

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

Mobile Link - サーバ起動同期

目次

1	Mobile Link - サーバ起動同期	4
1.1	サーバ起動同期.....	4
	サーバ起動同期のコンポーネント.....	6
	サーバ起動同期の配備に関する考慮事項.....	7
	サーバ起動同期のクイックスタート.....	8
1.2	サーバ起動同期の設定.....	8
	Push 要求.....	9
	Notifier.....	15
	Listener.....	17
	ライトウェイトポーラー.....	25
	ゲートウェイと Carrier.....	26
1.3	サーバ起動同期の Mobile Link サーバ設定.....	32
	ml_add_property システムプロシージャを使用したサーバ側の設定.....	33
	SQL Central を使用した Notifier、ゲートウェイ、または Carrier の設定.....	34
	Notifier 設定ファイルを使用したサーバ側の設定.....	36
	Notifier イベント.....	39
	共通プロパティ.....	52
	Notifier プロパティ.....	52
	ゲートウェイプロパティ.....	54
	Carrier プロパティ.....	59
1.4	Windows デバイス用の Mobile Link Listener ユーティリティ (dbsln).....	59
	Listener の配備のセキュリティ保護.....	61
	Windows デバイス用の受信ライブラリ.....	61
	Windows デバイス用の Mobile Link Listener オプション.....	62
	Windows デバイス用の Mobile Link Listener キーワード.....	81
	Windows デバイス用の Mobile Link Listener アクションコマンド.....	83
	Windows デバイス用の Mobile Link Listener action 変数.....	87
1.5	ライトウェイトポーリング API.....	89
	MLLightPoller クラス.....	90
	MLLPCreatePoller メソッド.....	94
	MLLPDestroyPoller メソッド.....	95
1.6	サーバ起動同期のシステムプロシージャ.....	96
	ml_delete_device システムプロシージャ.....	97
	ml_delete_device_address システムプロシージャ.....	97

	ml_delete_listening システムプロシージャ	98
	ml_set_device システムプロシージャ	99
	ml_set_device_address システムプロシージャ	100
	ml_set_listening システムプロシージャ	102
	ml_set_sis_sync_state システムプロシージャ	103
1.7	高度: メッセージ構文	104
	高度: sa_send_udp システムプロシージャを使用した Push 通知の送信	105
1.8	サーバ起動同期チュートリアル	106
	チュートリアル: ライトウェイトポーリングを使用したサーバ起動同期の設定	106
	チュートリアル: ゲートウェイを使用したサーバ起動同期の設定	120
1.9	このマニュアルの印刷、再生、および再配布	139

1 Mobile Link - サーバ起動同期

このマニュアルでは、Mobile Link のサーバ起動同期について説明します。サーバ起動同期とは、Mobile Link サーバから同期の開始またはリモートデバイス上でのアクションの実行を可能にする機能です。

このセクションの内容:

[サーバ起動同期 \[4 ページ\]](#)

Mobile Link サーバ起動同期を使用すると、統合データベースから同期を開始できます。リモートデータベースに Push 通知を送信し、リモートデータベースから統合データベースを更新できます。

[サーバ起動同期の設定 \[8 ページ\]](#)

サーバ起動同期の設定には、Push 要求、Mobile Link Notifier、Mobile Link Listener、ライトウェイトポーラー、およびゲートウェイや Carrier の設定が含まれます。

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

サーバ側設定は、Notifier プロパティ、ゲートウェイプロパティ、Carrier プロパティ、Notifier イベントで構成されません。

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

Mobile Link Listener ユーティリティにより、Windows デバイスは Push 通知を受領し、アクションを開始できます。

[ライトウェイトポーリング API \[89 ページ\]](#)

ライトウェイトポーリング API は、ご使用のデバイスアプリケーションに統合できるプログラミングインターフェースです。ライトウェイトポーリング API には、サーバのポーリングに必要なメソッドが含まれています。

[サーバ起動同期のシステムプロシージャ \[96 ページ\]](#)

サーバ起動同期のシステムプロシージャは、Mobile Link システムテーブル内のローの追加と削除を実行します。

[高度: メッセージ構文 \[104 ページ\]](#)

ライトウェイトポーリング (デフォルト)、UDP ゲートウェイ、SYNC ゲートウェイには、次のメッセージ構文が適用されません。

[サーバ起動同期チュートリアル \[106 ページ\]](#)

サーバ起動同期の使用方法についての理解を深めるには、次のチュートリアルを使用してください。

[このマニュアルの印刷、再生、および再配布 \[139 ページ\]](#)

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

1.1 サーバ起動同期

Mobile Link サーバ起動同期を使用すると、統合データベースから同期を開始できます。リモートデータベースに Push 通知を送信し、リモートデータベースから統合データベースを更新できます。

この Mobile Link コンポーネントには、統合データベースで発生した変更の検出による同期の開始、Push 通知を送信するデバイスの選択、その Push 通知に対するデバイスの応答方法の決定を行うためのプログラム可能なオプションが用意されています。

i 注記

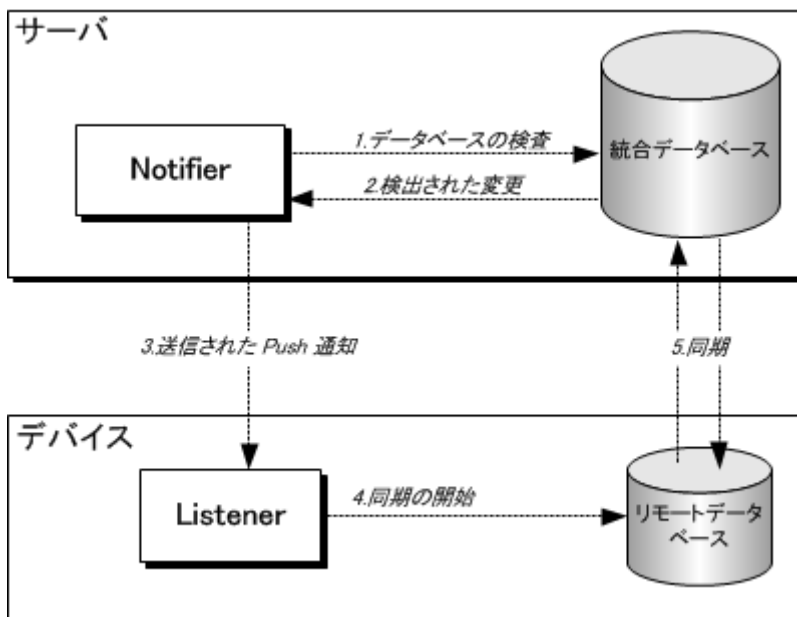
SQL Central を使用してリモートデータベースを管理し、その後、サーバ起動同期への代替機能としてサーバ起動リモートタスク (SIRT) を使用できます。

