ

MsgWarning 型のメッセージに対して Message イベントが呼び出されないようにします。

構文

Public Property **WarningMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

警告情報を Message イベントで処理しない場合は、パフォーマンスを向上させるため、このプロパティを False に設定してください。デフォルトは True で、Message イベントをトリガする MsgWarning 型のメッセージが有効になっています。

「Message イベント」 304 ページを参照してください。

例

次の例は、WarningMessageEnabled を False に設定します。

```
dbmlsync1.WarningMessageEnabled = False
```

InfoMessageEnabled プロパティ

MsgInfo 型のメッセージに対して Message イベントが呼び出されないようにします。

構文

Public Property **InfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

通常の進行状況情報を Message イベントで処理しない場合は、パフォーマンスを向上させるため、このプロパティを False に設定してください。デフォルトは True で、Message イベントをトリガする MsgInfo 型のメッセージが有効になっています。

「Message イベント」 304 ページを参照してください。

例

次の例は、InfoMessageEnabled を False に設定します。

```
dbmlsync1.InfoMessageEnabled = False
```

DetailedInfoMessageEnabled プロパティ

MsgDetailedInfo 型のメッセージに対して Message イベントが呼び出されないようにします。

構文

Public Property **DetailedInfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

詳細な進行状況情報を Message イベントで処理しない場合は、パフォーマンスを向上させるため、このプロパティを False に設定してください。デフォルトは True で、Message イベントをトリガする MsgDetailedInfo 型のメッセージが有効になっています。

「Message イベント」 304 ページを参照してください。

例

次の例は、DetailedInfoMessageEnabled を False に設定します。

```
dbmlsync1.DetailedInfoMessageEnabled = False
```

UseVB6Types プロパティ

Visual Basic 6 を使用している場合は、このプロパティを True に設定して、UploadRow イベントと DownloadRow イベントから返されるロー・データの処理を簡素化します。

構文

Public Property **DetailedInfoMessageEnabled( )** As Boolean
Member of **DbmlsyncCOM.Dbmlsync**

説明

Visual Basic 6 では、符号なしの 32 ビット値とすべての 64 ビット値がサポートされていません。これらの型のデータは、IRowTransferData オブジェクトの ColumnValue プロパティから返されることがあります。UseVB6Types を True に設定すると、これらの型のデータは Visual Basic 6 でサポートされる別の型に変換され、処理が簡素化されます。Uint32 の値は double 型に変換され、64 ビット値は文字列に変換されます。

参照

- 「IRowTransferData インタフェース」 311 ページ
- 「UploadRow イベント」 309 ページ
- 「DownloadRow イベント」 301 ページ

例

次の例は、Visual Basic 6.0 で使用される dbmlsync 統合コンポーネントのインスタンスのデータ型の強制変換を有効にします。

```
dbmlsync1.UseVB6Types = True
```

ExitCode プロパティ

最新の Run メソッドの呼び出しによって開始された同期からの終了コードを返します。

構文

Public Property **ExitCode( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

説明

ExitCode プロパティは、最後に呼び出された Run メソッドによって開始された同期の終了コードを返します。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。

> **注意：**
> DoneExecution イベントがトリガされる前にこのプロパティの値を取得すると、終了コードが無効な値になることがあります。

例

次の例は、DoneExecution イベントがトリガされた時点での最新の同期からの終了コードを表示します。

```
Private Sub dbmlsync1_DoneExecution() Handles dbmlsync1.DoneExecution
    MsgBox(dbmlsync1.ExitCode)
End Sub
```

EventChannelSize プロパティ

メソッドの呼び出しの処理に使用される内部バッファのサイズを指定します。

構文

Public Property **EventChannelSize( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

説明

ほとんどの場合、このプロパティを変更する必要はありません。

DispatchChannelSize プロパティ

イベント情報の処理に使用される内部バッファのサイズを指定します。

構文

Public Property **DispatchChannelSize( )** As Integer
Member of **DbmlsyncCOM.Dbmlsync**

説明

ほとんどの場合、このプロパティを変更する必要はありません。

dbmlsync 統合コンポーネントのイベント

イベントは、クライアント・アプリケーションが同期の進行状況の情報を受け取り、それに基づいてアクションを行うためのメカニズムを提供します。

BeginDownload イベント

BeginDownload イベントは、同期のダウンロード処理開始時にトリガされます。

構文

Public Event **BeginDownload( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、同期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。

例

次に示す Visual Basic .NET の例は、BeginDownload イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_BeginDownload()
Handles dbmlsync1.BeginDownload

    MsgBox("Beginning Download")

End Sub
```

BeginLogScan イベント

BeginLogScan イベントは、アップロードをアセンブルするために dbmlsync がトランザクション・ログをスキャンする直前にトリガされます。このイベントは、スクリプト化されたアップロードでは起動されません。

構文

Public Event **BeginLogScan(** ByVal *rescanLog* As Boolean **)**
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

rescanLog　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。

説明

このイベントを使用して、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。

例

次に示す Visual Basic .NET の例は、BeginLogScan イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_BeginLogScan(
 ByVal rescanLog As Boolean
)
Handles dbmlsync1.BeginLogScan

      MsgBox("Begin Log Scan")

End Sub
```

BeginSynchronization イベント

BeginSynchronization イベントは、各同期の開始時にトリガされます。

構文

Public Event **BeginSynchronization(_**
  ByVal *userName* As String, _
  ByVal *pubNames* As String _
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

userName　同期対象となる Mobile Link ユーザ。

pubNames　同期されるパブリケーション。パブリケーションが複数の場合は、カンマで区切ったリストで指定します。

説明

このイベントを使用して、同期の開始時にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、BeginSynchronization イベントがトリガされたときにメッセージを出力します。メッセージには、ユーザ名とパブリケーション名が出力されます。

```
Private Sub dbmlsync1_BeginSynchronization(
 ByVal userName As String,
 ByVal pubNames As String
)
Handles dbmlsync1.BeginSynchronization

     MsgBox("Beginning synchronization for: " + userName _
    + " publication: " + pubNames)

End Sub
```

BeginUpload イベント

BeginUpload イベントは、アップロード転送の直前にトリガされます。

構文

Public Event **BeginUpload( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、Mobile Link サーバへの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。

例

次に示す Visual Basic .NET の例は、BeginUpload イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_BeginUpload()
Handles dbmlsync1.BeginUpload

    MsgBox("Begin Upload")

End Sub
```

ConnectMobilink イベント

ConnectMobilink イベントは、コンポーネントが Mobile Link サーバに接続する直前にトリガされます。

構文

Public Event **ConnectMobilink( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、リモート・データベースが Mobile Link サーバに接続する直前にカスタム・アクションを追加します。この段階では、dbmlsync はアップロードの生成を完了しています。

ConnectMobiLink イベントは、BeginSynchronization イベントの後で発生します。

例

次に示す Visual Basic .NET の例は、ConnectMobilink イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_ConnectMobilink()
Handles dbmlsync1.ConnectMobilink

    MsgBox("Connecting to the MobiLink server")

End Sub
```


DisconnectMobilink イベント

DisconnectMobilink イベントは、コンポーネントが Mobile Link サーバから切断した直後にトリガされます。

構文

Public Event **DisconnectMobilink( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、リモート・データベースが Mobile Link サーバから切断した直後にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、DisconnectMobilink イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_DisconnectMobilink()
Handles dbmlsync1.DisconnectMobilink

    MsgBox("Disconnected from the MobiLink server")

End Sub
```

DoneExecution イベント

DoneExecution イベントは、Run メソッドの呼び出しによって開始された同期がすべて完了したときにトリガされます。

構文

Public Event **DoneExecution( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、Run メソッドの呼び出しによって開始された同期がすべて完了したときにカスタム・アクションを追加します。

例

ExitCode プロパティを使用する次の Visual Basic .NET 例は、最後に呼び出された Run メソッドによって開始された同期からの終了コードを出力します。

```
Private Sub dbmlsync1_DoneExecution()
Handles dbmlsync1.DoneExecution

    MsgBox(dbmlsync1.ExitCode)

End Sub
```

DownloadRow イベント

DownloadRow イベントは、Mobile Link サーバからローがダウンロードされるときにトリガされます。

構文

Public Event **DownloadRow(**
ByVal *rowData* As DbmlsyncCOM.IRowTransferData
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

rowData　ダウンロードされるローの詳細な情報が含まれている IRowTransferData オブジェクト。

IRowTransferData インタフェースの詳細については、「IRowTransferData インタフェース」 311 ページを参照してください。

説明

このイベントを使用して、Mobile Link サーバからダウンロードされるローを調べます。

DownloadRow イベントを有効にするには、DownloadEventsEnabled プロパティを使用します。

「DownloadEventsEnabled プロパティ」 292 ページを参照してください。

ローのダウンロード・イベントで削除操作が発生したときは、プライマリ・キー・カラムの値しか使用できません。

例

次に示す Visual Basic .NET の例は、DownloadRow イベントでローのすべてのカラムを反復処理します。この処理で、カラム値が null であるかどうかが判別され、カラム名と値が出力されます。

```
Private Sub dbmlsync1_DownloadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.DownloadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
   If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
      ' output the non-null column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + CStr(rowData.ColumnValue(liX)))
   Else
      ' output 'NULL' for the column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + "NULL")
   End If
Next liX

End Sub
```


EndDownload イベント

EndDownload イベントは、同期処理のダウンロード処理終了時にトリガされます。

構文

Public Event **EndDownload(**
  long *upsertRows*,
  long *deleteRows*
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

upsertRows　ダウンロード処理で更新または挿入されたローの数を示します。

deleteRows　ダウンロード処理で削除されたローの数を示します。

説明

このイベントを使用して、同期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。

例

次に示す Visual Basic .NET の例は、EndDownload イベントがトリガされたときに、挿入、更新、削除されたローの数とメッセージを出力します。

```
Private Sub dbmlsync1_EndDownload(
 ByVal upsertRows As Integer,
 ByVal deleteRows As Integer
)
Handles dbmlsync1.EndDownload

    MsgBox("Download complete." + _
     CStr(upsertRows) + "Rows updated or inserted" + _
     CStr(deleteRows) + "Rows deleted")

End Sub
```

EndLogScan イベント

EndLogScan イベントは、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た直後にトリガされます。このイベントは、スクリプトに記述されたアップロードでは起動されません。

構文

Public Event **EndLogScan( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、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た直後にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、EndLogScan イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_EndLogScan()
Handles dbmlsync1.EndLogScan

    MsgBox("Scan of transaction log complete...")

End Sub
```

EndSynchronization イベント

EndSynchronization イベントは、同期の完了時にトリガされます。

構文

Public Event **EndSynchronization(**
ByVal *exitCode* As Integer,
ByRef *restart* As Boolean
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

exitCode　0 (ゼロ) 以外の値に設定されている場合は、同期エラーが発生したことを示します。

restart　このイベントが呼び出されるときは、この値は False に設定されます。イベントによって値が True に変更された場合、dbmlsync は同期を再起動します。

説明

このイベントを使用して、同期の完了時にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、EndSynchronization イベントを使用して、最大で 5 つの失敗した同期を再起動します。同期の再起動にすべて失敗すると、"All restart attempts failed" というメッセージと、終了コードが出力されます。同期が成功すると、"Synchronization succeeded" というメッセージと、終了コードが出力されます。

```
' Global variable for the number of restarts
Dim numberOfRestarts As Integer

Private Sub dbmlsync1_EndSynchronization(
 ByVal ExitCode As Integer,
 ByRef restart As Boolean
)
Handles dbmlsync1.EndSynchronization

    If numberOfRestarts < 5 Then
        MsgBox("Restart Number: " + CStr(numberOfRestarts + 1))
        If ExitCode <> 0 Then
            ' restart the failed synchronization
            restart = True
            numberOfRestarts = numberOfRestarts + 1
```



```
        Else
            ' the last synchronization succeeded
            MsgBox("Synchronization succeeded. " + _
              "Exit code: " + CStr(ExitCode))
        End If
    Else
        MsgBox("All restart attempts failed. " + _
            "Exit code: " + CStr(ExitCode))
    End If

End Sub
```

EndUpload イベント

EndUpload イベントは、Mobile Link サーバへのアップロード転送の直後にトリガされます。

構文

Public Event **EndUpload( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、dbmlsync から Mobile Link サーバへのアップロード転送の直後にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、EndUpload イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_EndUpload()
Handles dbmlsync1.EndUpload

    MsgBox("End Upload")

End Sub
```

Message イベント

Message イベントは、dbmlsync が情報のログを取るときにトリガされます。

構文

Public Event **Message(_**
ByVal *msgClass* As DbmlsyncCOM.MessageClass, _
ByVal *msgID* As Integer, ByVal *msg* As String_
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

msgClass　メッセージの重大度を示します。次のいずれかの値を取ります。

- **MsgInfo**　同期の進行状況情報が含まれるメッセージです。

- **MsgDetailedInfo**　MsgInfo と似ていますが、より詳細な情報が含まれています。
- **MsgWarning**　問題が発生する可能性があることを示すメッセージですが、正常な同期の妨げにはならない問題です。
- **MsgError**　正常な同期の妨げになる問題を示すメッセージです。

msgID　メッセージのユニークな識別子です。msgID がゼロの場合、メッセージにはユニークな識別子はありません。

msg　メッセージのテキストです。

説明

このイベントを使用して、dbmlsync がログを取った情報を受け取ります。特定のメッセージが生成されたときに特別な処理を実行する場合は、MsgID でメッセージを確認します。このようにすると、メッセージのテキストが変更されても、コードはそのまま機能します。

例

次に示す Visual Basic .NET の例は、dbmlsync がログを取ったメッセージをリストボックス・コントロールに追加します。

```
Private Sub dbmlsync1_Message(
 ByVal msgClass As DbmlsyncCOM.MessageClass,
 ByVal msgId As Integer, ByVal msg As String
)
Handles dbmlsync1.Message

    Select Case msgClass
        Case DbmlsyncCOM.MessageClass.MsgError
            lstMessages.Items.Add("Error: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgWarning
            lstMessages.Items.Add("Warning: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgInfo
            lstMessages.Items.Add("Info: " + msg)
        Case DbmlsyncCOM.MessageClass.MsgDetailedInfo
            lstMessages.Items.Add("DetInfo: " + msg)
    End Select

End Sub
```

例

次に示す Visual Basic .NET の例は、エラーを処理するように Message イベントを設定します。エラー・メッセージは、lstMessages という ListBox コントロールに追加されます。

```
Private Sub dbmlsync1_Message(ByVal msgClass As DbmlsyncCOM.MessageClass, ByVal msgId As
Integer, ByVal msg As String) Handles dbmlsync1.Message
  If msgClass = DbmlsyncCOM.MessageClass.MsgError Then
    lstMessages.Items.Add("Error: " + msgId.ToString() + " " + msg)
  End If
End Sub
```

可能性のあるエラーの ID 値を表示するには、dbmlsync 統合コンポーネントのテストを実行します。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、Message イベントは次のエントリを lstMessages に挿入します。

```
Error: 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 dbmlsync 統合コンポーネントを設定します。Message イベントは、エラー 14173 が発生すると、restartSynchronization という変数を設定し、numberOfRestarts という変数をリセットします。EndSynchronization イベントは、最大 5 回、同期をリトライします。

```
' variables for restarting synchronization
Dim numberOfRestarts As Integer = 0
Dim restartSynchronization As Integer = 0



Private Sub dbmlsync1_Message
(
 ByVal msgClass As DbmlsyncCOM.MessageClass,
 ByVal msgId As Integer, ByVal msg As String ) Handles dbmlsync1.Message


 If msgClass = DbmlsyncCOM.MessageClass.MsgError Then
    lstMessages.Items.Add("Error: " + msgId.ToString() + " " + msg)


    If msgId = 14173 Then
      restartSynchronization = 1
      numberOfRestarts = 0

    End If
 End If
End Sub



Private Sub dbmlsync1_EndSynchronization(ByVal ExitCode As Integer, _
   ByRef restart As Boolean _
   ) Handles dbmlsync1.EndSynchronization



   If restartSynchronization = 1 Then
     If numberOfRestarts < 5 Then
       restart = True
       numberOfRestarts = numberOfRestarts + 1
     End If
   End If
End Sub
```

ProgressIndex イベント

ProgressIndex イベントは、dbmlsync が進行状況バーを更新したときにトリガされます。

構文

Public Event **ProgressIndex(_**
  ByVal *index* As Integer, _
  ByVal *max* As Integer _
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

インデックス　同期の進行状況を表す整数。

max　進行状況の最大値。完了した割合 (パーセント) は、index/max x 100 で計算できます。この値がゼロである場合、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。

説明

このイベントを使用して、進行状況バーのような、進行状況を示すインジケータを更新します。

例

次に示す Visual Basic .NET の例は、Index の値に基づいて、進行状況バー・コントロールを更新します。インデックスの最大値は、同期の開始時に設定されます。

```
Private Sub dbmlsync1_ProgressIndex(
 ByVal index As Integer,
 ByVal max As Integer
)
Handles dbmlsync1.ProgressIndex

   If max <> 0 Then
      ProgressBar1.Maximum = max
   End If
   ProgressBar1.Value = index

End Sub
```

ProgressMessage イベント

ProgressMessage イベントは、同期の進行状況情報が変更されたときにトリガされます。

構文

Public Event **ProgressMessage(** ByVal *msg* As String **)**
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

msg　新しい進行状況の文字列。

説明

このイベントを使用して、通常は dbmlsync の進行状況バーに表示される文字列を受け取ります。

例

次に示す Visual Basic .NET の例は、ProgressMessage イベントがトリガされたときに進行状況のラベルの値を設定します。

```
Private Sub dbmlsync1_ProgressMessage(
 ByVal msg As String
)
Handles dbmlsync1.ProgressMessage

   lblProgressMessage.Text = msg
```



```
End Sub
```

SetTitle イベント

SetTitle イベントは、ステータス情報が変更されたときにトリガされます。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。

構文

Public Event **SetTitle(** ByVal *title* **)** As String
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

title　dbmlsync ウィンドウのタイトル・バー内のタイトル。

説明

このイベントを使用して、dbmlsync ウィンドウの値が変更されたときに、このウィンドウに通常表示されるタイトルを受け取ります。

例

次に示す Visual Basic .NET の例は、SetTitle イベントがトリガされたときに Windows フォームのタイトルを設定します。

```
Private Sub dbmlsync1_SetTitle(
 ByVal title As String
)
Handles dbmlsync1.SetTitle

    Me.Text = title

End Sub
```

UploadAck イベント

UploadAck イベントは、コンポーネントがアップロードの確認を Mobile Link サーバから受信した後にトリガされます。

構文

Public Event **UploadAck(_**
 ByVal *status* As DbmlsyncCOM.UploadAckStatus _
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

status　アップロードが処理された後に、Mobile Link によってリモートに返されたステータスを示します。以下のいずれかの値になります。

- **StatCommitted**　Mobile Link サーバがアップロードを受信し、コミットしたことを示します。
- **StatRetry**　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 で異なっていたことを示します。Mobile Link サーバは、アップロードをコミットしませんでした。コンポーネントは、Mobile Link サーバのログ・オフセットから開始して別のアップロードを送信します。
- **StatFailed**　Mobile Link サーバがアップロードをコミットしなかったことを示します。

説明

このイベントを使用して、dbmlsync がアップロードの確認を Mobile Link サーバから受信した後にカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、UploadAck イベントがトリガされたときに、アップロードが失敗していた場合にメッセージを出力します。

```
Private Sub dbmlsync1_UploadAck(ByVal status As DbmlsyncCOM.UploadAckStatus) Handles
dbmlsync1.UploadAck

    If status = DbmlsyncCOM.UploadAckStatus.StatFailed Then
        MsgBox("Upload Failed")
    End If

End Sub
```

UploadRow イベント

UploadRow イベントは、Mobile Link サーバにローがアップロードされるときにトリガされます。

構文

Public Event **UploadRow(**
  ByVal *rowData* As DbmlsyncCOM.IRowTransferData
)
Member of **DbmlsyncCOM.Dbmlsync**

パラメータ

rowData　アップロードされるローの詳細な情報が含まれている IRowTransferData オブジェクト。

「IRowTransferData インタフェース」 311 ページを参照してください。

説明

このイベントを使用して、Mobile Link サーバにアップロードされるローを調べます。

UploadRow イベントを有効にするには、UploadEventsEnabled プロパティを使用します。
「UploadEventsEnabled プロパティ」 292 ページを参照してください。

例

次に示す Visual Basic .NET の例は、UploadRow イベントでローのすべてのカラムを反復処理します。この処理で、カラム値が null であるかどうかが判別され、カラム名と値が出力されます。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
   If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
      ' output the non-null column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + CStr(rowData.ColumnValue(liX)))
   Else
      ' output 'NULL' for the column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + "NULL")
   End If
Next liX

End Sub
```

WaitingForUploadAck イベント

WaitingForUploadAck イベントは、コンポーネントが Mobile Link サーバからのアップロード確認の待機を開始したときにトリガされます。

構文

Public Event **WaitingForUploadAck( )**
Member of **DbmlsyncCOM.Dbmlsync**

説明

このイベントを使用して、コンポーネントが Mobile Link サーバからのアップロード確認を待機しているときにカスタム・アクションを追加します。

例

次に示す Visual Basic .NET の例は、WaitingForUploadAck イベントがトリガされたときにメッセージを出力します。

```
Private Sub dbmlsync1_WaitingForUploadAck()
Handles dbmlsync1.WaitingForUploadAck

   MsgBox("Waiting for Upload Acknowledgement")

End Sub
```

IRowTransferData インタフェース

Public Interface **IRowTransferData**
Member of **DbmlsyncCOM**

UploadRow イベントと DownloadRow イベントは、パラメータとして DbmlsyncCOM.IRowTransferData オブジェクトを受け入れ、アップロードされたローとダウンロードされたローを調べます。このインタフェースは、テーブル名、ローの操作、カラム名などのローの詳細な情報を定義します。

RowOperation プロパティ

ローで実行される操作を指定します。

構文

Public Property **RowOperation( )** As DbmlsyncCOM.RowEventOp
Member of **DbmlsyncCOM.IRowTransferData**

説明

このプロパティの値は、次のいずれかです。

OpInsert　ローが挿入されました。

OpUpdate　ローが更新されました。

OpDelete　ローが削除されました。

OpTruncate　テーブルがトランケートされました (テーブルのすべてのローが削除されました)。RowOperation プロパティがこの値を保持しているときは、ColumnName プロパティと ColumnValue プロパティは無効な情報を返します。

注意： DownloadRow イベントの場合、アップサート (更新または挿入) 操作に値 OpInsert が指定されます。

TableName プロパティ

アップロード操作またはダウンロード操作が発生するテーブルの名前です。

構文

Public Property **TableName( )** As String
Member of **DbmlsyncCOM.IRowTransferData**

説明

TableName プロパティは、アップロード操作またはダウンロード操作が発生するテーブルの名前を指定します。次の例は、UploadRow イベントでの TableName プロパティの使い方を示しています。

「UploadRow イベント」 309 ページを参照してください。

例

Visual Basic .NET の例を示します。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

   MsgBox ("Table name:" + rowData.TableName)

End Sub
```

ColumnName プロパティ

アップロード操作またはダウンロード操作が発生するローのカラム名を取得します。

構文

Public Property **ColumnName(**ByVal *index* As Integer**)** As Object
Member of **DbmlsyncCOM.IRowTransferData**

パラメータ

index　取得するカラム名を指定するゼロベースの整数。インデックス値の範囲は、ゼロから、ColumnCount プロパティの値未満までです。

「ColumnCount プロパティ」 313 ページを参照してください。

説明

同じインデックスで ColumnValue プロパティを使用して、対応するカラム値を取得できます。

例

次に示す Visual Basic .NET の例は、UploadRow イベントでローのすべてのカラムを反復処理します。この処理で、カラム値が null であるかどうかが判別され、カラム名と値が出力されます。

「UploadRow イベント」 309 ページを参照してください。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
   If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
      ' output the non-null column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + CStr(rowData.ColumnValue(liX)))
   Else
      ' output 'NULL' for the column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + "NULL")
   End If
Next liX

End Sub
```

ColumnValue プロパティ

アップロード操作またはダウンロード操作が発生するカラムの値を取得します。

構文

Public Property **ColumnValue(** ByVal *index* As Integer **)** As Object
Member of **DbmlsyncCOM.IRowTransferData**

パラメータ

index　取得するカラム値を指定するゼロベースの整数。インデックス値の範囲は、ゼロから、ColumnCount プロパティの値未満までです。

「ColumnCount プロパティ」 313 ページを参照してください。

説明

更新操作が発生した場合、このプロパティで指定されるカラム値は更新が適用された後の値です。

同じインデックスで ColumnName プロパティを使用して、対応するカラム名を取得できます。

BLOB のカラム値を、このプロパティを通じて使用することはできません。BLOB のカラムが検出された場合、ColumnValue は文字列 "(blob)" です。

例

次に示す Visual Basic .NET の例は、UploadRow イベントでローのすべてのカラムを反復処理します。この処理で、カラム値が null であるかどうかが判別され、カラム名と値が出力されます。

「UploadRow イベント」 309 ページを参照してください。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

Dim liX As Integer
For liX = 0 To rowData.ColumnCount - 1
   If VarType(rowData.ColumnValue(liX)) <> VariantType.Null Then
      ' output the non-null column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + CStr(rowData.ColumnValue(liX)))
   Else
      ' output 'NULL' for the column value
      MsgBox("Column " + CStr(liX) + ": " + rowData.ColumnName(liX) + _
      ", " + "NULL")
   End If
Next liX

End Sub
```

ColumnCount プロパティ

アップロード操作またはダウンロード操作が発生するローに含まれるカラム数です。

構文

Public Property **ColumnCount( )** As Integer
Member of **DbmlsyncCOM.IRowTransferData**

説明

ColumnCount プロパティは、アップロード操作またはダウンロード操作が発生するローのカラム数を指定します。次の例は、UploadRow イベントでの ColumnCount プロパティの使い方を示しています。

「UploadRow イベント」 309 ページを参照してください。

例

Visual Basic .NET の例を示します。

```
Private Sub dbmlsync1_UploadRow(
 ByVal rowData As DbmlsyncCOM.IRowTransferData
)
Handles dbmlsync1.UploadRow

    MsgBox "Number of Columns:" + CStr(rowData.ColumnCount)

End Sub
```

第 12 章

dbmlsync の DBTools インタフェース

目次

dbmlsync の DBTools インタフェースの概要

データベース・ツール (DBTools) は、同期を含むデータベース管理をアプリケーションに統合するために使用できるライブラリです。データベース管理ユーティリティはすべて、DBTools によって構築されます。

「データベース・ツール・インタフェース」 『SQL Anywhere サーバ - プログラミング』を参照してください。

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、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_sync_db 構造体」 『SQL Anywhere サーバ - プログラミング』を参照してください。

- **a_syncpub 構造体** この構造体は、パブリケーション情報を保持します。同期用パブリケーションのリンク・リストを指定できます。

  「a_syncpub 構造体」 『SQL Anywhere サーバ - プログラミング』を参照してください。

- **DBSynchronizeLog 関数** この関数は同期処理を開始します。この関数のパラメータは、a_sync_db インスタンスへのポインタだけです。

  「DBSynchronizeLog 関数」 『SQL Anywhere サーバ - プログラミング』を参照してください。

dbmlsync 統合コンポーネント

dbmlsync 用の DBTools インタフェースを使用する代わりに、dbmlsync 統合コンポーネントを使用することもできます。

「dbmlsync 統合コンポーネント」 287 ページを参照してください。

dbmlsync の DBTools インタフェースの設定

この項では、dbmlsync の DBTools インタフェースの基本的な使用手順を示します。

DBTools ライブラリの詳細については、「データベース・ツール・インタフェースの概要」『SQL Anywhere サーバ - プログラミング』を参照してください。

ご使用の開発環境でのインポート・ライブラリの使用についての詳細は、「データベース・ツール・インタフェースの使い方」『SQL Anywhere サーバ - プログラミング』を参照してください。

♦ C や C++ で作成された DBTtools インタフェースを使用して dbmlsync の設定と起動を行うには、次の手順に従います。

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");
     }
     ```

     DBTools の初期化の詳細については、次の項を参照してください。

     - 「データベース・ツール・インタフェースの使い方」『SQL Anywhere サーバ - プログラミング』
     - 「a_dbtools_info 構造体」『SQL Anywhere サーバ - プログラミング』
     - 「DBToolsInit 関数」『SQL Anywhere サーバ - プログラミング』

3. a_sync_db 構造体を初期化します。

   - a_sync_db インスタンスを宣言します。たとえば、次のように dbsync_info というインスタンスを宣言します。

```
a_sync_db dbsync_info;
```

- a_sync_db 構造体のフィールドをクリアします。

  ```
  memset( &dbsync_info, 0, sizeof( dbsync_info ) );
  ```

- 必須の a_sync_db のフィールドを設定します。

  ```
  dbsync_info.version = DB_TOOLS_VERSION_NUMBER;
  dbsync_info.output_to_mobile_link = 1;
  dbsync_info.default_window_title
    = "dbmlsync dbtools sample";
  ```

- データベース接続文字列を設定します。

  ```
  dbsync_info.connectparms = "uid=DBA;pwd=sql";
  ```

  データベース接続パラメータの詳細については、「-c オプション」 130 ページを参照してください。

- 同期をカスタマイズするための他の a_sync_db のフィールドを設定します。

  ほとんどのフィールドは、dbmlsync コマンド・ライン・オプションに対応しています。この対応の詳細については、*dbtools.h* を参照してください。

  次の例では、冗長オペレーションが指定されています。

  ```
  dbsync_info.verbose_upload = 1;
  dbsync_info.verbose_option_info = 1;
  dbsync_info.verbose_row_data = 1;
  dbsync_info.verbose_row_cnts = 1;
  ```

a_sync_db のフィールドの詳細については、「a_sync_db 構造体」 『SQL Anywhere サーバ - プログラミング』を参照してください。

4. 同期中にフィードバックを受け取るコールバック関数を作成し、これらの関数を適切な a_sync_db のフィールドに割り当てます。

   次の関数は、標準出力ストリームを使用して dbmlsync のエラー、ログ、進行状況情報を表示します。

   DBTools のコールバック関数の詳細については、「コールバック関数の使い方」 『SQL Anywhere サーバ - プログラミング』を参照してください。

   - たとえば、生成されたエラー・メッセージを処理する dbsyncErrorCallBack という関数を作成します。

     ```
     extern short _callback dbsyncErrorCallBack( char *str )
     {
        if( str != NULL ) {
           printf( "Error Msg    %s¥n", str );
        }
        return 0;
     }
     ```

   - たとえば、生成された警告メッセージを処理する dbsyncWarningCallBack という関数を作成します。


```
extern short _callback dbsyncWarningCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Warning Msg  %s¥n", str );
    }
    return 0;
}
```

- たとえば、冗長情報メッセージを受け取る dbsyncLogCallBack という関数を作成します。この情報メッセージは、ウィンドウに表示する代わりにファイルに記録できます。

```
extern short _callback dbsyncLogCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Log Msg      %s¥n", str );
    }
    return 0;
}
```

- たとえば、同期中に生成された情報メッセージを受け取る dbsyncMsgCallBack という関数を作成します。

```
extern short _callback dbsyncMsgCallBack( char *str )
{
    if( str != NULL ) {
        printf( "Display Msg  %s¥n", str );
    }
    return 0;
}
```

- たとえば、進行状況テキストを受け取る dbsyncProgressMessageCallBack という関数を作成します。dbmlsync ユーティリティでは、このテキストは進行状況バーの真上に表示されます。

```
extern short _callback dbsyncProgressMessageCallBack(
 char *str )
{
    if( str != NULL ) {
        printf( "ProgressText %s¥n", str );
    }
    return 0;
}
```

- たとえば、進行状況インジケータ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。
  - **max**　進行状況の最大値。この値がゼロである場合、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。ほとんどの場合はこの値が返されるようにします。
  - **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. どのパブリケー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 -n 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、同期するパブリケー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 パブリケーションと template_p2 パブリケーションをまとめて識別するには、次のように指定します。

     ```
     publication_info.next = NULL; // linked list terminates
     publication_info.pub_name = "template_p1,template_p2";
     publication_info.ext_opt  = "sv=template_ver1";
     publication_info.alloced_by_dbsync = 0;
     ```

     これは、dbmlsync コマンド・ラインで -n 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指定された関連スクリプト・バージョンは、dbmlsync -eu オプションと同じ機能を提供します。

     「-eu オプション」 140 ページを参照してください。

   - 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 オプションの 1 つの定義に相当します。

   「-n オプション」 146 ページと「a_syncpub 構造体」『SQL Anywhere サーバ - プログラミング』を参照してください。

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 関数は、DBTools リソースを開放します。

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

   「DBToolsFini 関数」『SQL Anywhere サーバ - プログラミング』を参照してください。

第 13 章

スクリプト化されたアップロード

目次

スクリプト化されたアップロードの概要

スクリプト化されたアップグレードは、SQL Anywhere リモート・データベースを使用する Mobile Link アプリケーションだけに適用されます。

警告
スクリプト化されたアップロードを実装した場合、dbmlsync では、アップロードするデータの判別にトランザクション・ログは使用されません。その結果、スクリプトがすべての変更を取得しなかった場合は、リモート・データベース上のデータが失われることがあります。このため、ほとんどのアプリケーションの同期方法としては、ログベースの同期をおすすめします。

ほとんどの Mobile Link アプリケーションでは、データベースのトランザクション・ログによってアップロードが判別されるので、最後のアップロード以降にリモート・データベースに加えられた変更が同期されます。これは、ほとんどのアプリケーションでは適切な設計であり、リモート上のデータが失われないようにしています。

しかし、まれに、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、この章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、通常はリモート・データベース上のテーブルをロックする必要があります。ログベースの同期を行う場合は、ロックする必要はありません。
- スクリプト化されたアップロードを使用してトランザクション単位のアップロードを実装するのは非常に困難です。

スクリプト化されたアップロードの設定

次の手順では、Mobile Link 同期が設定済みであることを前提に、スクリプト化されたアップロードの設定に必要なタスクの概要について説明します。

♦ スクリプト化されたアップロードの設定の概要

1. アップロードするローを識別するストアド・プロシージャを作成します。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。