例

トラック運送会社がモバイルデバイスを自社の運転手に支給するとします。各デバイスではデータベースが実行されており、このデータベースには経路と配達場所が格納されています。道路が渋滞しているという情報がある運転手が送信すると、このレポートは統合データベースに送信されます。Notifier というサーバ側 Mobile Link コンポーネントによってレポートが検出され、渋滞の影響を受ける経路にいる他の運転手に Push 通知が送信されます。この Push 通知の結果、リモートデータベースが同期されるため、運転手は別の経路を使用できます。

サーバ起動同期のプロセス

次の図では、Notifier が統合データベースをチェックし、変更があるかどうかを確認しています。Notifier によって Push 通知がデバイスに送信され、その結果、リモートデータベースが統合データベースと同期されます。



サーバ起動同期プロセスでは、次の手順が実行されます。

1. Notifier はビジネスロジックに基づいたクエリを使用して統合データベースをチェックし、リモートデータベースとの同期が必要な変更があるかどうかを確認します。
2. 変更を検出すると、Notifier はデバイスに Push 通知を送信する準備を行います。
3. Notifier は Push 通知を送信します。Push 通知は、Device Tracker、UDP、SMTP、または SYNC ゲートウェイを使用して送信できます。
4. Listener は、件名、内容、または送信元をメッセージフィルタと照らし合わせて比較します。

5. フィルタ条件が満たされると、アクションが開始されます。たとえば、標準的な実装では、アクションによって Mobile Link クライアントを実行したり、Ultra Light アプリケーションを起動したりできます。

このセクションの内容:

[サーバ起動同期のコンポーネント \[6 ページ\]](#)

Mobile Link サーバ起動同期には、Push 要求、Mobile Link Notifier、Mobile Link Listener、ライトウェイトポーラー、およびゲートウェイが必要です。

[サーバ起動同期の配備に関する考慮事項 \[7 ページ\]](#)

サーバ起動同期を行うアプリケーションを配備する前に考慮すべきいくつかの事項について説明します。

[サーバ起動同期のクイックスタート \[8 ページ\]](#)

サーバ起動同期を設定する為に必要な手順を次に示します。

1.1.1 サーバ起動同期のコンポーネント

Mobile Link サーバ起動同期には、Push 要求、Mobile Link Notifier、Mobile Link Listener、ライトウェイトポーラー、およびゲートウェイが必要です。

Push 要求

Push 要求は結果セット内の値のローで、デバイスに Push 通知を送信するよう Notifier に指示します。Push 要求によって、サーバ起動同期が実行されます。Notifier などの、あらゆるデータベースアプリケーションで Push 要求を作成できます。たとえば、価格が変更されたときにアクティブになるデータベーストリガを使用して、Push 要求を作成できます。

Mobile Link Notifier

Notifier は、Mobile Link サーバに統合されたプログラムです。統合データベースを頻繁にチェックして、Push 要求があるかどうかを確認します。Notifier が Push 要求をチェックする頻度を制御するには、Notifier のプロパティを指定します。Push 要求をチェックするビジネスロジックと、通知対象のデバイスを決定するビジネスロジックを指定する必要があります。Notifier が Push 要求を検出すると、デバイスに Push 通知が送信されます。

Mobile Link Listener

Listener は、デバイス上で実行される 1 つのプログラムです。Notifier から Push 通知を受信すると、メッセージハンドラを使用してメッセージをフィルタし、アクションを開始します。一般的なアプリケーションのアクションは同期の呼び出しですが、アプリケーションから他のアクションも実行できます。選択したサーバソースからの Push 通知や特定の内容が含まれる Push 通知に対して、異なる動作をするように Mobile Link Listener を設定できます。

Windows デバイスでは、Mobile Link Listener はコマンドラインオプションを使用して設定する実行プログラムです。Push 通知を受信するには、デバイスの電源がオンになっていて、Mobile Link Listener が動作中である必要があります。

ライトウェイトポーラー

ライトウェイトポーラーは、指定された時間間隔で Push 通知をポーリングするデバイスアプリケーションです。ライトウェイトポーラーを使用すると、ゲートウェイを設定する代替手段となります。また、サーバへの永続的な接続を必要とせず、バッテリーの寿命を伸ばすことができるため、ライトウェイトポーラーを使用することをお奨めします。

Mobile Link Listener は、Mobile Link Listener コマンドラインオプションを使用して設定できるライトウェイトポーラーです。別の方法として、ライトウェイトポーリング API を使用して、独自のライトウェイトポーラーを作成できます。

ゲートウェイ (ライトウェイトポーラーの代替手段)

ゲートウェイでは、デバイスに Push 通知を送信するための Notifier インタフェースを提供します。ゲートウェイは、ライトウェイトポーラーの代替となる手段です。デバイストラッキングゲートウェイ、SYNC ゲートウェイ、UDP ゲートウェイ、または SMTP ゲートウェイを使用して、メッセージを送信できます。

i 注記

SQL Central を使用してリモートデータベースを管理し、その後、サーバ起動同期への代替機能としてサーバ起動リモートタスク (SIRT) を使用できます。

関連情報

[Push 要求 \[9 ページ\]](#)

[Notifier \[15 ページ\]](#)

[ゲートウェイと Carrier \[26 ページ\]](#)

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

[ライトウェイトポーリングオプションの設定 \[22 ページ\]](#)

[ライトウェイトポーリング API \[89 ページ\]](#)