   「スクリプト化されたアップロードのストアド・プロシージャの定義」 334 ページを参照してください。

2. WITH SCRIPTED UPLOAD というキーワードが含まれ、ストアド・プロシージャの名前を指定するパブリケーションを作成します。

   「スクリプト化されたアップロードのパブリケーションの作成」 338 ページを参照してください。

スクリプト化されたアップロードを使用するときは、dbmlsync の LockTables 拡張オプションにデフォルト設定を使用することをおすすめします。
スクリプト化されたアップロードの多くの問題は、LockTables にデフォルト設定を使用することで防ぐことができます。この設定により、dbmlsync は、すべての同期テーブルのロックを取得してから、アップロードを構築できます。これにより、スクリプトがアップロードを構築中に、他の接続が同期テーブルを変更するのを防ぐことができます。また、この設定を行うと、スクリプトがアップロードを構築中に、同期テーブルに影響するコミットされていないトランザクションをなくすことができます。

クイック・スタートのためのその他の資料

- 「スクリプト化されたアップロードの例」 339 ページ

スクリプト化されたアップロードの設計上の考慮事項

1 ローあたり 1 つの操作

アップロードには、1 つのローに対して複数の操作 (挿入、更新、または削除) を含めることはできません。ただし、複数の操作を 1 つのアップロード操作にまとめることはできます。たとえば、ローを挿入してから更新する場合は、2 つの操作を、最後の値を 1 回挿入する操作に置き換えることができます。

操作の順序

アップロードを統合データベースに適用すると、挿入と更新操作の後に削除操作が適用されます。特定のテーブル内での操作順序について他の想定をすることはできません。

競合の解決

同期と同期の間に複数のデータベースでローが更新されると、競合が発生します。アップロード内の各更新操作には、更新中のローの更新前イメージが含まれているので、Mobile Link サーバ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。

競合解決の実装の例については、「スクリプト化されたアップロードの例」 339 ページを参照してください。

競合を処理しない

競合の検出を処理する必要がない場合は、更新前イメージを追跡しなければ、アプリケーションを大幅に簡素化できます。代わりに、挿入操作として更新をアップロードします。次に、ローが存在しない場合は挿入し、存在する場合はローを更新する upload_insert スクリプトを統合データベースに作成します。SQL Anywhere 統合データベースを使用している場合、この操作は、upload_insert スクリプトの INSERT 文にある ON EXISTING 句を使用して実行できます。

「INSERT 文」 『SQL Anywhere サーバ - SQL リファレンス』を参照してください。

競合を処理せず、複数のリモート・データベースで同じローが変更されると、最後に同期したリモート・データベースによって、それまでの変更が上書きされます。

強制的な競合解決の処理

削除操作では、アップロードされたローのプライマリ・キーが正しいことが重要です。しかし、ほとんどの場合、非プライマリ・キー・カラムの値が、統合データベースの値と一致しているかどうかは問題ではありません。非プライマリ・キー・カラムが重要なのは、Mobile Link サーバ

で強制的な競合モードが使用されるときだけです。この場合、カラムのすべての値が、統合データベースの upload_old_row_insert スクリプトに渡されます。このスクリプトを実装した方法によっては、非プライマリ・キー・カラムを正しい値にする必要があります。

「強制的な競合解決」 『Mobile Link - サーバ管理』を参照してください。

ロッキング

スクリプト化されたアップロードの多くの問題は、dbmlsync 拡張オプション LockTables にデフォルト設定を使用することで防ぐことができます。この設定により、dbmlsync は、すべての同期テーブルの排他ロックを取得してから、アップロードを構築できます。これにより、スクリプトがアップロードを構築中に、他の接続が同期テーブルを変更するのを防ぐことができます。また、この設定を行うと、スクリプトがアップロードを構築中に、同期テーブルに影響するコミットされていないトランザクションをなくすことができます。

テーブルのロックをオフにする必要がある場合は、「テーブルをロックしない場合のスクリプト化されたアップロード」 330 ページを参照してください。

冗長なアップロード

多くの場合、リモート・データベースで各操作をアップロードするのは、一度だけです。この操作を支援するため、Mobile Link は各サブスクリプションの進行状況値を維持します。進行状況値は、デフォルトで、dbmlsync が正常な最終アップロードを構築し始めた時間です。この進行状況値は、sp_hook_dbmlsync_set_upload_end_progress フックを使用して異なる値で上書きできます。

「sp_hook_dbmlsync_set_upload_end_progress」 275 ページを参照してください。

アップロード手順の 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 フックを使用して、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だけを通知できるストアド･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、常に機能するとはかぎりません。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て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。アップロードのプロシージャで構築するアップロードに、sp_hook_dbmlsync_set_upload_end_progress hook フックの呼び出し前にリモート・データベースでコミットされた変更内容がすべて含まれる場合は、dbmlsync によってデータ損失が防止されます。

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。

時間	
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。

dbmlsync は、複数のメカニズムを使用して、sp_hook_dbmlsync_set_upload_end_progress フックの呼び出し時にコミットされていなかった変更または sp_hook_dbmlsync_set_upload_end_progress フックの呼び出し後にコミットされた変更がダウンロード内容で上書きされないようになっています。

フックの呼び出し前にコミットされた変更は保護されないので、ダウンロードの適用時に上書きが可能です。ただし、ダウンロード構築前に送信されるアップロードに変更内容が含まれれば、変更内容は Mobile Link サーバに送信されるので、サーバ側のスクリプトで、統合データベース内のデータとの間で競合を解決してから、ダウンロードを構築できます。

テーブルをロックしない場合のスクリプト化されたアップロード

デフォルトでは、dbmlsync によって同期対象のテーブルがロックされてからアップロード・スクリプトが呼び出され、このロックはダウンロードがコミットされるまで保持されます。拡張オプション LockTables をオフに設定すると、テーブルのロックを無効にできます。

可能な場合は、テーブルをロックするデフォルトの動作を使用することをおすすめします。テーブルをロックしないでスクリプト化されたアップロードを行うと、考慮する必要がある問題が増え、実行可能な正しい解決法を作成するのが難しくなります。これは、データベースの同時実行性と同期の概念をよく理解している上級ユーザ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。ロー R は、今後も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。ロー R’ が統合データベースから間違って削除される可能性があります。

参照

- 「sa_transactions システム・プロシージャ」『SQL Anywhere サーバ - SQL リファレンス』
- 「独立性レベルの設定」『SQL Anywhere サーバ - SQL の使用法』

スクリプト化されたアップロードのストアド・プロシージャの定義

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は、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。

次の名前と値の組み合わせが定義されています。

名前	値	説明
start progress	タイムスタンプ文字列	リモート・データベースに対するすべての変更がアップロードされるまでの時間。アップロードが反映されるのは、この時間以降に発生した操作だけです。
raw start progress	64 ビット符号なし整数	符号なし整数で示された開始進行状況
end progress	タイムスタンプ文字列	アップロード期間の終了。アップロードが反映されるのは、この時間の前に発生した操作だけです。
raw end progress	64 ビット符号なし整数	符号なし整数で示された終了進行状況
generating download exclusion list	true|false	同期がダウンロード専用またはファイルベースの場合は true。この場合、アップロードは送信されず、ダウンロードがスクリプト化されたアップロードのストアド・プロシージャ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)。
publication_*n*	パブリケーション名	同期されているパブリケーション (*n* は整数)。*n* の番号はゼロから始まります。
script version	バージョン名	同期に使用される Mobile Link スクリプト・バージョン
MobiLink user	Mobile Link ユーザ名	同期対象となる Mobile Link ユーザ

「#hook_dict テーブル」 221 ページを参照してください。

スクリプト化されたアップロードの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、終了進行状況値をタイムスタンプまたは符号なし整数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。

次の項を参照してください。

- 「sp_hook_dbmlsync_set_upload_end_progress」 275 ページ
- 「sa_convert_ml_progress_to_timestamp システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』
- 「sa_convert_timestamp_to_ml_progress システム・プロシージャ」 『SQL Anywhere サーバ - SQL リファレンス』

挿入用ストアド・プロシージャの定義

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・プロシージャの定義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の例」 339 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアド・プロシージャ 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
  )
)
```

削除用ストアド・プロシージャの定義

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･プロシージャの定義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の例」 339 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアド･プロシージャ 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 WITH SCRIPTED UPLOAD p1(
  // Order of columns here is ignored
  TABLE t1( c3, pk, c1 ) USING (
    PROCEDURE t1_delete FOR UPLOAD DELETE
  )
)
```

更新用ストアド・プロシージャの定義

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)。

つまり、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

更新用のストアド･プロシージャの定義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の例」 339 ページを参照してください。

次の例では、t1 というテーブルと p1 というパブリケーションを作成します。パブリケーションは、WITH SCRIPTED UPLOAD を指定し、ストアド･プロシージャ 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
    )
)
```

スクリプト化されたアップロードのパブリケーションの作成

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
    )
)
```

参照

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

スクリプト化されたアップロードの例

この例は、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。例では、スクリプト化されたアップロードで必要な統合データベースとリモート・データベース、ストアド・プロシージャ、パブリケーション、サブスクリプションを作成します。この例は、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。

統合データベースの作成

サンプル・ファイルを保持するディレクトリを作成します。ここでは、名前を scripteduploadとします。コマンド・プロンプトを開き、そのディレクトリに移動します。

(この例では、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。)

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 consol.db
```

次のコマンドを実行して、統合データベースの ODBC データ・ソースを定義します。

```
dbdsn -w dsn_consol -y -c "uid=DBA;pwd=sql;dbf=consol.db;eng=consol"
```

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、統合データベースとして consol.db を設定します。

```
dbisql -c "dsn=dsn_consol" %sqlany10%¥MobiLink¥setup¥syncsa.sql
```

Interactive SQL を開き、dsn_consol DSN を使用して consol.db に接続します。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 ( ?, ?, ? )' );

CALL ml_add_table_script( 'default', 'employee', 'upload_update',
     'UPDATE employee SET name = ?, salary = ? WHERE id = ?' );

CALL ml_add_table_script( 'default', 'employee', 'upload_delete',
     'DELETE FROM employee WHERE id = ?' );

CALL ml_add_table_script( 'default', 'employee', 'download_cursor',
     'SELECT * from employee' );
```

リモート・データベースの作成

サンプル・ディレクトリのコマンド・プロンプトで、次のコマンドを実行して、リモート・データベースを作成します。

```
dbinit remote.db
```

次のコマンドを実行して、ODBC データ・ソースを定義します。

```
dbdsn -w dsn_remote -y -c "uid=dba;pwd=sql;dbf=remote.db;eng=remote"
```

Interactive SQL で、dsn_remote DSN を使用して remote.db に接続します。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'
);
```

次に、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。

挿入の処理

まず、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。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 1900-01-0 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。

```
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;
```

次に、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。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。

```
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;
```

更新の処理

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。

まず、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 )
);
```

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。

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、タイムスタンプが同じ場合を除く)。これは一見、無駄に見えるので、ローの更新前イメージがテーブルにない場合のみ保存し、sp_hook_dbmlsync_upload_end フックに依存して、更新前イメージがアップロードされた後に削除しがちです。

しかし、sp_hook_dbmlsync_upload_end フックは、この目的では確実ではありません。アップロードを送信した後で受信確認される前に、ハードウェアまたはソフトウェアの障害により dbmlsync が停止した場合、フックが呼び出されない可能性があります。その結果、ローが正常にアップロードされても、ローは更新前イメージ テーブルから削除されません。また、通信障害が発生すると、dbmlsync はサーバからアップロードの受信確認を受けれ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・テーブルがクリア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。

```
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;
```

次に、更新を処理する更新プロシージャを作成します。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。

```
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';

   // 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;
```

削除の処理

まず、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
);
```

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。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。

```
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;
```

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。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。

```
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;
```

更新前イメージのクリーンアップ

次に、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。この例では、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。

```
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;
```

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

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';
```

スクリプト化されたアップロードの実行

リモート・データベースに接続し、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。たとえば、Interactive SQL のリモート・データベースに対して次の 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;
```

コマンド・プロンプトで、Mobile Link サーバを起動します。

```
mlsrv10 -c "dsn=dsn_consol" -o mlserver.mls -v+ -dl -zu+
```

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+
```

これで、挿入がアップロードされたことを確認できます。

例のクリーンアップ

例を完了後コンピュータをクリーンアップするには、次の手順を実行します。

```
mlstop -h -w
dbstop -y -c eng=consol
dbstop -y -c eng=remote

dberase -y consol.db
dberase -y remote.db

del remote.mlc mlserver.mls
```

パート III. Ultra Light クライアント

パート III では、Mobile Link 同期のために Ultra Light クライアントを設定し実行する方法について説明します。

第 14 章

Ultra Light クライアント

目次

Ultra Light クライアントの概要

Ultra Light のランタイムには、現場や移動中の社員と企業のバックエンド・システムを結ぶ双方向の同期フレームワークが組み込まれているので、Ultra Light データの同期は、他のリモート・クライアントほど複雑ではありません。組み込みのフレームワークによって、Ultra Light データベース内のデータはすべてデフォルトで自動的に同期されます。Mobile Link 同期を初めて行うときには、このデフォルトの動作を使用できます。業務上、必要になれば、統合データベースと同期する Ultra Light データを変更するように同期をカスタマイズできます。

Ultra Light の詳細については、「Ultra Light の概要」 『Ultra Light - データベース管理とリファレンス』を参照してください。SQL Anywhere データベースを Mobile Link クライアントとして使用する方法については、「SQL Anywhere クライアント」 83 ページを参照してください。

ヒント
Ultra Light アプリケーションをバックアップする最適な方法として、統合データベースとの同期を行うことが挙げられます。Ultra Light データベースをリストアするには、空のデータベースを作成し、統合データベースと同期してデータを移植します。

ヒント
配備するファイルが複数ある場合や、配備するファイルのバージョンがユーザ ID ごとに異なる場合は、Mobile Link サーバを使用してファイルを転送できます。「Mobile Link ファイル転送の使用」 367 ページを参照してください。

Ultra Light の組み込みの同期機能

Ultra Light では、データ管理レイヤに Mobile Link 同期テクノロジが含まれます。したがって、SQL Anywhere リモートのように同期機能を追加するために Ultra Light の専有容量を増やす必要がありません。

Ultra Light のランタイムに組み込まれている重要な同期機能には、ローのステータスを追跡するメカニズムと、進行状況のカウンタがあります。

ローのステータスを追跡するメカニズム

テーブルとローのステータスの追跡は、データ同期時に特に重要です。Ultra Light データベースの各ローには、ローのステータスを記録する 1 バイトのマーカがあります。Ultra Light では、ローのステータスは、同期だけでなく、トランザクション処理とデータ・リカバリの制御にも使用されます。「Ultra Light でのローのステータス」 『Ultra Light - データベース管理とリファレンス』を参照してください。

進行状況のカウンタ

Ultra Light は、進行状況カウンタを使用して堅牢な同期を行います。各アップロードには、識別のためのユニークな ID が付けられます。この ID により、アップロードが正常に行われたかどうか、通信エラーが発生していないかどうかを判別できます。

新しいデータベースを作成したときは、同期進行状況カウンタは必ず 0 に設定されます。進行状況のカウンタの値 0 は、データベースが新しい Ultra Light データベースなので、Mobile Link サーバでこのクライアントのステータス情報をリセットする必要があることを示します。

警告
Ultra Light は、同期が発生すると進行状況カウンタを 1 つずつ増やすので、Ultra Light データベースを別の統合データベースに同期することはできません。進行状況カウンタの値が 0 ではなく、統合データベースに格納されているシーケンス番号と一致しない場合は、Mobile Link 同期はオフセット不一致をレポートし、同期は失敗します。

参照

- 「ml_subscription」 『Mobile Link - サーバ管理』

Ultra Light クライアントの同期の動作のカスタマイズ

Ultra Light にカスタムの同期のサポートを追加するには、次の 3 つの作業を行います。

- **複数のリモート・クライアントを含む同期モデル内でプライマリ・キーの一意性を維持する。** 必須。同期システムでは、プライマリ・キーは、異なるデータベース (リモートと統合) 内の同じローを識別する唯一の方法であり、競合を検出する唯一の方法です。したがって、複数のクライアントがある場合は、以下の規則に従う必要があります。
  - 同期される各テーブルには、プライマリ・キーが存在する必要がある。
  - プライマリ・キーの値は更新しない。
  - プライマリ・キーは、同期されるすべてのデータベース間でユニークでなければならない。

  「ユニークなプライマリ・キーの管理」 『Mobile Link - サーバ管理』と「Ultra Light でのプライマリ・キーの一意性」 353 ページを参照してください。
- **小数データが失われないようにデータ・カラムを設定する。** SQL Anywhere 統合データベースの場合は、これは問題ではありません。ただし、Oracle などのデータベースでは、互換性の問題を考慮する必要がある場合があります。たとえば、Ultra Light と Oracle のデータベースでは、タイムスタンプ精度が同じである必要があります。また、Oracle データベースに TIMESTAMP を追加して、Ultra Light リモート・データベースから統合データベースにデータをアップロードするときに秒の小数点以下のデータが失われないようにする必要があります。「Oracle 統合データベース」 『Mobile Link - サーバ管理』と「Ultra Light precision プロパティ」 『Ultra Light - データベース管理とリファレンス』を参照してください。
- **統合データベースにアップロードするデータ・サブセットを指定する。** オプション。デフォルトですべてのデータを同期しない場合にのみ行います。同期するデータを指定するには、1 つまたは複数のサブセット方法を使用します。「Ultra Light での同期の設計」 358 ページを参照してください。

  たとえば、優先度の高いデータのパブリケーションを作成することができます。ユーザは高速無線ネットワークを経由してこのデータを同期できます。無線ネットワークには使用コストが

かかるので、このコストは、業務に必要なものだけに制限する必要があります。緊急でないデータは後でクレードルから同期できます。

- **Ultra Light アプリケーションから同期を初期化し、セッションに関するパラメータを指定する。** 必須。同期のプログラミング時には、セッションの設定を行ってから、同期操作を初期化します。

  セッションの設定を行うには、同期の通信ストリーム (ネットワーク・プロトコル) とそのストリームのパラメータを選択し、同期スクリプトのバージョンを設定し、Mobile Link ユーザを割り当てます。ただし、ほかにも設定できるパラメータがあります。たとえば、upload_only パラメータと download_only パラメータを使用して同期の方向をデフォルトの双方向から一方向に変更できます。「Ultra Light アプリケーションへの同期の追加」 362 ページを参照してください。

その他の重要な同期の動作は、Mobile Link 同期スクリプトで制御します。各 Mobile Link リモート・データベースは、異なるデータのサブセットを統合データベース内に持つことができるので、複数のスクリプトが必要です。

これらのコードを以下に示します。

- Ultra Light リモート・データベース内のテーブルに更新としてダウンロードするデータ
- リモート・データベースからアップロードされた変更内容に必要な処理

つまり、自分専用の同期スクリプトを作成して、適切な方法でリモート・データベース間でデータを「分割」できます。

参照

- 「Mobile Link 統合データベース」 『Mobile Link - サーバ管理』
- 「同期スクリプトの概要」 『Mobile Link - サーバ管理』
- 「ダイレクト・ロー・ハンドリング」 『Mobile Link - サーバ管理』
- 「リモート・データベース間でローを分割する」 『Mobile Link - サーバ管理』

Ultra Light でのプライマリ・キーの一意性

Ultra Light では、Mobile Link でサポートされている任意の方法でプライマリ・キーの一意性を維持できます。「ユニークなプライマリ・キーの管理」 『Mobile Link - サーバ管理』を参照してください。

方法の一つとして、グローバル ID を使用できます。グローバル ID は、GLOBAL AUTOINCREMENT カラムの値が生成される方法に影響します。Mobile Link サーバで複数のクライアントの同期を管理する必要がある場合は、GLOBAL AUTOINCREMENT と宣言したカラムを使用する必要があります。

GLOBAL AUTOINCREMENT は AUTOINCREMENT と同じですが、ドメインが分割されます。Ultra Light では、データベースのグローバル ID に割り当てられた分割からのみカラムの値が設定されます。

参照

- 「Ultra Light でのグローバル・データベース ID の考慮事項」 『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light global_database_id オプション」 『Ultra Light - データベース管理とリファレンス』

Ultra Light での GLOBAL AUTOINCREMENT の使用

Ultra Light データベースにあるカラムのデフォルト値は、GLOBAL AUTOINCREMENT 型として宣言できます。ただし、これらのカラム ID を自動的に増分するには、その前に Ultra Light リモートのグローバル ID を設定する必要があります。

警告
Mobile Link 同期を介してダウンロードされた GLOBAL AUTOINCREMENT カラムの値は、GLOBAL AUTOINCREMENT 値のカウンタを更新しません。したがって、ある Mobile Link クライアントによって別のクライアントの分割に値が挿入された場合は、エラーが発生する可能性があります。この問題を回避するため、Ultra Light アプリケーションのコピーでは、そのコピー自体の分割にだけ値が挿入されるようにします。

♦ Ultra Light データベースで GLOBAL AUTOINCREMENT カラムを宣言するには、次の手順に従います。

1. データベースの各コピーにユニークなグローバル ID 番号を割り当てます。

   global_database_id データベース・オプションは、Ultra Light データベースに値を設定します。Ultra Light を配備するときには、各データベースに対して必ず異なる ID 番号を割り当てる必要があります。「Ultra Light global_database_id オプション」 『Ultra Light - データベース管理とリファレンス』を参照してください。

2. Ultra Light は、Ultra Light のデータベース番号でユニークに識別された分割を使用して、カラムにデフォルト値を設定します。Ultra Light は、次の規則に従います。

- カラムに現在の分割の値が含まれていない場合、最初のデフォルト値は *pn* + 1 である。ここで、*p* は分割サイズ、*n* はグローバル ID 番号を表します。
- カラムに現在の分割の値が含まれていても、そのすべてが *p*(*n* + 1) 未満であれば、この範囲内でこれまで使用した最大値より 1 大きい値が次のデフォルト値になる。
- デフォルトのカラム値は、現在の分割以外のカラムの値の影響を受けない。つまり、*pn* + 1 より小さいか *p*(*n* + 1) より大きい数には影響されない。Mobile Link 同期を介して別のデータベースからレプリケートされた場合に、このような値が存在する可能性があります。

  たとえば、Ultra Light データベースにグローバル ID 番号 1 を割り当て、分割サイズが 1000 の場合、データベースのデフォルト値は 1001 ー 2000 の範囲から選択されます。このデータベースの別のコピーで、ID 番号 2 が割り当てられたデータベースからは、2001 ー 3000 の範囲で同一カラムのデフォルト値が指定されます。
- グローバル ID 番号は負の値に設定できないので、Ultra Light で GLOBAL AUTOINCREMENT カラムに選択される値は常に正の数です。ID 番号の最大値を制限するのは、カラムのデータ型と分割サイズだけです。
- グローバル ID の値を設定しないか、分割内の値がなくなった場合は、カラムに NULL 値が挿入されます。NULL 値が許可されていない場合にローを挿入しようとすると、エラーが発生します。

3. GLOBAL AUTOINCREMENT と宣言されたカラムに使用可能な値がなくなったか、なくなりそうになったら、新しいグローバル ID を設定する必要があります。GLOBAL AUTOINCREMENT の値は、グローバル ID 番号によって識別される分割から、最大値に達するまで選択されます。値を超えると、NULL 値が生成され始めます。新しいグローバル ID 番号を割り当てることで、Ultra Light で別の分割から適切な値を設定できるようになります。

   新しいグローバル ID を選択する方法の一つとして、未使用のグローバル ID の値のプールを管理できます。このプールは、プライマリ・キー・プールと同じ方法で管理されます。「プライマリ・キー・プールの使用」『SQL Remote』を参照してください。

ヒント
Ultra Light の API を使うと、使用済みの番号の割合を取得できます。戻り値は、これまでに使用された値のパーセンテージを表す 0 ー 100 の範囲の SHORT 型です。たとえば、値 99 は、未使用の値がほとんど残っていないので、データベースに新しい ID 番号を割り当てる必要があることを示します。この ID 番号の設定方法は、使用するプログラミング・インタフェースによって異なります。

参照

- 「オートインクリメント・カラムの分割サイズの上書き」 356 ページ
- 「Ultra Light でのグローバル・データベース ID の考慮事項」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for AppForge：「プロパティ」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「Connection プロパティ」『Ultra Light - .NET プログラミング』

- Ultra Light for C/C++：「Synchronize 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「setDatabaseID メソッド」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULSetDatabaseID 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「プロパティ」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「GlobalAutoIncrementUsage プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「GetSynchResult 関数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「getGlobalAutoIncrementUsage メソッド」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULGlobalAutoincUsage 関数」『Ultra Light - C/C++ プログラミング』

最後に割り当てられた GLOBAL AUTOINCREMENT の値の割り出し

最後の挿入操作中に選択された GLOBAL AUTOINCREMENT の値は、取り出すことができます。これらの値はプライマリ・キーで頻繁に使用されるので、生成された値がわかると、最初のローのプライマリ・キーを参照するローをより簡単に挿入できます。次のいずれかを使用して値を確認できます。

- **Ultra Light for C/C++** ULConnection オブジェクトの GetLastIdentity 関数を使用します。「GetLastIdentity 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
- **Ultra Light for Mobile VB** ULConnection オブジェクトの GetLastIdentity 関数を使用します。
- **Ultra Light.NET** ULConnection クラスの LastIdentity プロパティを使用します。「LastIdentity プロパティ」『Ultra Light - .NET プログラミング』を参照してください。
- **Ultra Light for M-Business Anywhere** Connection クラスの GetLastIdentity メソッドを使用します。「getLastIdentity メソッド」『Ultra Light - M-Business Anywhere プログラミング』を参照してください。

戻り値は、符号なし 64 ビット整数であるデータベース・データ型 UNSIGNED BIGINT です。この文では最後に割り当てられたデフォルト値がわかるだけなので、間違った結果を取らないために INSERT 文を実行した直後にこの値を取り出してください。

注意
ときには、1 つの INSERT 文に GLOBAL AUTOINCREMENT 型のカラムが複数含まれていることがあります。この場合、戻り値は生成されたデフォルト値のいずれか 1 つですが、そのうちのどの値であるかを判別する信頼できる方法はありません。このため、このような状況を回避するようなデータベースの設計と INSERT 文の記述を行ってください。

オートインクリメント・カラムの分割サイズの上書き

この分割サイズには任意の正の整数を設定できますが、通常、分割サイズは、サイズの値がすべての分割で不足しないように選択されます。

カラムの型が INT または UNSIGNED INT である場合、デフォルトの分割サイズは $2^{16} = 65536$ です。他の型のカラムの場合、デフォルトの分割サイズは $2^{32} = 4294967296$ です。これらのデフォルト値は適切でないことがあるため、分割サイズを明示的に指定するのが最も賢明です。

Ultra Light アプリケーションと SQL Anywhere データベースでは、一部のデータ型のデフォルト分割サイズが異なります。他のデータベースの一貫性を保つ場合は、分割サイズを明示的に宣言します。

♦ Ultra Light 分割値を上書きするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light データベースに接続します。
2. 選択したカラムを右クリックし、ポップアップ・メニューから [プロパティ] を選択します。

   [カラム] プロパティ・シートが表示されます。
3. [値] タブをクリックします。
4. [分割サイズ] フィールドには任意の正の整数を入力します。

♦ Ultra Light でオートインクリメント・カラムを宣言するには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. AUTOINCREMENT キーワードの直後に分割サイズをカッコで指定した DEFAULT GLOBAL AUTOINCREMENT フレーズを使用して、CREATE TABLE または ALTER TABLE 文を実行します。「Ultra Light CREATE TABLE 文」『Ultra Light - データベース管理とリファレンス』と「Ultra Light ALTER TABLE 文」『Ultra Light - データベース管理とリファレンス』を参照してください。

   たとえば、次の文では 2 つのカラム (顧客 ID 番号を保持する整数カラム、顧客名を保持する文字列カラム) を持つ簡単なリファレンス・テーブルが作成されます。このテーブルの分割サイズには、5000 を設定する必要があります。

```
CREATE TABLE customer (
  id   INT         DEFAULT GLOBAL AUTOINCREMENT (5000),
  name VARCHAR(128) NOT NULL,
  PRIMARY KEY (id)
)
```

参照

- Ultra Light for AppForge：「プロパティ」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「GetColumnPartitionSize メソッド」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「GetGlobalAutoincPartitionSize 関数」『Ultra Light - C/C++ プログラミング』

- Ultra Light for M-Business Anywhere：「getColumnPartitionSize メソッド」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「ULGlobalAutoincUsage 関数」『Ultra Light - C/C++ プログラミング』

Ultra Light での同期の設計

Ultra Light データベース内のすべてのデータは、デフォルトで同期されます。Mobile Link リモート・データベースとして Ultra Light を初めて配備する場合、最初は Ultra Light リモート全体を同期することを検討します。

処理に慣れてきたら、同期操作の動作をカスタマイズして、より複雑なビジネス論理を展開できます。カスタム同期動作を設計するときは、次の点を確認する必要があります。業務上の要件が単純である場合は、単一の同期機能だけを使用できます。ただし、非常に複雑な配備では、複数の同期機能を使用すると、必要な同期動作をより詳細に制御できます。

設計時の確認事項	「はい」の場合の処理
同期からテーブルを除外するかどうか。	テーブル名サフィックス nosync を使用して、同期しないテーブルを指定できます。「Ultra Light の nosync テーブル」 359 ページを参照してください。
データが変更されてないときも含め、常にテーブル全体を同期するかどうか。	テーブル名サフィックス allsync を使用すると、変更が検出されなくてもテーブル全体を同期できます。「Ultra Light の allsync テーブル」 359 ページを参照してください。
テーブル全体を同期するか、特定の条件を満たすローだけを同期するか。重要度または緊急度により、同期の優先度が高いデータがあるかどうか。	パブリケーションには、同期が必要なテーブルをリストするアーティクルが含まれます。アーティクルには、ローが定義した基準を満たすかどうかに基づいてアップロードするローを指定する WHERE 句を含めることができます。 複数のパブリケーションを使用すると、特定の Ultra Light データを他のデータよりも前にアップロードする必要がある場合の優先度の問題に対処できます。「Ultra Light のパブリケーション」 360 ページを参照してください。
外部キーの循環があるので、同期するテーブルの順序を指定する必要があるかどうか。	Table Order 同期パラメータを使用すると、外部キーの循環があるときの同期操作の順序を指定できます。ただし、外部キーの循環は一般に Ultra Light にはおすすめしません。「Ultra Light のテーブルの順序」 361 ページを参照してください。
同期の動作を制御するかどうか。たとえば、アップロードと同時にダウンロードを行うかどうか。また、同期の方向を双方向から一方向に変更するかどうか。	次の場所で適切な同期パラメータを使用します。 ◆ アプリケーションの同期構造体 (または同期列挙) ◆ ulsync ユーティリティの -e オプション 「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページを参照してください。

設計時の確認事項	「はい」の場合の処理
同期は、時間ベース (スケジュールが組まれている) で行うのか、クレードルでトリガするのか、ユーザが開始するか。または、これらを組み合わせて同期するか。	プログラムで適切なインタフェースを使用して異なる動作を実行できます。場合によっては、HotSync または ActiveSync で同期処理を管理できます。「Ultra Light アプリケーションへの同期の追加」 362 ページを参照してください。
Ultra Light クライアントで TLS を有効にするかどうか。	選択した暗号化アルゴリズムにより、デバイスで実行するプラットフォームに従ってデバイスを設定する方法が決まります。「XX」を参照してください。

参照

- 「同期処理」 『Mobile Link - クイック・スタート』
- 「アップロードとダウンロード」 『Mobile Link - クイック・スタート』

Ultra Light の nosync テーブル

テーブル名に **_nosync** サフィックスを追加することで、テーブル全体をアップロード操作から除外することを指定できます。これらの非同期テーブルは、統合データベースでは不要なクライアント固有の永続的なデータ用に使用できます。同期からは除外されていますが、テーブルの使用方法自体は Ultra Light データベース内の他のテーブルとまったく同じです。

nosync サフィックスを含むテーブルを作成し、名前を変更する場合は、nosync サフィックスが保持されるようにする必要があります。たとえば、次の RENAME 句を指定した ALTER DBSPACE 文は、変更した名前の末尾が nosync ではなくなるので使用できません。

```
ALTER TABLE purchase_comments_nosync
RENAME comments
```

修正するには、次のようにこのサフィックスを含む文に書き換えます。

```
ALTER TABLE purchase_comments_nosync
RENAME comments_nosync
```

またはパブリケーションを使用して同じ効果を上げることができます。「Ultra Light のパブリケーション」 360 ページを参照してください。

Ultra Light の allsync テーブル

テーブル名に **_allsync** サフィックスを追加することで、アップロード時に、前回の同期セッション後に変更されたデータがない場合にも、すべてのテーブル・データを同期するように同期の動作を変更することを指定できます。

一部の Ultra Light アプリケーションでは、allsync テーブルに保管できるユーザ固有またはクライアント固有のデータが必要です。したがって、統合データベースのテンポラリ・テーブルにこのテーブルのデータをアップロードすると、他のスクリプトで同期を制御するのにそのデータを使用できます。そのため、統合データベースにそのデータを保持する必要はありません。たとえ

ば、Ultra Light のアプリケーションを使用して複数のチャネルやトピックの中から関連のあるものを示し、この情報を使って適切なローをダウンロードする場合などが該当します。

Ultra Light のパブリケーション

パブリケーションを使用して、同期するデータを指定するアーティクルを定義します。通常は、各アーティクルは 1 つのテーブルの全体を構成します。または、テーブル内にデータのサブセットを定義できます。特定のテーブルのローのサブセットを定義する場合は、オプションで述部 (WHERE 句) を含めることができます。

パブリケーションは、nosync テーブル・サフィックスの方法よりも柔軟なので、より詳細な制御に使用できます。Ultra Light データベースのデータ・サブセットを「*別々*」に同期させるには、複数のパブリケーションを使用します。パブリケーションをアップロード専用またはダウンロード専用の同期パラメータと組み合わせると、優先度の高い変更を効率よく同期することが可能です。

パブリケーションの追加

パブリケーションは、Sybase Central を使用して Ultra Light データベースに追加したり、Interactive SQL から追加したりできます。Ultra Light 同期では、パブリケーション内の各アーティクルに、完全なテーブルか WHERE 句のいずれかを入れることができます (Palm OS の HotSync を除く)。

注意
Ultra Light パブリケーションでは、カラムのサブセットの定義と、SUBSCRIBE BY 句がサポートされていません。Ultra Light テーブル内のカラムが SQL Anywhere 統合データベース内のテーブルと完全に一致しない場合は、Mobile Link スクリプトを使用してその違いを解決します。
パブリケーションでテーブル同期順序を設定する必要はありません。配備においてテーブル順序が重要な場合は、Ultra Light データベースを同期するときに Table Order 同期パラメータを設定して、テーブル順序を設定できます。