1.1.2 サーバ起動同期の配備に関する考慮事項

サーバ起動同期を行うアプリケーションを配備する前に考慮すべきいくつかの事項について説明します。

UDP ゲートウェイを使用する場合のデバイスの制限事項

- デバイスの IP アドレスには、Mobile Link サーバから直接アクセスできる必要があります。
- Windows デバイスの IP アドレスに Mobile Link サーバから直接アクセスできない場合、UDP 通知の IP 追跡は機能しません。

デバイストラッキングの制限事項

SQL Anywhere 9.0.0 または以前の Mobile Link Listener では、デバイストラッキングをサポートしていません。これらの Mobile Link Listener でデバイストラッキングを使用するには、デバイストラッキングを手動で設定する必要があります。

サポートされるデバイスプラットフォーム

Mobile Link Listener は、Windows と Windows Mobile でサポートされています。

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

1.1.3 サーバ起動同期のクイックスタート

サーバ起動同期を設定する為に必要な手順を次に示します。

これらの手順を完了する前に、通常の同期用の Mobile Link を設定する必要があります。

1. Mobile Link サーバで、Push 要求を格納するための統合データベースを準備します。
2. Mobile Link サーバで、Push 要求を作成および管理する Notifier イベントを設定します。
3. デバイスで、ライトウェイトポーラーを設定します。
ライトウェイトポーラーを使用しない場合は、Mobile Link サーバでサポートされているゲートウェイを設定します。SMTP ゲートウェイを使用する場合は、Carrier も設定する必要があります。
4. デバイスで、メッセージをフィルタしてアクションを実行する Mobile Link Listener を設定します。

その他のリソース

- サンプルアプリケーションが %SQLANY%SAMP17%\¥MobiLink¥ ディレクトリにインストールされています。サーバ起動同期に関連するすべてのアプリケーションは、SIS_ プレフィックスの付いたディレクトリに配置されています。

関連情報

[Notifier イベントとプロパティの設定 \[16 ページ\]](#)

[ライトウェイトポーラー \[25 ページ\]](#)

[ゲートウェイと Carrier \[26 ページ\]](#)

[メッセージハンドラ \[18 ページ\]](#)

[Push 要求の要件 \[9 ページ\]](#)

1.2 サーバ起動同期の設定

サーバ起動同期の設定には、Push 要求、Mobile Link Notifier、Mobile Link Listener、ライトウェイトポーラー、およびゲートウェイや Carrier の設定が含まれます。

このセクションの内容:

[Push 要求 \[9 ページ\]](#)

Push 要求は、Notifier が Push 通知をデバイスに送信する必要があるかどうかを確認する場合にチェックする、結果セット内の値のローです。

[Notifier \[15 ページ\]](#)

Notifier は Mobile Link サーバに統合されたプログラムで、統合データベースで Push 要求を頻繁にチェックします。Push 要求が検出されると、デバイスに Push 通知を送信します。

[Listener \[17 ページ\]](#)

Listener は、デバイス上で実行される 1 つのプログラムです。Listener は、Notifier からの Push 通知を受信して、アクションを開始します。サーバ起動同期にゲートウェイを使用している場合、Listener は、デバイストラッキング情報を統合データベースにアップロードできます。

[ライトウェイトポーラー \[25 ページ\]](#)

ライトウェイトポーラーは、指定された時間間隔で Push 通知をポーリングするデバイスアプリケーションです。

[ゲートウェイと Carrier \[26 ページ\]](#)

ゲートウェイと Carrier は、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトです。ゲートウェイにはサーバ起動同期に対するメッセージ送信方法が、Carrier にはサーバ起動同期で使用される通信業者に関する情報が含まれます。

1.2.1 Push 要求

Push 要求は、Notifier が Push 通知をデバイスに送信する必要があるかどうかを確認する場合にチェックする、結果セット内の値のローです。

Notifier は Push 通知内に Push 要求を配置して、Push 通知を送信します。典型的なサーバ起動同期設定では、Push 要求にメッセージの内容とターゲットデバイスの情報が含まれます。Push 通知を送信するには、まず、Notifier で Push 要求を検出できるように Notifier イベントを設定する必要があります。

このセクションの内容:

[Push 要求の要件 \[9 ページ\]](#)

Push 要求の要件は、Mobile Link サーバがデバイスとの通信に使用している方法によって異なります。すべての Push 要求に、subject カラムと content カラムが必要となります。

[Push 要求の使用方法 \[12 ページ\]](#)

Push 要求を生成するには、サーバ起動同期に必要な Push 要求のカラムが対象となるデータベースに含まれている必要があります。また、単一のデータベースクエリを使用して値を取得できる必要もあります。

1.2.1.1 Push 要求の要件

Push 要求の要件は、Mobile Link サーバがデバイスとの通信に使用している方法によって異なります。すべての Push 要求に、subject カラムと content カラムが必要となります。

ライトウェイトポーラーを使用して Push 通知をポーリングする場合は、poll key カラムを作成して Push 通知を識別できるようにします。

ゲートウェイを使用して Push 通知を送信する場合は、gateway カラムと address カラムを作成する必要があります。

システムに Push 要求カラムがある場合は、カラムを作成する必要はありません。Push 要求の要件が満たされると、Push 要求を使用できます。

ライトウェイトポーラーを使用する場合の Push 要求の要件 (推奨)

ライトウェイトポーラーを使用して Push 通知をポーリングする場合は、次のカラムを作成します。

カラム	タイプ	説明
Poll key	VARCHAR	ライトウェイトポーラーを識別するために使用するキー。各ライトウェイトポーラーはユニークなキーを送信して、Mobile Link サーバ上で自身を識別します。
Subject	VARCHAR	メッセージの件名の行です。
Content	VARCHAR	メッセージの内容。

ゲートウェイを使用する場合の Push 要求の要件 (推奨)

特に指定がないかぎり、ゲートウェイを使用して Push 通知を送信する場合は、次のカラムを作成してください。

カラム	タイプ	説明
Request ID	INTEGER	省略可能です。Push 要求のユニークな ID。一部の Notifier イベントでは、このカラム名が必須です。
Gateway	VARCHAR	メッセージの送信先ゲートウェイの名前。
Subject	VARCHAR	メッセージの件名の行です。
Content	VARCHAR	メッセージの内容。
Address	VARCHAR	デバイスの送信先アドレス。

カラム	タイプ	説明
Resend interval	VARCHAR	省略可能です。メッセージが再送される時間間隔。 resend interval は、信頼性の低いネットワークで UDP ゲートウェイを使用する場合に便利です。Notifier は、Push 要求に関連付けられたすべての属性が変更されないことを前提としています。要求を最初にポーリングした後、後続の更新は無視されます。次のポーリング時刻の前に Push 通知を送信する必要がある場合、Notifier は次のポーリング間隔を自動的に調整します。Push 要求の送信を停止するには、request_cursor イベントで同期ロジックを使用します。対象 Mobile Link Listener から受信確認が届くと、次の再送は停止されます。
Time to live	VARCHAR	省略可能です。再送の有効期限が切れるまでの時間です。

例

次の例では、SQL Anywhere 統合データベースのテーブルに必要なカラムを作成して、ライトウェイトポーリングを使用する場合の Push 要求の要件を満たします。

```
CREATE TABLE PushRequest (
  req_id INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,
  poll_key VARCHAR(128),
  subject VARCHAR(128),
  content VARCHAR(128)
)
```

このようなテーブルの作成が必要になるのは、Push 要求のカラムが他の場所で使用できない場合のみです。Push 要求のカラムは、複数のテーブル間、既存の複数のテーブル、または 1 つのビューに作成できます。

関連情報

[Notifier イベント \[39 ページ\]](#)

[Notifier イベントとプロパティの設定 \[16 ページ\]](#)

[Push 要求の使用法 \[12 ページ\]](#)

[request_cursor イベント \[44 ページ\]](#)

1.2.1.2 Push 要求の使用方法

Push 要求を生成するには、サーバ起動同期に必要な Push 要求のカラムが対象となるデータベースに含まれている必要があります。また、単一のデータベースクエリを使用して値を取得できる必要もあります。

Push 要求は、Push 要求のカラムを選択する request_cursor イベントでデータベースクエリを指定すると、自動的に生成されます。

Push 要求の制限事項

次の表は、Push 要求の制限事項をカラムごとに示します。

カラム	Type	制限
Request ID	INTEGER	この値は、ユニークなプライマリキーにしてください。
Poll key	VARCHAR	ライトウェイトポーラーの使用時にのみ必要です。 ポーリングキーには制限がありません。
Gateway	VARCHAR	ゲートウェイを使用する場合にのみ必要です。 この値には、有効なゲートウェイの名前を設定してください。独自のカスタムゲートウェイ名を指定するか、または次の事前に設定されたゲートウェイ名のいずれかを選択します。 <ul style="list-style-type: none">• Default-DeviceTracker• Default-SMTP• Default-SYNC• Default-UDP
Subject	VARCHAR	この値の設定では、英数字以外の文字を使用しないでください。中カッコ、カレット、二重引用符、一重引用符、角カッコは内部用に予約されているため、subject カラムで使用しないでください。
Content	VARCHAR	メッセージの内容には制限がありません。

カラム	Type	制限
Address	VARCHAR	<p>ゲートウェイを使用する場合にのみ必要です。</p> <p>UDP ゲートウェイの場合、この値は IP アドレスまたはホスト名にしてください。次のフォーマットのポート番号のサフィックスがサポートされています。</p> <ul style="list-style-type: none"> IP-address:port-number hostname:port-number <p>SMTP ゲートウェイの場合、この値は電子メールアドレスにしてください。</p> <p>SYNC ゲートウェイとデバイストラッキングゲートウェイの場合、この値は Mobile Link Listener -t+ オプションで定義されている受信者名にしてください。</p>
Resend interval	VARCHAR	<p>デフォルトでは、この値は分単位で測定されます。秒、分、時間それぞれの単位として <i>S</i>、<i>M</i>、および <i>H</i> を指定できます。また、単位を組み合わせることもできます。たとえば 1H 30M 10S は、メッセージを 1 時間ごと、30 分ごと、10 秒ごとに再送するよう Notifier に通知します。</p> <p>この値が NULL または未指定の場合、デフォルトでは一度だけ送信され、再送は行われません。</p>
Time to live	VARCHAR	<p>デフォルトでは、この値は分単位で測定されます。秒、分、時間それぞれの単位として <i>S</i>、<i>M</i>、および <i>H</i> を指定できます。また、単位を組み合わせることもできます。たとえば、3H 30M 10S は、メッセージの最初の送信の 3 時間後、30 分後、10 秒後に再送を停止するよう Notifier に通知します。</p> <p>この値が NULL または未指定の場合、デフォルトでは一度だけ送信され、再送は行われません。</p>

Push 要求の検出と Push 通知の送信

Notifier では、request_cursor イベントを頻繁に起動することによって、Push 要求を検出します。デフォルトでは、このイベントにはスクリプトが指定されていません。Notifier が Push 要求を検出できるよう、request_cursor イベントを指定してください。典型的なアプリケーションでは、request_cursor イベントスクリプトは SELECT 文です。

次の例では、ml_add_property システムプロシージャを使用して、Simple という名前のカスタム Notifier 用の request_cursor イベントスクリプトを作成します。SELECT 文は、テーブル PushRequest から Push 要求を検出するよう Notifier に通知します。

```
CALL ml_add_property('SIS', 'Notifier(Simple)', 'request_cursor',  
  'SELECT poll_key, subject, content FROM PushRequest'  
);
```

i 注記

カラムは、Push 要求で指定されている順序と同じ順序で選択してください。

Push 要求の削除

Push 通知がビジネス規則を満たし、この規則に従って送信された後に通知されたデバイスの情報が更新されない場合、Notifier は通知を再送します。Push 要求が満たされたら、Notifier で古い Push 要求が検出されないようにする必要があります。同期目的で Push 通知が送信された場合は、同期スクリプトを使用して Push 要求を削除できます。

request_delete イベントを使用して要求 ID ごとに Push 要求を削除できますが、Push 要求に request ID カラムが含まれている必要があります。また、配信確認を有効にしてください。