♦ Ultra Light のデータベースからデータをパブリッシュするには、次の手順に従います (Sybase Central の場合)。

1. Ultra Light プラグインを使用して、Ultra Light データベースに接続します。
2. [パブリケーション] フォルダを開き、右ペイン上で右クリックし、[新規] - [パブリケーション] をクリックします。
3. 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。
4. [テーブル] タブで、[使用可能なテーブル] のリストからテーブルを 1 つ選択します。[追加] をクリックします。選択したテーブルが、右側の [選択したテーブル] リストに表示されます。
5. テーブルを追加します。

6. 必要な場合は、[WHERE] タブをクリックし、パブリケーションに含めるローを指定します。カラムのサブセットは指定できません。HotSync 同期を使用している場合は、WHERE 句を指定しないでください。
7. [終了] をクリックします。

♦ Ultra Light のデータベースからデータをパブリッシュするには、次の手順に従います (Interactive SQL の場合)。

1. Ultra Light データベースに接続します。
2. CREATE PUBLICATION 文を実行して、新しく作成するパブリケーションの名前とパブリッシュするテーブルの名前を指定します。

参照

- 「Table Order 同期パラメータ」 402 ページ
- 「Download Only 同期パラメータ」 386 ページ
- 「Upload Only 同期パラメータ」 404 ページ
- 「データのパブリッシュ」 89 ページ
- 「Ultra Light CREATE PUBLICATION 文」 『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light のパブリケーションの操作」 『Ultra Light - データベース管理とリファレンス』
- 「同期スクリプトの概要」 『Mobile Link - サーバ管理』

Ultra Light のテーブルの順序

Table Order 同期パラメータを設定することで、同期操作の順序を制御できます。同期のテーブル順序を指定する場合は、Table Order パラメータをプログラムで使用するか、テスト中に ulsync ユーティリティの一部として使用します。Table Order パラメータは、アップロードするテーブルの順序を指定します。「Table Order 同期パラメータ」 402 ページを参照してください。

テーブル順序は、Ultra Light データベースに次のいずれかがある場合にのみ明示的に設定します。

- 外部キー循環。循環に含まれるすべてのテーブルをリストする必要があります。
- 統合データベースとは異なる外部キー関係

外部キー循環に関連する同期の問題の防止

テーブル順序は、外部キー循環を使用する Ultra Light データベースに特に重要です。循環が形成されるようにテーブルをリンクさせると、循環が発生します。ただし、統合データベースと Ultra Light リモート間の循環が異なる場合は外部キー循環は複雑になるのでおすすめしません。

外部キー循環がある場合は、プライマリ・テーブルへの操作が先に実行され、関連する外部テーブルへの操作が後に実行される順序にテーブルを並べてください。Table Order パラメータによって、外部テーブルへの挿入が外部キーの参照整合性制約を満たしていることが保証されます (削除などの他の操作でも同様です)。

テーブル順序とは別に、同期の問題を防ぐ方法として、参照整合性を確認してから操作をコミットできます。統合データベースが SQL Anywhere データベースの場合は、1 つの外部キーを **check on commit** (コミット時にチェック) に設定します。このように設定すると、外部キーの参照整合性が、操作の開始時ではなくコミット・フェーズでチェックされます。次に例を示します。

```
create table c (
    id integer not null primary key,
    c_pk integer not null
);
create table p (
    pk integer not null primary key,
    c_id integer not null,
    foreign key p_to_c (c_id) references c(id)
);
alter table c
   add foreign key c_to_p (c_pk)
   references p(pk)
   check on commit;
```

他のデータベース・ベンダーの統合データベースを使用している場合は、参照総合性のチェックが同様の方法で行われているかどうかを確認します。同じであれば、この方法を実装します。方法が異なる場合は、すべての外部キー循環をなくすようにテーブル関係を再設計してください。

参照

- 「参照整合性と同期」 『Mobile Link - クイック・スタート』

Ultra Light アプリケーションへの同期の追加

Ultra Light では、設定されている通信ストリーム (ネットワーク・プロトコル) で Mobile Link サーバとの特定の接続を開始することで同期が開始されます。ネットワークに直接接続している場合の同期のサポートに加えて、Palm OS デバイスでは HotSync 同期が、Windows CE デバイスでは ActiveSync 同期がサポートされます。

接続の定義

各 Ultra Light リモートは、ネットワーク・プロトコルによって Mobile Link サーバと同期します。ネットワーク・プロトコルは同期ストリーム・パラメータを使用して設定します。サポートされているネットワーク・プロトコルには、TCP/IP、HTTP、HTTPS、TLS があります。選択するプロトコルについて、Mobile Link サーバのホストやポートなどその他の必要な接続情報を定義するストリーム・パラメータを指定する必要があります。

また、user_name パラメータと version パラメータで Mobile Link のユーザ情報と同期スクリプトのバージョンを指定する必要もあります。

同期動作の定義

同期動作は、さまざまな同期パラメータを設定することで制御できます。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。

検討する必要がある重要な動作は、次のとおりです。

- **同期の方向**　デフォルトでは、同期は双方向です。一方向だけの同期を行うには、upload_only パラメータまたは download_only パラメータを使用します。一方向の同期を実行することによって、同期に必要な時間を最小限に抑えることができます。また、ダウンロード専用同期では、Ultra Light データベースに対してあらかじめすべての変更をコミットする必要はありません。コミットしなかった変更は同期に含まれず、アップロードされません。したがって、不完全なトランザクションでも問題は起こりません。

  ダウンロード専用同期を使用する場合は、ダウンロードと重なっているローがローカル・レベルで変更されていないことを確認してください。ローカル・レベルでデータが変更されていると、SQLE_DOWNLOAD_CONFLICT エラーによって、Ultra Light アプリケーションの同期は失敗します。

- **同期中の変更**　アップロード・フェーズでは、Ultra Light アプリケーションは、読み込み専用として Ultra Light データベースにアクセスできます。ダウンロード・フェーズでは、読み込み／書き込みアクセスが許可されますが、アプリケーションがローを変更してからダウンロードによってこのローが変更されようとすると、ダウンロードが失敗し、ロールバックが行われます。disable_concurrency 同期パラメータを設定することで、同期中のデータへの同時アクセスを無効にすることができます。

- **Ultra Light アプリケーションに同期コードを追加するには、次の手順に従います。**

1. 同期情報の構造体のフィールドとして、セッションに必要な同期パラメータとプロトコル・オプションを指定します。

   たとえば、C/C++ API を使用する場合は、ul_synch_info 構造体に適切な値を設定して、Ultra Light アプリケーションに同期を追加します。

   ```
   ul_synch_info info;
       // define a sync structure named "info"
        ULEnableTcpipSynchronization( &sqlca );
       // use a TCP/IP stream
       conn->InitSynchInfo( &info );
       // initialize the structure
       info.stream = ULSocketStream();
       // specify the Socket Stream
       info.stream_parms= UL_TEXT( "host=myMLserver;port=2439"  );
       // set the MobiLink host information
       info.version = UL_TEXT( "custdb 10.0" );
       // set the MobiLink version information
       info.user_name = UL_TEXT( "50" );
       // set the MobiLink user name
       info.download_only =ul_true;
       // make the synchronization download-only
   ```

2. 同期を初期化します。

   TCP/IP を使用して直接同期する場合は、API 固有の同期関数を呼び出します。これらの関数は、同期操作の成功または失敗を示すブール値を返します。同期が失敗した場合は、別の構造体で詳細なエラー・ステータスのフィールドを検証して、追加のエラー情報を取得できます。

   HotSync 同期の場合は、引数として ul_synch_info 構造体を指定して ULSetSynchInfo 関数を呼び出す必要があります。ul_synch_info 構造体を指定すると、以後の同期で使用できるように、この構造体が現在のデータベースにアタッチされます。

ActiveSync 同期の場合は、ActiveSync プロバイダからの同期メッセージを取得し、DoSync 関数を使用して ULSynchronize を呼び出す必要があります。

3. 同期の進行状況をユーザにレポートする場合は、observer コールバック関数を使用します。

ヒント
DLL が非常に大きいか、ネットワーク接続の信頼性が低いため、DLL を正常に実行できない場合は、再開可能なダウンロードを実装できます。「失敗したダウンロードの処理」『Mobile Link - サーバ管理』と「失敗したダウンロードの再開」『Mobile Link - サーバ管理』を参照してください。

参照

- 「Ultra Light クライアント用 ActiveSync と HotSync の配備」 369 ページ
- 「アップロード専用の同期とダウンロード専用の同期」『Mobile Link - サーバ管理』
- 「アップロードとダウンロード」『Mobile Link - クイック・スタート』
- 「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページ
- Ultra Light for AppForge：「データの同期」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「Ultra Light アプリケーションの同期」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「データの同期」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「Palm アプリケーションに HotSync 同期を追加する」『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「アプリケーションへの ActiveSync 同期の追加」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データの同期」『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：「アプリケーションへの同期の追加」『Ultra Light - C/C++ プログラミング』

Ultra Light での TLS が有効化された同期の設定

Mobile Link の Ultra Light クライアント・アプリケーションは、TLS が有効になった同期を使用するように設定する必要があります。トランスポート・レイヤ・セキュリティにより、暗号化、改ざん検出、証明書ベースの認証が実現します。「トランスポート・レイヤ・セキュリティの概要」『SQL Anywhere サーバ - データベース管理』を参照してください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」『SQL Anywhere 10 - 紹介』を参照してください。

プラットフォームのサポート
RSA、ECC、FIPS の各暗号化は、すべてのプラットフォームで使用できるわけではありません。各プラットフォームでサポートされている暗号化方法については、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」に掲載されている Ultra Light の表を参照してください。

♦ Ultra Light クライアント・アプリケーションおよびデバイスで TLS 同期を設定するには、次の手順に従います。

1. 次のいずれかを呼び出して、暗号化された同期を有効にします。
   - RSA 暗号化を有効にするには、ULEnableRsaSyncEncryption を呼び出します。「ULEnableRsaSyncEncryption 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
   - ECC 暗号化を有効にするには、ULEnableEccSyncEncryption を呼び出します。「ULEnableEccSyncEncryption 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
   - FIPS RSA 暗号化を有効にするには、ULEnableRsaFipsSyncEncryption を呼び出します。この機能が必要なのは、Palm OS クライアントだけです。「ULEnableRsaFipsSyncEncryption 関数」『Ultra Light - C/C++ プログラミング』を参照してください。
2. 同期の情報ストリームを TLS または HTTPS に設定します。
3. ECC または FIPS 暗号化を有効にしている場合は、以下の設定を行う必要もあります。
   - **ECC** **tls_type** ネットワーク・プロトコル・オプションを **ECC** に設定します。「tls_type」 66 ページを参照してください。
   - **FIPS** **fips** ネットワーク・プロトコル・オプションを **Yes** に設定します。「fips」 51 ページを参照してください。
4. 適切なライブラリにリンクしていることを確認します。

プラットフォーム	リンク	RSA 暗号化	ECC 暗号化	FIPS 暗号化
Windows デスクトップ	静的 [1]	*ulrsa.lib*	*ulecc.lib*	なし
Windows デスクトップ	動的 [2]	なし	なし	なし
Windows CE	静的 [1]	*ulrsa.lib*	*ulecc.lib*	なし
Windows CE	動的 [2]	なし	なし	なし
Palm OS	静的 [1]	*ulrsa.lib*	*ulecc.lib*	*ulfips.lib*, *gse1st.lib*

1 ulrt.lib にもリンクする必要があります。

2 ulimp.lib にもリンクする必要があります。

5. 適切なファイルがデバイスにコピーされていることを確認します。

プラットフォーム	リンク	RSA 暗号化	ECC 暗号化	FIPS 暗号化
Windows デスクトップ	静的	なし	なし	*mlcrsafips10.dll* *sbgse2.dll*
Windows デスクトップ	動的 [1]	*mlcrsa10.dll*	*mlcecc10.dll*	*mlcrsafips10.dll* *sbgse2.dll*
Windows CE	静的	なし	なし	*mlcrsafips10.dll* *sbgse2.dll*
Windows CE	動的 [1]	*mlcrsa10.dll*	*mlcecc10.dll*	*mlcrsafips10.dll* *sbgse2.dll*
Palm OS	静的	なし	なし	*libsbgse_4i.prc*
Windows CE コンポーネントおよび Ultra Light エンジン	静的 [2]	*mlcrsa10.dll*	*mlcecc10.dll*	*mlcrsafips10.dll* *sbgse2.dll*

1 ulrt10.dll を配備する必要もあります。

2 コンポーネント .DLL ファイルと uleng10.exe、またはそのいずれかを配備する必要もあります。

参照

- 「トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定」『SQL Anywhere サーバ - データベース管理』
- Ultra Light for AppForge：「暗号化と難読化」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「暗号化と難読化」『Ultra Light - .NET プログラミング』
- Ultra Light for C++：「データの暗号化」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データベースの暗号化と難読化」『Ultra Light - M-Business Anywhere プログラミング』

Mobile Link ファイル転送の使用

M-Business Anywhere を除くすべての Ultra Light ライブラリで、MobiLink サーバからのファイル転送がサポートされています。M-Business Anywhere には、ファイルの配備または転送の独自のメカニズム (チャネル同期) があるので、この機能は必要ありません。

その他のすべての API については、次の場合に Mobile Link のファイル転送メカニズムを使用します。

- 特にセキュリティ対策に企業のファイアウォールを使用していて、複数のファイルを複数のデバイスに配備する必要がある場合。Mobile Link は、企業のファイアウォールを介して同期を処理するように設定済みなので、MLFileTransfer メカニズムにより、アップグレードや他のタイプのファイル転送のためのデバイスのプロビジョニングが非常に便利になります。
- 特定の Mobile Link ユーザ ID を対象とするファイルがある場合。必要なユーザ ID ごとに、Mobile Link サーバで 1 つまたは複数のユーザ固有のディレクトリを作成する必要があります。ファイルのバージョンが 1 つだけの場合は、デフォルトのディレクトリを使用できます。

ファイル転送のしくみ

Mobile Link から開始する 2 種類のファイル転送方法のどちらかを使用して、ファイルをデバイスにダウンロードできます。デスクトップ転送用の mlfiletransfer ユーティリティを実行する方法か、使用する API に適切な関数を呼び出して Ultra Light アプリケーションをコーディングする方法です。どちらの方法でも次の処理が必要です。

1. 転送先を指定します。

   デスクトップから mlfiletransfer ユーティリティを使用するか、API に適切な関数を使用するかに関係なく、転送先のデバイスまたはデスクトップ・コンピュータにあるファイルのローカル・パスとファイル名を設定する必要があります。アプリケーションで、またはエンド・ユーザがこの情報を指定しなかった場合、ファイルは転送元のファイル名で現在のディレクトリに保存されます。

   ターゲットの転送先ディレクトリは、次のようにデバイスのオペレーティング・システムによって異なります。

   - Palm OS では、転送先が外部記憶メディアの場合は、ローカル・パスの前に **vfs:** を付ける必要があります。

     転送先が NULL の場合、mlfiletransfer では、ダウンロードする必要があるファイルが、デバイスのレコード・ストアへの Palm のレコード・データベース (**.pdb* ファイル) であると見なされます。

     ファイル名は、Symbian OS のファイルの命名規則に従っている必要があります。「Palm OS」『Ultra Light - データベース管理とリファレンス』を参照してください。
   - Windows CE では、転送先が NULL の場合、ファイルはルート・ディレクトリ (¥) に格納されます。

ファイル名は、Windows CE のファイルの命名規則に従っている必要があります。「Windows CE」 『Ultra Light - データベース管理とリファレンス』を参照してください。

- デスクトップ・コンピュータでは、転送先が NULL の場合、ファイルは現在のディレクトリに格納されます。

  ファイル名は、デスクトップ・システムのファイルの命名規則に従っている必要があります。「Windows デスクトップ」 『Ultra Light - データベース管理とリファレンス』を参照してください。

2. ユーザが識別され、正しいファイルがダウンロードされるように、ユーザ認証を設定します。

   このユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するために使用されます。

3. 使用するストリーム・タイプを設定し、必要なストリームのパラメータを定義します。このパラメータは、Ultra Light で Mobile Link 同期用にサポートされているパラメータと同じです。「Ultra Light 同期パラメータとネットワーク・プロトコル・オプション」 381 ページを参照してください。

   ほとんどの同期ストリームでは、Mobile Link サーバのアドレスを識別したり、その他の動作を制御したりするパラメータが必要です。ストリーム・タイプを、プラットフォームに適さない無効な値に設定すると、ストリーム・タイプは TCP/IP に設定されます。

4. 転送メカニズムに必要な動作を指定します。

   たとえば、転送先にファイルが既に存在し、変更されていない場合でも、ダウンロードを強制したり、部分的なダウンロードを再開したりするようにプロパティを設定できます。また、ダウンロードの進行状況をモニタし、レポートするかどうかも設定できます。

5. Mobile Link サーバが実行中であり、-ftr オプションを指定して起動されたことを確認します。

6. 転送を開始し、必要な場合はダウンロードの進行状況をモニタします。

   ダウンロードの進行状況を表示することで、ユーザはダウンロードをキャンセルし、後で再開できます。

参照

- 「-ftr オプション」 『Mobile Link - サーバ管理』
- 「Mobile Link ファイル転送ユーティリティ [mlfiletransfer]」 32 ページ
- Ultra Light for C/C++：「MLFileTransfer 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULFileTransfer クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「ULFileTransfer クラス」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：サポートされていません。