例

ライトウェイトポーラーに対する Push 要求の生成例 - Mobile Link サーバが unique_device_ID として検出したリモートデバイスがあり、統合データベースに次の SQL 文を使用して作成された PushRequest という名前のテーブルが含まれるとします。

```
CREATE TABLE PushRequest (  
  req_id INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,  
  poll_key VARCHAR(128),  
  subject VARCHAR(128),  
  content VARCHAR(128)  
);
```

この例では、Push 要求の準備に、統合データベースで次の SQL 文を実行します。

```
INSERT INTO PushRequest (poll_key, subject, content) VALUES ('unique_device_ID',  
'synchronize', 'ASAP');
```

上記のスクリプトを使用して PushRequest テーブルに値を挿入しても、Push 要求自体は生成されません。Mobile Link サーバの request_cursor イベントでデータベースクエリを設定して、挿入する値を選択し、Push 要求を生成できるようにします。

この例では、Mobile Link サーバの request_cursor イベントスクリプトで次の SQL 文を定義します。

```
SELECT poll_key, subject, content FROM PushRequest;
```

これで unique_device_ID デバイスがサーバに対して Push 通知をポーリングし、request_cursor イベントが PushRequest テーブルでデータを検出すると、Push 要求が生成されます。デバイスに送信されると、Push 通知の件名は *synchronize* と定義され、内容は *ASAP* と定義されます。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[ライトウェイトポーラーの代替手段としてのゲートウェイ \[26 ページ\]](#)

[チュートリアル: ライトウェイトポーリングを使用したサーバ起動同期の設定 \[106 ページ\]](#)

[チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

[Push 要求の要件 \[9 ページ\]](#)

[request_cursor イベント \[44 ページ\]](#)

[request_delete イベント \[46 ページ\]](#)

[-t dblsn オプション \[76 ページ\]](#)

1.2.2 Notifier

Notifier は Mobile Link サーバに統合されたプログラムで、統合データベースで Push 要求を頻繁にチェックします。Push 要求が検出されると、デバイスに Push 通知を送信します。

また、Notifier は一連のイベントを実行して、データのモニタ、Push 要求の管理、配信確認の処理、エラーの処理を行うスクリプトを作成できるようにします。

Mobile Link サーバを最初にロードすると、Notifier が起動します。Mobile Link サーバの単一インスタンス内で複数の Notifier を実行できます。複数の Notifier の使用方法の例については、`%SQLANYSAMP17%¥MobiLink`
`¥SIS_MultipleNotifier`にあるサンプルアプリケーションを参照してください。

データベースへの接続が失われると、Notifier は再びアクセスできるまで接続のリカバリを試みます。リカバリ後、Notifier は引き続き同じ設定で動作します。

このセクションの内容:

[Mobile Link サーバファームでの Notifier \[16 ページ\]](#)

Notifier は、ファーム内のすべての Mobile Link サーバで実行できます。これらの Notifier によって、同じ Mobile Link Listener への冗長な Push 通知がなくなります。

[Notifier イベントとプロパティの設定 \[16 ページ\]](#)

Notifier イベントを使用することにより、サーバ起動同期処理全体を管理するスクリプトを埋め込むことができます。

[Notifier の起動 \[17 ページ\]](#)

Mobile Link サーバをロードすると、有効な Notifier がすべて起動します。Notifier を無効にするには、有効な Notifier のプロパティ値を false に設定してください。

1.2.2.1 Mobile Link サーバファームでの Notifier

Notifier は、ファーム内のすべての Mobile Link サーバで実行できます。これらの Notifier によって、同じ Mobile Link Listener への冗長な Push 通知がなくなります。

ローカルの Mobile Link サーバに接続する必要があるときは、mlsrv17 -lsc サーバオプションを使用して、他のサーバに情報を渡すことができます。

この機能を使用すると、1つの Notifier がプライマリ、他のすべての Notifier がセカンダリになります。プライマリ Notifier は、直接、またはセカンダリを通じて間接的に、Push 通知を制御します。セカンダリ Notifier も、Mobile Link Listener 情報をプライマリ Notifier にルート指定するので、Mobile Link Listener の場所と、Mobile Link Listener への経路を認識しています。

プライマリ Notifier を実行している Mobile Link サーバに障害が発生した場合、サーバファームは新しいプライマリ Notifier を選択し、通知を継続します。

Mobile Link Listener は、どれがプライマリサーバかを知らなくても、ファーム内のどの Mobile Link サーバにも接続できます。

この機能を使用するには、ファーム内のすべての Mobile Link サーバに次の mlsrv17 コマンドラインオプションが必要です。

- -lsc
- -notifier
- -zs

例

host001 の場合:

```
mlsrv17 -notifier -zs ml001 -lsc tcpip(host=host001;port=2439) ...
```

host007 の場合:

```
mlsrv17 -notifier -zs ml007 -lsc tcpip(host=host007;port=2439) ...
```

1.2.2.2 Notifier イベントとプロパティの設定

Notifier イベントを使用することにより、サーバ起動同期処理全体を管理するスクリプトを埋め込むことができます。

たとえば、次のようなタスクを実行する Notifier イベントを設定できます。

- request_cursor イベントを使用して、Push 要求で送信する情報、送信方法、送信先を決定します。
- begin_poll イベントを使用して、統合データベースの変化に応じて Push 要求を作成する (高度な使用法)。
- request_delete イベントを使用して、Push 要求を削除する (要求に応じて)。
- end_poll イベントを使用して、Notifier のポーリングを追跡し、テーブルデータをクリーンアップする (高度な使用法)。

Notifier プロパティはイベントに似ています。イベントは通知プロセスを管理しますが、プロパティは Notifier の動作を管理します。たとえば、Notifier プロパティでは、Notifier が統合データベースをポーリングする頻度や起動時に Notifier を有効にするかどうかを決定します。Notifier プロパティおよびイベントは、サーバ側の設定として設定します。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Notifier イベント \[39 ページ\]](#)

1.2.2.3 Notifier の起動

Mobile Link サーバをロードすると、有効な Notifier がすべて起動します。Notifier を無効にするには、有効な Notifier のプロパティ値を false に設定してください。

Notifier を起動するには、次のいずれかの方法を使用します。

- Notifier を設定し、指定した `-notifier` オプションで `mlsrv17` を実行します。
- 設定が Notifier 設定ファイルに格納されている場合は、コマンドラインで `mlsrv17` を実行してデータベースをロードし、`-notifier` オプションを使用してファイルを指定します。たとえば、ファイル `myfirst.Notifier` を使用する場合は、次のコマンドによって、このファイルに指定されているプロパティおよびイベントを使用するよう Mobile Link サーバを設定します。

```
mlsrv17 ... -notifier c:¥myfirst.Notifier
```

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Notifier イベントとプロパティの設定 \[16 ページ\]](#)

[Notifier プロパティ \[52 ページ\]](#)

1.2.3 Listener

Listener は、デバイス上で実行される 1 つのプログラムです。Listener は、Notifier からの Push 通知を受信して、アクションを開始します。サーバ起動同期にゲートウェイを使用している場合、Listener は、デバイストラッキング情報を統合データベースにアップロードできます。

例

次のコマンドは、Windows デバイスの Mobile Link Listener ユーティリティを起動します。

```
dblsn -v2 -m -ot dblsn.log
-l "poll_connect='host=localhost';
poll_notifier=notifier_name1;
poll_key=sis_user1;
poll_every=10;
subject=sync;
action='start dbmlsync.exe
-c SERVER=rem1;UID=DBA;PWD=passwd
```

```
-ot dbmlsyncOut.txt -qc';"
```

このコマンドは、冗長性レベルが 2 に設定された Mobile Link Listener をロードし、メッセージのログギングを有効にして、サーバが *localhost* にあることを指定します。dblsn.log ファイルは、出力が書き込まれる前にトランケートされます。Mobile Link Listener は、10 秒ごとに Push 通知をポーリングします。Mobile Link Listener が **sync** という件名の Push 通知を受信すると、Mobile Link クライアントアプリケーションが起動します。

このセクションの内容:

[メッセージハンドラ \[18 ページ\]](#)

メッセージハンドラは Mobile Link Listener コンポーネントで、Push 通知のメッセージの内容をスキャンしてアクションを開始します。また、メッセージハンドラを使用すると、サーバ検索やポーリング頻度などのライトウェイトポーリングオプションを指定できます。

関連情報

[デバイストラッキングゲートウェイ \[27 ページ\]](#)

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

[Windows デバイス用の Mobile Link Listener オプション \[62 ページ\]](#)

1.2.3.1 メッセージハンドラ

メッセージハンドラは Mobile Link Listener コンポーネントで、Push 通知のメッセージの内容をスキャンしてアクションを開始します。また、メッセージハンドラを使用すると、サーバ検索やポーリング頻度などのライトウェイトポーリングオプションを指定できます。

メッセージハンドラは、次のコンポーネントから構成されています。

フィルタのキーワード

Push 通知が前処理されると、フィルタのキーワードを使用してメッセージの内容をスキャンできます。フィルタ条件が満たされると、アクションが開始されます。たとえば、*subject* キーワードを指定して特定の件名を含むメッセージをフィルタしたり、*sender* キーワードを指定して特定の Mobile Link サーバから受信したメッセージをフィルタしたりできます。

アクション

メッセージでフィルタ条件が満たされると、アクションが開始されます。典型的なアプリケーションでは、アクションを指定して同期を開始しますが、別の操作も実行できます。エラー処理の支援として、元のアクションが失敗した場合にインスタンスを処理できるよう代替アクションを指定します。

ポーリング設定

ポーリング設定では、Mobile Link Listener が Mobile Link サーバで Push 通知をポーリングする方法を設定できます。

オプション

オプションを使用すると、配信確認やアクション確認などのリモート設定を制御できます。

メッセージハンドラは、dblsn -l オプションを使用して作成できます。複数のメッセージハンドラを指定できます。

このセクションの内容:

[メッセージフィルタ \[19 ページ\]](#)

Mobile Link Listener が Push 通知を受信すると、Push 通知はメッセージを抽出します。メッセージは複数のキーワードに分割されています。

[高度: フィルタとしてのリモート ID \[23 ページ\]](#)

リモート ID によってメッセージをフィルタリングするには、`-r` オプションと `$remote_id` action 変数を使用します。

[高度: 接続起動同期 \[24 ページ\]](#)

Windows デバイスでは、接続の変更時に同期を開始できます。

関連情報

[-l dblsn オプション \[69 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

1.2.3.1.1 メッセージフィルタ

Mobile Link Listener が Push 通知を受信すると、Push 通知はメッセージを抽出します。メッセージは複数のキーワードに分割されています。

`message` キーワードには、メッセージ全体が未加工形式で記述されています。このメッセージは、`subject` キーワード、`content` キーワード、`sender` キーワードに分割されます。これらのキーワードは、メッセージフィルタを介して実行され、開始するアクションを決定します。

フィルタキーワードを使用して、Push 通知の一部とユーザ定義のフレーズを比較します。2 つのフレーズのテキストが同等の場合、アクションが開始されます。

フィルタキーワードを指定するには、次の構文を使用して Mobile Link Listener を実行します。

```
dblsn ... -l "filter-keyword-name='content to filter';action='...'"
```

`-l` オプションを複数回使用すると複数のファイルを作成できますが、各 `-l` インスタンスのアクションも指定してください。アクションは、すべてのフィルタが満たされた場合にのみ開始されます。

次の各キーワードは、メッセージハンドラに 1 回のみ表示されます。

content

メッセージのフィルタリングには、このキーワードと `subject` キーワードを使用することをお奨めします。このキーワードは、内容に基づいてメッセージをフィルタリングするために使用します。次に例を示します。

```
dblsn -l "content='your content filter here';action='...'"
```

subject

メッセージのフィルタリングには、このキーワードと content キーワードを使用することをお奨めします。このキーワードは、件名に基づいてメッセージをフィルタリングするために使用します。次に例を示します。

```
dblsn -l "subject='your subject filter here';action='...'"
```

message

このキーワードは、未加工データに基づいてメッセージをフィルタリングするために使用します。フィルタ値がメッセージの正確な長さとも一致するようにしてください。このキーワードには変数構造があるため、使用しないことをお奨めします。

message_start

このキーワードは、未加工データの先頭からの一部に基づいてメッセージをフィルタリングするために使用します。

このキーワードを指定すると、Mobile Link Listener は action 変数の \$message_start と \$message_end を作成します。

sender

このキーワードは、送信者に基づいてメッセージをフィルタリングするために使用します。このキーワードは、特定の Notifier が送信した Push 通知を追跡するのに役立ちます。この値は、使用されているゲートウェイによって異なります。UDP ゲートウェイの場合、この値はゲートウェイのホストの IP アドレスです。SYNC ゲートウェイの場合は、[MobiLink](#) です。また、SMTP ゲートウェイの場合は、ご使用の無線通信事業者によって異なります。