第 15 章

Ultra Light クライアント用 ActiveSync と HotSync の配備

目次

Palm OS の HotSync

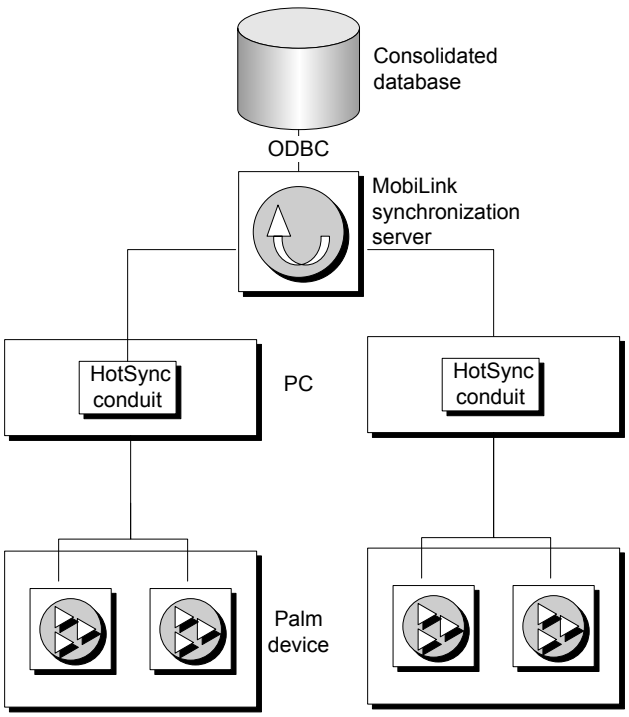
Palm OS デバイスのデータは、Ethernet、Wi-Fi、RAS 接続を介して同期することもできますが、この項では HotSync 同期用にデスクトップとデバイスを設定する方法について説明します。

HotSync 同期では、Palm デバイス上のすべての Ultra Light データベースをユーザのデスクトップから同時に管理できます。HotSync 同期は、デバイスをデスクトップに接続したときに HotSync によって外部的に開始されます。これを行うには、アプリケーションを HotSync 同期用に設定しておく必要があります。「Palm アプリケーションに HotSync 同期を追加する」 『Ultra Light - C/C++ プログラミング』を参照してください。

HotSync 同期を使用せずに Ultra Light をアプリケーションから直接同期する場合、エンド・ユーザは各アプリケーションを個別に開いて、各データベースを順番に同期する必要があります。これをプログラムで処理するよう実装するには、API 固有の同期機能を使用して同期を初期化します。「Palm アプリケーションに TCP/IP、HTTP、または HTTPS 通信による同期を追加する」『Ultra Light - C/C++ プログラミング』を参照してください。

HotSync アーキテクチャ

HotSync アーキテクチャには、リモート・データベースに送受信されるデータを同期した統合データベースが必要です。Mobile Link サーバは、これらのデータベース間の同期イベントを管理します。Ultra Light HotSync コンジットは、エンド・ユーザのデスクトップで同期イベントをローカルに管理します。



HotSync コンジット・インストール・ユーティリティ (ulcond10) を使用して、コンジットをインストールし、各 Ultra Light データベースを登録してください。コンジットを使用して各データベースを登録する場合、Ultra Light HotSync コンジットは次のような設定になります。

- アプリケーションに関連付けられたすべてのデータベースの HotSync 同期と登録された作成者 ID を管理する。該当する作成者 ID ごとにコンジットをインストールしてください。
- 適切な接続文字列を使用して各データベースに接続する。複数のデータベースの登録が必要な場合は、ulcond10 ユーティリティで -a オプションを使用します。このオプションでは、各接続文字列を -c オプションで定義した接続文字列に追加します。

これにより、デスクトップから HotSync で同期を初期化することで、すべてのデータベースを同時に同期できます。

参照

- 「Palm OS 用 Ultra Light HotSync コンジットのインストール・ユーティリティ (ulcond10)」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for AppForge：「データの同期」『Ultra Light - AppForge プログラミング』
- Ultra Light for C/C++：「Palm OS の Ultra Light アプリケーションの開発」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「アプリケーションへの同期の追加」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データの同期」『Ultra Light - M-Business Anywhere プログラミング』

HotSync 同期の概要

1. ul_sync_info 構造体を使用して、必要な同期パラメータをアプリケーションで指定したことを確認してください。SetSynchInfo 関数を呼び出すと、指定した同期パラメータは HotSync 同期用に設定されます。同期パラメータの一部には、Mobile Link サーバとの通信に必要なネットワーク・プロトコル・オプションも含まれています。「Mobile Link 同期のプロトコル・オプションの設定」 374 ページを参照してください。
2. Ultra Light アプリケーションを閉じると、Ultra Light アプリケーションの状態は、データベースとは別のテンポラリファイルに保存されます。「Ultra Light Palm アプリケーションのステータスの管理」『Ultra Light - C/C++ プログラミング』を参照してください。
3. Palm デバイスの同期時に、HotSync は特定の作成者 ID について、デスクトップの Ultra Light HotSync コンジットを呼び出します。
4. Ultra Light HotSync コンジットは、適切な接続文字列を使用して登録されたすべてのデータベースに接続し、それらを同期します。SetSynchInfo を適切に呼び出さないデータベースでは、同期が失敗します。
5. アプリケーションの起動時に、直前に保存された Ultra Light アプリケーションの状態がロードされます。「Ultra Light Palm アプリケーションの状態のリストア」『Ultra Light - C/C++ プログラミング』を参照してください。

参照

- 「ul_synch_info_a 構造体」『Ultra Light - C/C++ プログラミング』
- 「SetSynchInfo 関数」『Ultra Light - C/C++ プログラミング』

エンド・ユーザのデスクトップへの Ultra Light HotSync コンジットの配備

Ultra Light は、開発の段階で SQL Anywhere インストーラを使用して、デスクトップにインストールすることになります。ただし、その後に必要な HotSync コンジット・ファイルをエンド・ユーザのコンピュータに配備する必要があります。

Ultra Light の HotSync コンジット・ファイル

- *install-path¥win32¥condmgr¥condmgr.dll*– HotSync のインストール・パスを特定し、HotSync でコンジットを登録するユーティリティの DLL
- *install-path¥win32¥ulcond10.exe*– デスクトップ・コンピュータへの Ultra Light HotSync コンジットのインストールとアンインストールを行う Ultra Light HotSync コンジット・インストール・ユーティリティ。「Palm OS 用 Ultra Light HotSync コンジットのインストール・ユーティリティ (ulcond10)」『Ultra Light - データベース管理とリファレンス』を参照してください。
- *install-path¥win32¥dbhsync10.dll*– HotSync によって呼び出されるコンジットの DLL
- *install-path¥win32¥dblgen10.dll*– 言語リソース・ライブラリ。英語以外の言語では、ファイル名の *en* という文字が、言語を示す 2 文字の省略形に置換され、*dblgde10.dll* (ドイツ語) や *dblgja10.dll* (日本語) になります。
- **ストリーム用の DLL**　省略可。Ultra Light HotSync コンジットと Mobile Link サーバ間の暗号化されたネットワーク通信に必要なストリーム用の DLL。
  - TLS と HTTPS を使用する RSA 暗号化の場合は、*install-path¥win32¥mlcrsa10.dll*。
  - TLS と HTTPS を使用する ECC 暗号化の場合は、*install-path¥win32¥mlcecc10.dll*。
  - TLS と HTTPS を使用する RSA FIPS 暗号化の場合は、*install-path¥win32¥mlcrsafips10.dll*。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化されたストリーム用 DLL には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
別途ライセンスが必要なコンポーネントの注文については、http://www.iAnywhere.jp/sas/price.html を参照してください。
コンポーネントとプラットフォームのサポートについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある SQL Anywhere、Ultra Light、Mobile Link の「別途ライセンスが必要なコンポーネント」の項を参照してください。
サポートされるコンポーネントとプラットフォームの最新情報については、http://www.ianywhere.com/products/components_platforms_1001.html を参照してください。

♦ Ultra Light HotSync コンジットを配備して登録するには、次の手順に従います。

1. エンド・ユーザのデスクトップで、次のディレクトリを作成します。

- *MyDir¥win32¥*
- *MyDir¥win32¥condmgr*

2. 次のファイルのコピーを *MyDir¥win32¥* ディレクトリに配備します。
   - *ulcond10.exe*
   - *dbhsync10.dll*
   - *dblgen10.dll*
3. *Condmgr.dll* ファイルのコピーを *MyDir¥win32¥condmgr* ディレクトリに配備します。
4. 次のレジストリ・キーを見つけます。

   HKEY_CURRENT_USER¥Software¥Sybase¥SQL Anywhere¥10.0¥
5. レジストリ・キーに **Location** という名前の値を作成し、コンジットの配備用ルート・フォルダの名前をデータに指定します。たとえば、*MyDir* のように指定します。
6. エンド・ユーザに通信ストリームを暗号化するための証明書が必要な場合は、ルート証明書をデスクトップ・コンピュータにインストールしてコンジットがアクセスできるようにします。
7. ulcond10 を実行して、-c オプション (場合によっては、-a オプション) を使用して各 Ultra Light データベースの接続文字列を設定していることを確認してください。また、正しい作成者 ID が設定されていることも確認してください。

   これにより、Ultra Light HotSync コンジットが配備され、正しく設定されます。

   ヒント
   暗号化キーを使用している場合は、接続文字列にはキーを設定しないでください。キーを設定すると、セキュリティ上のリスクが発生する場合があります。その代わりに、コンジットでユーザにキーの入力を要求するプロンプトを表示するようにします。

   たとえば、次のコマンドは、作成者 ID が Syb2 であるアプリケーションの CustDB というコンジットをインストールします。

   ulcond10 -c "DBF=custdb.udb;UID=DBA;PWD=sql" -n CustDB Syb2
8. Ultra Light アプリケーションの ul_sync_info 構造体に同期パラメータを指定しなかった場合は、HotSync または ulcond10 で設定してください。「Mobile Link 同期のプロトコル・オプションの設定」 374 ページを参照してください。

♦ HotSync コンジットが正しく配備されたことをチェックするには、次の方法に従います。

· 次のようにして、コンジットが配備されていることを確認します。

a. コンピュータのシステム・トレイで、[HotSync Manager] を右クリックします。

b. ポップアップ・メニューで [動作設定] を選択します。

HotSync ユーザ別のコンジット・リストが表示されます。コンジットがリストに表示されていることを確認します。コンジットが表示されている場合は、Mobile Link サーバを起動して、同期操作をテストできます。

HotSync 操作のデバッグ

HotSync ログ・ファイルは、*Palm-install¥User-dir* ディレクトリに保存されます。デフォルトでは、このファイルには次の情報が含まれます。

- 同期イベントの実行時刻
- 登録された各コンジットのステータス

UL_DEBUG_CONDUIT_LOG 環境変数を設定すると、詳細なデバッグ情報をファイルに取得することができます。情報量のレベルは 2 つあり、取得する必要がある情報量に応じて選択できます。

UL_DEBUG_CONDUIT_LOG = 1　基本的な情報を記録します。たとえば、同期パラメータやレジストリの場所などです。

UL_DEBUG_CONDUIT_LOG = 2　その他の Ultra Light の詳細情報 (入出力操作を含む) を記録します。

HotSync を再起動して、新しい設定を有効にします。

参照

- 「SQL Anywhere 環境変数の概要」『SQL Anywhere サーバ - データベース管理』

Mobile Link 同期のプロトコル・オプションの設定

プロトコル・オプションは、Ultra Light HotSync コンジットから Mobile Link サーバへの接続を定義します。通常は、この情報をアプリケーションの同期コードの一部として追加します。ただし、必要なパラメータは HotSync や ulcond10 から入力することもできます。

ul_sync_info 構造体を使用している場合、引数は次の形式になります。

```
stream=stream_name;parameters
```

stream_name は、ネットワーク・プロトコルのタイプを示します。*stream_name* のデフォルト値は TCP/IP です。

parameters は、コンジットが使用するネットワーク・プロトコル・オプションのセットで、使用するプロトコルの **stream_parms** 引数と同じ形式です。指定されたストリームに対して、*parameters* は、プロトコルの **stream_parms** 引数と同じデフォルトを使用します。

注意
プロトコル・オプションを指定しない場合、コンジットは、ユーザがレジストリで ulcond10 の実行時に入力した可能性のあるプロトコル名とプロトコル・オプションを探します。
有効なネットワーク・プロトコルが見つからないと、デフォルトのプロトコルとプロトコル・オプションが使用されます。このデフォルト・ストリーム・パラメータの設定は、次のとおりです。

```
stream=tcpip;host=localhost
```

参照

- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページ
- Ultra Light for MobileVB：「ULStreamType 列挙」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「ULStreamType 列挙」 『Ultra Light - .NET プログラミング』
- Ultra Light for Embedded SQL：「ULSetSynchInfo 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「SetSynchInfo 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for C/C++：「ul_synch_info_a 構造体」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「setStream メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

Windows CE の ActiveSync

Windows CE のデータは、Ethernet や Wi-Fi 接続を介して同期することもできますが、この項では ActiveSync 同期で使用するためにデスクトップとデバイスを設定する方法について説明します。別の方法を使用して同期を直接行う場合は、適切な同期機能を使用してアプリケーションのプログラムを設定する必要があります。

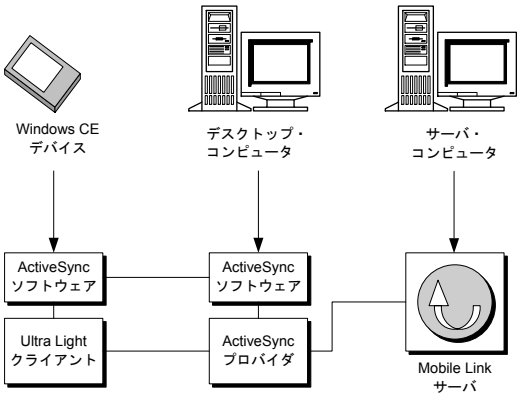
ActiveSync 起動の同期を使用するには、次のことが必要です。

- ActiveSync を使用して ActiveSync 起動の同期を使用するのに必要なすべてのアプリケーションを登録する。
- ActiveSync プロバイダをデスクトップにインストールし、デバイスに配備する。

  プロバイダがサポートされているプラットフォームについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある Ultra Light の表の「同期」の項を参照してください。

ActiveSync アーキテクチャ

次の図は、ActiveSync アーキテクチャで必要なコンピューティング・レイヤを示しています。



ActiveSync Manager は、デスクトップだけでなくデバイスにもインストールしてください。1 台のコンピュータでは 1 つの ActiveSync プロバイダのみ使用できます。ただし、複数の Ultra Light

アプリケーションを Windows CE デバイスにインストールしている場合は、アプリケーションを同じプロバイダに登録して同時に同期が行われるようにすることができます。

参照

- Ultra Light for AppForge：「データの同期」『Ultra Light - AppForge プログラミング』
- Ultra Light for C/C++：「アプリケーションへの ActiveSync 同期の追加」『Ultra Light - C/C++ プログラミング』
- Ultra Light for Embedded SQL：「アプリケーションへの同期の追加」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「データの同期」『Ultra Light - M-Business Anywhere プログラミング』

ActiveSync 同期の概要

1. ActiveSync Manager によって同期セッションが開始されます。
2. ActiveSync プロバイダは、同期の通知メッセージをデバイス上の最初に登録されているアプリケーションに送信します。アプリケーションが実行中でない場合は、起動されます。
3. 登録されたアプリケーションごとに WndProc が起動されます。
4. アプリケーションは、メッセージが ActiveSync からの同期通知メッセージであると判断すると、ULIsSynchronizeMessage を呼び出してデータベース同期プロシージャを起動します。
5. 同期処理が完了すると、アプリケーションは ULSignalSyncIsComplete を呼び出して、同期が終了したことをプロバイダに通知します。
6. プロバイダに登録されている各アプリケーションについて、手順 2 ー 5 が繰り返されます。

エンド・ユーザのデスクトップへの ActiveSync プロバイダの配備

Ultra Light は、開発の段階で SQL Anywhere インストーラを使用して、デスクトップにインストールすることになります。ただし、Ultra Light をエンド・ユーザに配備する場合は、エンド・ユーザのコンピュータに ActiveSync プロバイダを手動でインストールして登録する必要があります。これにより、ActiveSync は特定のアプリケーションに対してプロバイダの特定のインスタンスを呼び出すタイミングを認識します。

- **mlasinst.exe**　ActiveSync プロバイダをインストールして、ActiveSync Manager で登録します。このユーティリティは、同期用に ActiveSync プロバイダで使用するアプリケーションも登録します。
- **mlasdesk.dll**　ActiveSync Manager によってデスクトップにロードされる DLL。DLL ファイルのロケーションは、*mlasinst.exe* によって ActiveSync Manager に登録されます。
- **mlasdev.dll**　ActiveSync Manager によってデバイスにロードされる DLL。DLL ファイルは、*mlasinst.exe* によってデバイスの正しいロケーションに配備されます。
- **dblgen10.dll**　言語リソース・ライブラリ。

プロバイダがサポートされているプラットフォームについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」にある Ultra Light の表の「同期」の項を参照してください。