このセクションの内容:

[アクションの開始 \[20 ページ\]](#)

メッセージがフィルタの条件を満たしている場合に、アクションが開始されます。

[action 変数 \[21 ページ\]](#)

action 変数を使用すると、メッセージフィルタまたはアクションからの Push 通知の一部を参照できます。

[ライトウェイトポーリングオプションの設定 \[22 ページ\]](#)

メッセージハンドラを使用して、ポーリングを処理できます。ライトウェイトポーリングオプションを使用すると、サーバのロケーション、Notifier 名、ポーリング頻度、ポーリングキーを指定できます。または、ライトウェイトポーリング API を使用してこれらのプロパティを指定することもできます。

関連情報

[ゲートウェイと Carrier \[26 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[高度: メッセージ構文 \[104 ページ\]](#)

1.2.3.1.1.1 アクションの開始

メッセージがフィルタの条件を満たしている場合に、アクションが開始されます。

アクションを指定するには、次の構文を使用して Mobile Link Listener を実行します。

```
dblsn ... -l "...;action='action-command command statement'"
```

次のアクションコマンドを使用すると、メッセージのフィルタリング時にさまざまなタスクを実行できます。

START

アプリケーションを開始し、バックグラウンドで実行されるようにします。

RUN

アプリケーションを実行し、追加の Push 通知を受信する前にアプリケーションの完了を待機します。

POST

すでに実行中のプロセスにウィンドウメッセージを送信します。このコマンドは、Windows デバイスでしか使用できません。

SOCKET

TCP/IP 接続を使用して、アプリケーションにメッセージを送信します。

DBLSN FULL SHUTDOWN

Mobile Link Listener を停止します。

関連情報

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

[action 変数 \[21 ページ\]](#)

1.2.3.1.1.2 action 変数

action 変数を使用すると、メッセージフィルタまたはアクションからの Push 通知の一部を参照できます。

action 変数の設定方法

ほとんどの action 変数は、Push 通知が受信されるたびに自動的に設定されます。変数名は、メッセージ構文で指定されている名前と似ています。たとえば、`message` は `$message` action 変数を、`subject` は `$subject` action 変数を、`sender` は `$sender` action 変数を、`content` は `$content` action 変数をそれぞれ設定します。

action 変数の使用

action 変数は、Mobile Link Listener の実行時にコマンドラインで使用します。使用方法は、メッセージハンドラと開始するアクションによって異なります。次の例は、Mobile Link クライアントアプリケーションの起動に使用する RUN アクションコマンドの使用例を示します。

```
dblsn ... -l "subject=publish;action='RUN dbmlsync.exe @dbmlsync.txt -n $content'"
```

このメッセージハンドラは、件名のテキストが "publish" と同等の場合にメッセージをフィルタリングします。フィルタリング後に、-n オプションを使用して dbmsync が実行され、パラメータとして \$content action 変数が渡されます。content は同期パブリケーションの名前を参照すると仮定して、dbmsync はパブリケーションを使用して、デバイスデータベースと統合データベースを同期します。

次の例は、action 変数を使用したメッセージのフィルタリングを示します。

```
dblsn ... -l "subject=$content;action='RUN script.bat'"
```

このメッセージハンドラは、*subject* のテキストが *content* と同等の場合にメッセージをフィルタリングします。フィルタリング後に、デバイスはカスタムバッチスクリプトを実行します。

関連情報

[アクションの開始 \[20 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

[Windows デバイス用の Mobile Link Listener action 変数 \[87 ページ\]](#)

[高度: フィルタとしてのリモート ID \[23 ページ\]](#)

[高度: メッセージ構文 \[104 ページ\]](#)

1.2.3.1.1.3 ライトウェイトポーリングオプションの設定

メッセージハンドラを使用して、ポーリングを処理できます。ライトウェイトポーリングオプションを使用すると、サーバのロケーション、Notifier 名、ポーリング頻度、ポーリングキーを指定できます。または、ライトウェイトポーリング API を使用してこれらのプロパティを指定することもできます。

ライトウェイトポーリングオプションを指定するには、次の構文を使用して Mobile Link Listener を実行します。

```
dblsn ... -l
  "poll_connect=protocol-options;
  poll_notifier=Notifier-name;
  poll_key=identifier-string;
  poll_every=number-of-seconds;..."
```

1つのメッセージハンドラには、次のオプションのいずれか1つのみを含めることができます。

poll_connect

このオプションは、サーバへの接続に必要なプロトコルオプションの指定に使用します。または、dblsn -x オプションを使用してデフォルトのプロトコルオプションを指定することもできます。poll_connect オプションを使用すると、メッセージハンドラのデフォルトのプロトコルオプションが上書きされます。

poll_notifier

このオプションは、Push 要求を処理するために Mobile Link サーバで使用される Notifier の指定に使用します。Mobile Link サーバでは複数の Notifier を受け入れることができるため、このオプションが必要になります。

poll_key

このオプションは、Notifier に対する Mobile Link Listener の識別に使用します。Mobile Link サーバはこの値を使用して、デバイスを対象とした Push 通知を送信します。典型的なアプリケーションでは、この値をデバイスのリモート ID にしてください。

poll_every

このオプションは、Mobile Link Listener が Notifier をポーリングする頻度の指定に使用します。デフォルトでは、Mobile Link Listener は Mobile Link サーバから自動的にこの値を取得します。この値は、秒単位です。