♦ ActiveSync アプリケーションをインストールするには、次の手順に従います。

1. エンド・ユーザのコンピュータで次のことを確認します。
   - ActiveSync Manager がインストールされていること。
   - ActiveSync プロバイダ・ファイルが開発マシンからエンド・ユーザのハード・ドライブにコピーされていること。デフォルト・ロケーションを使用する場合は、*install-dir¥win32* ディレクトリを再作成してください。
2. mlasinst を実行して、ActiveSync のプロバイダをインストールします。このユーティリティを使用して、Ultra Light アプリケーションをユーザの Windows CE デバイスに登録して配備することもできます。ただし、可能かどうかは、使用するコマンド・ライン構文によります。Ultra Light アプリケーションで複数のファイルを使用する場合は、必要なファイルを手動でコピーしてください。

   たとえば、*mlasdesk.dll* と *mlasdev.dll* はすでにユーザの *install-dir¥win32* ディレクトリにコピーされているため、次のコマンド・ラインでは -k と -v オプションが省略されています。ただし、このユーティリティでは *myULapp.exe* アプリケーションを *c:¥My Files* からデバイスの *¥Program Files¥* にコピーする必要もあります。-p -x 引数は、ActiveSync によって起動されるときの *myULapp.exe* に対するコマンド・ライン・オプションです。

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

   注意
   ActiveSync マネージャを使用して、後で Ultra Light アプリケーションを登録することもできます。「ActiveSync Manager を使用したアプリケーションの登録」 380 ページを参照してください。
3. ActiveSync が新しいプロバイダを認識できるよう、コンピュータを再起動します。
4. Mobile Link プロバイダを有効にします。
   a. [ActiveSync] ウィンドウで [オプション] をクリックします。
   b. リストにある [Mobile Link クライアント] を有効にして [OK] をクリックし、Mobile Link プロバイダをアクティブにします。
   c. 登録されたアプリケーションのリストを表示するには、[オプション] をクリックし、[Mobile Link クライアント] を選択して [設定] をクリックします。

参照

- 「ActiveSync Manager を使用したアプリケーションの登録」 380 ページ
- 「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」 29 ページ

ActiveSync Manager を使用したアプリケーションの登録

ActiveSync で使用するアプリケーションを登録するには、ActiveSync プロバイダのインストール・ユーティリティを使用する方法と、ActiveSync Manager 自体を使用する方法があります。この項では、ActiveSync Manager を使用する方法について説明します。

- **ActiveSync Manager で使用するアプリケーションを登録するには、次の手順に従います。**

1. ActiveSync を起動します。
2. [ActiveSync] ウィンドウで [オプション] を選択します。
3. 情報タイプのリストから、[Mobile Link クライアント] を選択し、[設定] をクリックします。
4. [Mobile Link 同期] ダイアログで [新規] をクリックします。[プロパティ] ダイアログが表示されます。
5. アプリケーションについて次の情報を入力します。
   - **[アプリケーション名]** ActiveSync ユーザ・インタフェースに表示されるアプリケーションを識別する名前。
   - **[クラス名]** アプリケーションの登録済みクラス名。「アプリケーションに対するクラス名の割り当て」『Ultra Light - C/C++ プログラミング』を参照してください。
   - **[パス]** アプリケーションのデバイス上のロケーション。
   - **[引数]** ActiveSync でアプリケーションの起動時に使用するコマンド・ライン引数。
6. [OK] をクリックしてアプリケーションを登録します。

参照

- 「ActiveSync プロバイダ・インストール・ユーティリティ [mlasinst]」 29 ページ

第 16 章

Ultra Light 同期パラメータとネットワーク・プロトコル・オプション

目次

Ultra Light の同期パラメータ

同期パラメータは、Ultra Light データベースと Mobile Link サーバ間の同期を制御します。パラメータの設定方法は、使用している Ultra Light インタフェースによって異なります。この章では、パラメータの影響について説明し、これらの設定方法が説明された他の項へのリンクを示します。

注意
この章で説明するパラメータが適用されるのは、Ultra Light リモート・データベースだけです。SQL Anywhere リモート・データベースを同期させる場合は、「Mobile Link SQL Anywhere クライアント・ユーティリティ [dbmlsync]」 117 ページを参照してください。

API 固有の詳細情報については、次の項を参照してください。

- Ultra Light for Embedded SQL：
- Ultra Light for Mobile VB：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「ULSyncParms クラス」 『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「Synchronize 関数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

必須パラメータ

次のパラメータは必須です。

- Stream Type パラメータ。「Stream Type 同期パラメータ」 399 ページを参照してください。
- User Name パラメータ。「User Name 同期パラメータ」 405 ページを参照してください。
- Version パラメータ。「Version 同期パラメータ」 406 ページを参照してください。

これらのパラメータが設定されていない場合、同期関数は例外 (SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

パラメータの競合

次のパラメータは、いずれか 1 つだけ指定できます。

- Download Only パラメータ。「Download Only 同期パラメータ」 386 ページを参照してください。
- Ping パラメータ。「Ping 同期パラメータ」 392 ページを参照してください。
- Upload Only パラメータ。「Upload Only 同期パラメータ」 404 ページを参照してください。

これらのパラメータが複数 true に設定されると、同期関数は例外 (たとえば SQLCode.SQLE_SYNC_INFO_INVALID またはこれと同じもの) をスローします。

Authentication Parameters 同期パラメータ

Mobile Link イベントの認証パラメータにパラメータを渡します。

構文

構文は、使用する API によって異なります。

説明

パラメータには、ユーザ名やパスワードなどがあります。

このパラメータを使用する場合、パラメータの数も指定してください。「Number of Authentication Parameters パラメータ」 390 ページを参照してください。

指定可能な値

文字列の配列。文字列の値として Null は使用できませんが、空の文字列は指定できます。

参照

- 「Number of Authentication Parameters パラメータ」 390 ページ
- 「認証パラメータ」 『Mobile Link - サーバ管理』
- 「authenticate_parameters 接続イベント」 『Mobile Link - サーバ管理』
- Ultra Light for C/C++ :「auth_parms 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for MobileVB :「AddAuthenticationParm メソッド」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「AuthenticationParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 setAuthenticationParms メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_char * Params[ 3 ] = { UL_TEXT( "parm1" ),
                          UL_TEXT( "parm2" ),
                          UL_TEXT( "parm3" ) };
// ...
info.num_auth_parms = 3;
info.auth_parms = Params;
```

Authentication Status 同期パラメータ

Mobile Link のユーザ認証のステータスをレポートします。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。

指定可能な値

使用できる値は、インタフェース固有の列挙に格納されています。たとえば、C/C++ アプリケーションの列挙は次のようになります。

定数	値	説明
UL_AUTH_STATUS_UNKNOWN	0	認証ステータスが不明です。接続がまだ同期していない可能性があります。
UL_AUTH_STATUS_VALID	1	ユーザ ID とパスワードは、同期時には有効でした。
UL_AUTH_STATUS_VALID_BUT_EXPIRES_SOON	2	ユーザ ID とパスワードは、同期時には有効でしたが、まもなく有効期限が切れます。
UL_AUTH_STATUS_EXPIRED	3	認証に失敗しました。ユーザ ID またはパスワードの有効期限が切れています。
UL_AUTH_STATUS_INVALID	4	認証に失敗しました。不正なユーザ ID またはパスワードです。
UL_AUTH_STATUS_IN_USE	5	認証に失敗しました。ユーザ ID はすでに使用されています。

説明

統合データベースでカスタム **authenticate_user** 同期スクリプトが異なる値を返す場合は、authenticate_user 接続イベントで指定されているルールに従って値が解釈されます。「authenticate_user 接続イベント」『Mobile Link - サーバ管理』を参照してください。

カスタム認証スキームを実装している場合、authenticate_user または authenticate_user_hashed 同期スクリプトは、このパラメータの有効値の 1 つを返します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「Mobile Link ユーザ」 9 ページ.
- Ultra Light for C/C++：「auth_status 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncResult クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「AuthStatus プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info_a info;
// ...
returncode = info.auth_status;
```

Authentication Value 同期パラメータ

カスタム Mobile Link のユーザ認証スクリプトの結果をレポートします。Mobile Link サーバが、この情報をクライアントに提供します。

構文

構文は、使用する API によって異なります。

説明

デフォルトの Mobile Link ユーザ認証メカニズムによって設定される値については、authenticate_user 接続イベントと Authentication Status 同期パラメータの項で説明します。

このパラメータは Mobile Link サーバによって設定されるため、読み込み専用です。

参照

- 「authenticate_user 接続イベント」『Mobile Link - サーバ管理』
- 「authenticate_user_hashed 接続イベント」『Mobile Link - サーバ管理』
- 「Authentication Status 同期パラメータ」 383 ページ
- Ultra Light for C/C++：「auth_value 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncResult クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「AuthValue プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info_a info;
// ...
returncode = info.auth_value;
```

Checkpoint Store 同期パラメータ

同期中にデータベースのチェックポイントを追加して、同期プロセス中のデータベースの拡張を制限します。

構文

構文は、使用する API によって異なります。

デフォルト

制限付きのチェックポイント

説明

チェックポイント操作は、アプリケーションと Palm コンジットの I/O 操作を増やすため、同期が低速になります。このオプションは、多くの更新を行う大規模なダウンロードに最適です。低速なフラッシュ・メモリを使用するデバイスでは、パフォーマンスの低下が望ましくないことがあります。

参照

- Ultra Light for C/C++：「checkpoint_store 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「CheckpointStore プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定します。

```
ul_synch_info_a info;
// ...
info.checkpoint_store = ul_true ;
```

Disable Concurrency 同期パラメータ

この同期中は他のスレッドからデータベースへのアクセスを禁止します。

構文

構文は、使用する API によって異なります。

デフォルト

False (データベースへの同時アクセスを許可)アップロード時のデータ・アクセスは読み込み専用で、それ以外は読み込み、書き込み可能です。

指定可能な値

Boolean

参照

- 「Ultra Light での同時実行性」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「disable_concurrency 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：なし
- Ultra Light.NET：「DisableConcurrency プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.disable_concurrency = ul_true ;
```

Download Only 同期パラメータ

この同期中は、Ultra Light データベースから変更をアップロードしません。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

競合するパラメータ

ping とアップロード専用

説明

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。

ulsync の場合　ダウンロード専用同期が発生する場合、ulsync はデータの変更をアップロードしません。このとき、次の処理が行われます。

- スキーマに関する情報と進行状況のカウンタに格納されている値をアップロードします。
- ダウンロード専用同期中に、リモートでの変更が上書きされないようにします。

これらの処理は、ulsync で Ultra Light データベース・ログをスキャンし、統合データベースへの操作が保留中のローを追跡することで行われます。ulsync で競合が検出されると、データベースがロールバックされ、同期は失敗します。この競合を解決するには、完全な同期 (アップロードとダウンロード) を行う必要があります。

参照

- 「Upload Only 同期パラメータ」 404 ページ
- 「進行状況のカウンタ」 350 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「download_only 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「DownloadOnly プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync では、拡張同期パラメータとしてこのパラメータがサポートされています。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;DownloadOnly=ON"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
returncode=info.ignored_rows;
```


Ignored Rows 同期パラメータ

同期中にスクリプトがないために Mobile Link サーバによってローが 1 つでも無視されると、true に設定されます。

構文

構文は、使用する API によって異なります。

指定可能な値

ブール式

説明

このパラメータは読み込み専用です。

参照

- Ultra Light for C/C++：「ignored_rows 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncResult クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「IgnoredRows プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info_a info;
// ...
info.ignored_rows = ul_true;
```

Keep Partial Download 同期パラメータ

同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで部分的なダウンロードを保持するかどうかを制御します。

構文

構文は、使用する API によって異なります。

デフォルト

False (ダウンロードが失敗したときにすべての変更をロールバック)

指定可能な値

ブール式

参照

- 「失敗したダウンロードの再開」『Mobile Link - サーバ管理』
- 「Resume Partial Download 同期パラメータ」 395 ページ
- Ultra Light for C/C++：「keep_partial_download 変数」『Ultra Light - C/C++ プログラミング』

- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「KeepPartialDownload プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.keep_partial_download = ul_true;
```

New Password 同期パラメータ

ユーザ名に対する新しい Mobile Link パスワードを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

説明

このパラメータはオプションです。

参照

- 「Mobile Link ユーザ」 9 ページ.
- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「new_password 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「NewPassword プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;Newpassword=mynewpassword"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.password = UL_TEXT( "myoldpassword" );
info.new_password = UL_TEXT( "mynewpassword" );
```

Number of Authentication Parameters パラメータ

Mobile Link イベントの認証パラメータに渡す認証パラメータの数を指定します。

構文

構文は、使用する API によって異なります。

デフォルト

カスタム認証スクリプトに渡されるパラメータはありません。

説明

Authentication Parameters とともにパラメータを使用して、カスタム認証スクリプトに情報を提供します。

参照

- 「Authentication Parameters 同期パラメータ」 383 ページ
- 「authenticate_parameters 接続イベント」 『Mobile Link - サーバ管理』
- 「認証パラメータ」 『Mobile Link - サーバ管理』
- Ultra Light for C/C++ :「num_auth_parms 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「AuthenticationParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.num_auth_parms = 3;
```

Observer 同期パラメータ

同期をモニタするコールバック関数またはイベント・ハンドラへのポインタです。

構文

構文は、使用する API によって異なります。

参照

- 「User Data 同期パラメータ」 405 ページ
- Ultra Light for C/C++ :「observer 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULConnection クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「ULSyncProgressListener メンバ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 synchronizeWithParm メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.observer=callfunction;
```

Partial Download Retained 同期パラメータ

同期時の通信エラーが原因でダウンロードが失敗したときに、変更をロールバックしないで、ダウンロードされたこの変更が適用されたかどうかを示します。

構文

構文は、使用する API によって異なります。

指定可能な値

ブール式

説明

ダウンロード・エラーが発生して部分的なダウンロードが保持された場合、同期時にパラメータが設定されます。

部分的ダウンロードが保持されるのは、Keep Partial Download が true に設定されている場合のみです。「Keep Partial Download 同期パラメータ」 388 ページを参照してください。

参照

- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「Resume Partial Download 同期パラメータ」 395 ページ
- Ultra Light for C/C++：「partial_download_retained 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncResult クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「PartialDownloadRetained プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

このパラメータにアクセスするには、次のように入力します。

```
ul_synch_info_a info;
// ...
returncode=info.partial_download_retained;
```

Password 同期パラメータ

ユーザ名に対する Mobile Link パスワードを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

説明

このパラメータはオプションです。

この Mobile Link のユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。「User Name 同期パラメータ」 405 ページを参照してください。

Mobile Link クライアントにパスワードが設定されている場合は、New Password パラメータを使用してパスワードを変更します。「New Password 同期パラメータ」 389 ページを参照してください。

参照

- 「Mobile Link ユーザ」 9 ページ.
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「password 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「Password プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;Password=mypassword"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.password = UL_TEXT( "mypassword" );
```

Ping 同期パラメータ

Ultra Light クライアントと Mobile Link サーバ間の通信を確認します。このパラメータが true に設定されている場合は、同期は行われません。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

説明

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。

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。

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 ID 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 に追加します。

Mobile Link サーバに次のスクリプトが存在する場合、ping 要求に対してこれらのスクリプトを実行できます。

- begin_connection
- authenticate_user
- authenticate_user_hashed
- end_connection

参照

- 「-pi オプション」 153 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「ping 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「PingOnly プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;Ping=True"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.ping = ul_true;
```

publication 同期パラメータ

同期させるパブリケーションを指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync で使用することもできます。

デフォルト

すべてのパブリケーションを同期します。

説明

C/C++ での同期時には、publication 同期パラメータを「**パブリケーション・マスク**」、つまり、パブリケーション定数を OR で結合したリストに設定します。「PublicationMask 同期パラメータ」 394 ページを参照してください。

参照

- 「Ultra Light のパブリケーション」 360 ページ
- 「Ultra Light のパブリケーションの操作」 『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「publication 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULPublicationSchema クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「ULPublicationSchema クラス」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 PublicationSchema クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e
"Username=remoteA;Version=2;Publication=UL_PUB_MYPUB1,UL_PUB_MYPUB2"
```

PublicationMask 同期パラメータ

最終ダウンロード時間を取得するパブリケーションを OR で結合したリスト。このセットはマスクとして提供されます。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

0、またはすべてのデータの同期

説明

各パブリケーションにはバイナリ・マスクがあります。マスクを OR で結合することによって、同期するパブリケーションを指定します。特殊なパブリケーション・マスク UL_SYNC_ALL は、パブリケーションに含まれるかどうかに関係なく、データベース内のすべてのテーブルを表します。マスク UL_SYNC_ALL_PUBS は、データベース内のパブリケーションに含まれるすべてのテーブルを表します。

注意
スキーマ変更によりマスクが更新される場合があるので、Publications 同期パラメータを使用して、名前の付いたパブリケーションを使用する必要があります。「publication 同期パラメータ」 394 ページを参照してください。

参照

- 「Ultra Light のパブリケーション」 360 ページ
- 「Ultra Light のパブリケーションの操作」 『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for AppForge：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「PublicationMask プロパティ」 『Ultra Light - .NET プログラミング』

例

Ultra Light for C/C+ アプリケーションでこのパラメータを設定するには、次のようにします。

```
ul_synch_info_a info;
// ...
info.publication = UL_PUB_MYPUB1 | UL_PUB_MYPUB2 ;
```

Resume Partial Download 同期パラメータ

失敗したダウンロードを再開します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

説明

同期によって変更はアップロードされず、失敗したダウンロードでダウンロードされるはずだった変更のみがダウンロードされます。

参照

- 「失敗したダウンロードの再開」 『Mobile Link - サーバ管理』
- 「Keep Partial Download 同期パラメータ」 388 ページ
- 「Partial Download Retained 同期パラメータ」 391 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「resume_partial_download 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「ResumePartialDownload プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.resume_partial_download = ul_true;
```

Send Column Names 同期パラメータ

アップロード時にカラム名が送信されるように指定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

説明

このオプションは、Mobile Link サーバによってダイレクト・ロー・ハンドリングに使用されます。ダイレクト・ロー・ハンドリングを使用する場合は、このオプションを有効にする必要があります。それ以外の場合、このオプションは効果がありません。「ダイレクト・ロー・ハンドリング」 『Mobile Link - サーバ管理』を参照してください。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「send_column_names 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「SendColumnNames プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;SendColumnNames = true"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.send_column_names = ul_true;
```

Send Download Acknowledgment 同期パラメータ

クライアントからダウンロード確認を提供するかどうかを Mobile Link サーバに指示します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

ブール式

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「send_download_ack 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「SendDownloadAck プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;SendDownloadACK = true"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.send_download_ack = ul_true;
```

Stream Error 同期パラメータ

通信エラー・レポート情報を保持する構造体です。

構文

構文は、使用する API によって異なります。

適用対象

このパラメータは、C/C++ インタフェースにのみ適用されます。

指定可能な値

このパラメータにはデフォルト値がないので、サポートされているフィールドの一つを使用して明示的に設定する必要があります。**ul_stream_error** のフィールドは、次のとおりです。

- **stream_id** 必須ではありません。値は常にゼロです。
- **stream_context** 開く、読み込み、または書き込みなど、実行される基本的なネットワーク・オペレーション。詳細については、*sserror.h* を参照してください。
- **stream_error_code** 必須ではありません。値は常にゼロです。

  エラー番号のリストについては、「Mobile Link 通信エラー・メッセージ」 『SQL Anywhere 10 - エラー・メッセージ』を参照してください。エラー・コードのサフィックスについては、*sserror.h* を参照してください。
- **system_error_code** システム固有のエラー・コード。エラー・コードの詳細については、プラットフォームのマニュアルを参照してください。Windows プラットフォームの場合は、Microsoft Developer Network (MSDN) マニュアルを参照してください。
  Windows における一般的なシステム・エラーを次に示します。
  - **10048 (WSAADDRINUSE)** アドレスがすでに使用されています。
  - **10053 (WSAECONNABORTED)** ソフトウェアが接続のアボートを引き起こしました。
  - **10054 (WSAECONNRESET)** 通信の他方の側がソケットを閉じました。
  - **10060 (WSAETIMEDOUT)** 接続がタイムアウトしました。
  - **10061 (WSAECONNREFUSED)** 接続が拒否されました。通常、Mobile Link サーバが稼働していないか、指定されたポートで受信していません。Microsoft Developer Network Web サイトを参照してください。
- **error_string** アプリケーションが提供するエラー・メッセージ。文字列は空の場合もあれば、空でない場合もあります。空でない error_string は、stream_error_code とともに情報を提供します。たとえば、書き込みエラー (エラー・コード 9) の場合、エラー文字列は、書き込もうとしたバイト数を示します。
- **error_string_length** エラー文字列バッファのサイズ

説明

Ultra Light C++ コンポーネント以外の Ultra Light アプリケーションは、Sync Result パラメータの一部として通信エラー情報を受け取ります。「Sync Result 同期パラメータ」 401 ページを参照してください。

stream_error フィールドは **ul_stream_error** 型の構造体です。


```
typedef struct ss_error_a {
    ss_stream_id        stream_id;
    ss_stream_context   stream_context;
    ss_error_code       stream_error_code;
    asa_uint32          system_error_code;
    char            *error_string;
    asa_uint32          error_string_length;
} ss_error_a, *p_ss_error_a;
```

この構造体は、SQL Anywhere のインストール・ディレクトリの *h* サブディレクトリにある *sserror.h* ファイルで定義されています。

次のように SQLE_COMMUNICATIONS_ERROR をチェックします。

```
Connection conn;
auto ul_synch_info_a info;
...
conn.InitSynchInfo( &info );
info.stream_error.error_string = error_buff;
info.stream_error.error_string_length =
                sizeof( error_buff );
if( !conn.Synchronize( &synch_info ) ){
    if( SQLCODE == SQLE_COMMUNICATIONS_ERROR ){
        printf( error_buff );
    // more error handline here
```

参照

- 「stream_error 変数」 『Ultra Light - C/C++ プログラミング』

Stream Type 同期パラメータ

同期に使用する Mobile Link ネットワーク・プロトコルを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

説明

このパラメータは必須です。デフォルト値はありません。

ほとんどのネットワーク・プロトコルでは、Mobile Link サーバ・アドレスや他の動作を識別するプロトコル・オプションが必要です。これらのオプションは、Stream Parameters パラメータで指定します。「Stream Parameters 同期パラメータ」 400 ページを参照してください。

ネットワーク・プロトコルがオプションを必要とする場合は、Stream Parameters を使用してそのオプションを渡すか、あるいは Stream Parameters パラメータを null に設定します。

ストリームのタイプには次の種類がありますが、すべてのターゲット・プラットフォームですべての関数を使用できるわけではありません。

ネットワーク・プロトコル	説明
HTTP	HTTP を介して同期します。

ネットワーク・プロトコル	説明
HTTPS	HTTPS を介して同期します。 HTTPS プロトコルは、基本のセキュリティ・レイヤとして TLS を使用します。TCP/IP を介して機能します。
TCP/IP	TCP/IP を介して同期します。
TLS	トランスポート・レイヤ・セキュリティ (TLS) 付きの TCP/IP を介して同期します。TLS は、デジタル証明書とパブリック・キー暗号方式を使用して、クライアント／サーバ通信をセキュリティ保護します。

サポートされているプラットフォームのリストについては、「プラットフォーム別 SQL Anywhere 10.0.1 コンポーネント」に掲載されている Ultra Light の表を参照してください。

参照

- 「証明書作成ユーティリティ [createcert]」『SQL Anywhere サーバ - データベース管理』
- 「証明書ビューワ・ユーティリティ [viewcert]」『SQL Anywhere サーバ - データベース管理』
- 「トランスポート・レイヤ・セキュリティ」『SQL Anywhere サーバ - データベース管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページ
- Ultra Light for C/C++：「stream 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「Stream プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定します。

```
Connection conn;
auto ul_synch_info_a info;
...
conn.InitSynchInfo( &info );
info.stream = "http";
```

Stream Parameters 同期パラメータ

ネットワーク・プロトコルを設定するオプションを設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

Null

指定可能な値

文字列

説明

このパラメータはオプションです。

ネットワーク・プロトコル・オプションのリストで、セミコロンで区切られます。各オプションは *keyword=value* の形式で、許可されるキーワード・セットはネットワーク・プロトコルによって異なります。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- 「Ultra Light 同期ストリームのネットワーク・プロトコルのオプション」 408 ページ
- Ultra Light for C/C++ :「stream_parms 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET :「StreamParms プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.stream_parms= UL_TEXT( "host=myserver;port=2439" );
```

Sync Result 同期パラメータ

同期のステータスをレポートします。

構文

構文は、使用する API によって異なります。

説明

このパラメータは Ultra Light によって設定されるため、読み込み専用です。

C/C++ インタフェースでは、この情報を ul_sync_info 構造体の中で複数の別個のパラメータで受け取ります。それ以外の場合は、この情報は、さまざまな情報がそれぞれ別個のフィールドに格納された複合パラメータとして定義されます。

- **Authentication Status**　認証の成功または失敗をレポートします。「Authentication Status 同期パラメータ」 383 ページを参照してください。
- **Ignored Rows**　無視されるローの数をレポートします。「Ignored Rows 同期パラメータ」 388 ページを参照してください。

- **Stream Error 情報**　Stream Error 情報には、Stream Error Code、Stream Error Context、Stream Error ID、Stream Error System が含まれています。「Stream Error 同期パラメータ」 397 ページを参照してください。
- **Upload OK**　アップロード・フェーズの成功または失敗をレポートします。「Upload OK 同期パラメータ」 403 ページを参照してください。

参照

- Ultra Light for AppForge：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「ULSyncParms クラス」 『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「ul_synch_info_a 構造体」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』
- Ultra Light for Embedded SQL：

Table Order 同期パラメータ

Ultra Light のデフォルトのテーブルの順序が適切でない場合に、優先同期に必要なテーブルの順序を設定します。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

Null または空の文字列。外部キーの関係に基づいてテーブルを並べます。

指定可能な値

アップロードするすべてのテーブルを示す、カンマで区切られたテーブル名のリスト

説明

統合データベースの外部キーが Ultra Light リモートに一致し、外部キー循環がない場合は、通常、デフォルトの順序を使用できます。

一重または二重引用符でテーブル名を囲みます。たとえば、"Customer,Sales" と 'Customer,Sales' の両方が Ultra Light でサポートされています。ulsync では、テーブル名をセミコロン (;) で区切ります。

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。

ダウンロードのテーブルの順序は、アップロードで定義した順序と同じです。

Ultra Light テーブルが次のいずれかに該当する場合は、テーブル順序を明示的に設定するだけですみます。

- 外部キー循環に含まれる場合。循環に含まれるすべてのテーブルをリストする必要があります。

- 統合データベース内の外部キーの関係が異なる場合。

> **注意**
> 定義するテーブル順序は、参照整合性を保つようにします。リストしないテーブルは、リモート・データベースで定義した外部キーに基づいて適切に保存されます。

参照

- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「TableOrder プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for C/C++：「table_order 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;TableOrder='Customer;Sales'"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.table_order = UL_TEXT( "Customer,Sales" );
```

Upload OK 同期パラメータ

Mobile Link サーバにアップロードされたデータのステータスをレポートします。

構文

構文は、使用する API によって異なります。

説明

このパラメータは Ultra Light によって設定されるため、読み込み専用です。

同期後、このパラメータには、アップロードが成功した場合は **true**、それ以外の場合は **false** が入ります。同期エラーがあったかどうかについてこのパラメータを確認することによって、エラーが発生する前にデータが正常にアップロードされたかどうかを確認できます。

参照

- Ultra Light for C/C++：「upload_ok 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncResult クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「UploadOK プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncResult クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

Ultra Light for C/C++ アプリケーションは、次のようにパラメータにアクセスできます。

```
ul_synch_info_a info;
// ...
returncode = info.upload_ok;
```

Upload Only 同期パラメータ

現在の同期中にダウンロードは発生しないことを示します。これにより、特に低速の通信リンクでは、通信時間を節約できます。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

デフォルト

False

指定可能な値

Boolean

競合するパラメータ

Download Only、Ping、Resume Partial Download

説明

true に設定すると、クライアントは Mobile Link サーバからのアップロード確認を待ってから、同期セッションを正常終了します。

参照

- 「Ultra Light での同期の設計」 358 ページ
- 「Download Only 同期パラメータ」 386 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「upload_only 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」 『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「UploadOnly プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「 SyncParms クラス」 『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;UploadOnly = True"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.upload_only = ul_true;
```

User Data 同期パラメータ

アプリケーション固有の情報を同期 observer で使用できるようにします。

適用対象

C/C++ と MobileVB アプリケーションのみ。Ultra Light.NET などのコンポーネントは、ユーザ・データの処理に別個のパラメータを必要としないので、User Data パラメータはありません。

構文

構文は、使用する API によって異なります。

説明

同期 observer コールバック関数またはイベント・ハンドラの実装時に、User Data パラメータを使用して情報を指定することによって、アプリケーション固有の情報を使用可能にできます。

参照

- 「Observer 同期パラメータ」 390 ページ
- Ultra Light for C/C++ :「user_data 変数」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULConnection クラス」 『Ultra Light - AppForge プログラミング』

User Name 同期パラメータ

Mobile Link サーバが認証用に使用する文字列。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

説明

パラメータにはデフォルト値がないので、明示的に設定してください。

この Mobile Link のユーザ名とパスワードは他のデータベース・ユーザ ID やパスワードとは別のもので、アプリケーションを Mobile Link サーバに対して識別し、認証するためだけに使用されます。「Password 同期パラメータ」 391 ページを参照してください。

同期システムに含まれるユーザの場合は、Mobile Link サーバを使用してユーザ名を登録する必要があります。ユーザ名は、統合データベースの ml_user Mobile Link システム・テーブルの名前カラムに格納されます。

リモート ID が使用されている場合は、ユーザ名はユニークである必要はありません。「リモート ID」 15 ページを参照してください。

参照

- 「Mobile Link ユーザ」 9 ページ
- 「Ultra Light ユーザ認証」 10 ページ
- 「Ultra Light 同期ユーティリティ (ulsync)」 『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++ :「user_name 変数」 『Ultra Light - C/C++ プログラミング』

- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「UserName プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2"
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.user_name= UL_TEXT( "mluser" );
```

Version 同期パラメータ

Ultra Light アプリケーションで、同期スクリプトのセットから 1 種類を選択することを可能にします。

構文

構文は、使用する API によって異なります。このパラメータは ulsync を使用して設定することもできます。

指定可能な値

文字列

説明

このパラメータは必須です。

統合データベースの同期スクリプトは、それぞれバージョン文字列で区別されます。たとえば、異なる文字列バージョンによって特定される 2 種類の **download_cursor** スクリプトがあります。

参照

- 「スクリプト・バージョン」『Mobile Link - サーバ管理』
- 「Ultra Light 同期ユーティリティ (ulsync)」『Ultra Light - データベース管理とリファレンス』
- Ultra Light for C/C++：「version 変数」『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge：「ULSyncParms クラス」『Ultra Light - AppForge プログラミング』
- Ultra Light.NET：「Version プロパティ」『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere：「SyncParms クラス」『Ultra Light - M-Business Anywhere プログラミング』

例

ulsync は、次のようにこのパラメータを拡張同期パラメータとして設定できます。

```
ulsync -c DBF=myuldb.udb -k http -e "Username=remoteA;Version=2;
```

Ultra Light for C/C++ アプリケーションは、次のようにパラメータを設定できます。

```
ul_synch_info_a info;
// ...
info.version = UL_TEXT( "default" );
```


Ultra Light 同期ストリームのネットワーク・プロトコルのオプション

アプリケーションでネットワーク・プロトコルを設定する必要があります。各 Ultra Light データベースは、ネットワーク・プロトコルによって Mobile Link サーバと同期します。使用可能なネットワーク・プロトコルには、TCP/IP、HTTP 、HTTPS、TLS があります。また、Palm OS での HotSync 同期、Windows CE での ActiveSync 通知がサポートされています。各プロトコルでサポートされているオプションについては、「Mobile Link クライアントのネットワーク・プロトコル・オプション」 35 ページを参照してください。

別途ライセンスが必要な必須コンポーネント
ECC 暗号化と FIPS 承認の暗号化には、別途ライセンスが必要です。強力な暗号化テクノロジはすべて、輸出規制対象品目です。
「別途ライセンスが必要なコンポーネント」 『SQL Anywhere 10 - 紹介』を参照してください。

TLS を使用するために Ultra Light を設定する方法については、「トランスポート・レイヤ・セキュリティを使用する Ultra Light クライアントの設定」 『SQL Anywhere サーバ - データベース管理』を参照してください。

設定したネットワーク・プロトコルでは、対応するプロトコル・*オプション*のセットから選択することによって、Ultra Light アプリケーションが Mobile Link 同期サーバを特定して正しく通信できるようにします。ネットワーク・プロトコルのオプションは、アドレス情報 (ホストとポート) やプロトコル固有の情報などを提供します。使用しているストリーム・タイプに使えるオプションを判別するには、下記を参照してください。

同期ストリームとオプションの設定

Stream Parameters パラメータを設定することによって、アプリケーション内で Mobile Link サーバを特定するのに必要な情報を提供できます。「Stream Parameters 同期パラメータ」 400 ページを参照してください。

ただし、HotSync を使用している場合、Stream Parameter の値を指定しない場合、または値を NULL として指定した場合は、必要なパラメータを HotSync Manager から入力できます。「Mobile Link 同期のプロトコル・オプションの設定」 374 ページを参照してください。

Ultra Light 同期の呼び出しでストリーム・パラメータを指定する方法については、次の項を参照してください。

- Ultra Light for C/C++ :「Palm アプリケーションに HotSync 同期を追加する」 『Ultra Light - C/C++ プログラミング』
- Ultra Light for AppForge :「ULSQLCode 列挙」 『Ultra Light - AppForge プログラミング』
- Ultra Light for Ultra Light.NET :「StreamParms プロパティ」 『Ultra Light - .NET プログラミング』
- Ultra Light for M-Business Anywhere :「 setStreamParms メソッド」 『Ultra Light - M-Business Anywhere プログラミング』

zlib 圧縮に関する注意
zlib 圧縮は、Palm OS または Symbian OS ではサポートされていません。

参照

- 「Mobile Link クライアントのネットワーク・プロトコル・オプション」 35 ページ

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

と

な

に

ね

は

ひ

ふ

へ

ほ

ま

み

も

ゆ

よ

り

れ

ろ