関連情報

[Notifier イベントとプロパティの設定 \[16 ページ\]](#)

[ライトウェイトポーラー \[25 ページ\]](#)

[Push 要求の要件 \[9 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[ライトウェイトポーリング API \[89 ページ\]](#)

1.2.3.1.2 高度: フィルタとしてのリモート ID

リモート ID によってメッセージをフィルタリングするには、`-r` オプションと `$remote_id` action 変数を使用します。

SQL Anywhere リモートデータベースを初めて同期すると、データベースの ID を含むリモート ID ファイルが作成されます。このファイルの名前は、データベースと同じで、拡張子 `.rid` が付き、データベースと同じディレクトリに保存されます。Ultra Light データベースの場合はリモート ID ファイルがなく、リモート ID はデータベースから直接抽出されます。

Mobile Link Listener を起動する場合は、`dblsn -r` オプションを使用してリモート ID ファイルまたは Ultra Light データベースの名前とロケーションを指定し、`dblsn -l` オプションを使用してメッセージハンドラを作成します。

メッセージフィルタには、リモート ID を直接入力できます。ただし、リモート ID はデフォルトでは GUID なので、わかりやすい名前を指定しないと、簡単に覚えることができません。

i 注記

`dblsn` コマンドラインでは、`-r` オプションと `-l` オプションの複数のインスタンスを指定できます。`-l` オプションで使用される `$remote_id` action 変数は、通常、その前の `-r` オプションで指定されています。そのため、`-l` オプションの前に `-r` オプションを指定することが重要です。

次の例は、複数のリモート ID の使用方法を示します。ここでは、`business.db` という SQL Anywhere データベースと `personal.udb` という Ultra Light データベースがデバイス上にあることを前提としています。この例で、`ulpersonal` は Ultra Light アプリケーションのウィンドウクラス名です。

```
dblsn ... -r "c:\app\%db%\business.rid"
         -l "subject=$remote_id;action='dbmlsync.exe -k -c dsn=business';"
         -r "c:\ulapp\%personal.udb"
         -l "subject=$remote_id;action=post dbas_synchronize to ulpersonal;"
```

関連情報

[action 変数 \[21 ページ\]](#)

[-r dblsn オプション \[75 ページ\]](#)

[Windows デバイス用の Mobile Link Listener action 変数 \[87 ページ\]](#)

1.2.3.1.3 高度: 接続起動同期

Windows デバイスでは、接続の変更時に同期を開始できます。

IP 接続が確立されたり、失われたりすると、デバイスはメッセージ `_IP_CHANGED_` を含む Push 通知を Mobile Link Listener に送信します。デバイスは、Mobile Link サーバへの新しい最適パスを見つけると、メッセージ `_BEST_IP_CHANGED_` を含む Push 通知を Mobile Link Listener に送信します。メッセージハンドラを使用すると、接続に関するこれらの変更を検出し、アクションを開始できます。

接続におけるすべての変更の識別

`_IP_CHANGED_` メッセージは、IP 接続が変更されたことを示します。接続の変更は、デバイスが Wi-Fi ネットワークの範囲に入ったり、ユーザが RAS 接続を開始したり、ユーザがデバイスをクレドールに置いたりした場合に発生します。

`_IP_CHANGED_` メッセージを参照するには、次の構文を使用して Mobile Link Listener を実行します。

```
dblsn ... -l "message=_IP_CHANGED_;action='...'"
```

次の例は、`_IP_CHANGED_` メッセージの使用法を示します。メッセージハンドラはメッセージをフィルタリングし、サーバに送信します。接続が失われると、エラーが生成されます。

```
dblsn -l "message=_IP_CHANGED_;
action='
SOCKET port=12345;
sendText=IP changed: $adapters|$network_names;
recvText=beeperAck;
timeout=5';
continue=yes;"
```

Mobile Link サーバへの最適パスの変更の識別

`_BEST_IP_CHANGED_` メッセージは、Mobile Link サーバへの最適パスが変更されたことを示します。このメッセージを参照するには、次の構文を使用して Mobile Link Listener を実行します。

```
dblsn ... -x MobiLink-protocol-options -l "message=_BEST_IP_CHANGED_;action='...'"
```

`_BEST_IP_CHANGED_` メッセージの実行時に、最善の IP 接続を表すローカルの IP アドレスに置き換える `$best_ip` action 変数を使用すると、役立つアクションを開始できます。IP 接続が存在しない場合、`$best_ip` は 0.0.0.0 を返します。

次の例では、`_BEST_IP_CHANGED_` メッセージを使用して、最善の IP 接続が変更されたときに同期を起動しています。接続が失われると、エラーが生成されます。

```
dblsn -x http(host=mlserver.company.com)
-v2 -m -i 3 -ot dblsn.log
-l "message=_BEST_IP_CHANGED_;
  action='
    START dbmlsync.exe -ra -c SERVER=remote;UID=DBA;PWD=sql -n test_pub'"
```

i 注記

ご使用のアプリケーションで接続起動同期をテストする場合は、Mobile Link Listener を Mobile Link サーバとは別のコンピュータで実行します。

関連情報

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

[Windows デバイス用の Mobile Link Listener action 変数 \[87 ページ\]](#)

1.2.4 ライトウェイトポーラー

ライトウェイトポーラーは、指定された時間間隔で Push 通知をポーリングするデバイスアプリケーションです。

ゲートウェイを設定する代わりにライトウェイトポーラーを使用できます。SYNC ゲートウェイとは異なりサーバへの永続的接続を必要とせず、また、UDP ゲートウェイとは異なり連続的な接続を必要としないため、ライトウェイトポーラーの使用をお奨めします。

デバイスは、サーバのポーリング時に、ポーリングキーと Notifier 名を送信します。Mobile Link サーバは、Notifier 名をチェックして、Push 要求のキャッシュをチェックする Notifier を確認します。ポーリングキーは、ポーリングキーを使用してデバイスを対象とした Push 要求を検出する Notifier に対するデバイスを識別します。Push 通知は Push 要求の検出後に送信されます。

Mobile Link Listener コマンドラインオプションを使用して、ライトウェイトポーラーを設定します。または、ライトウェイトポーリング API を使用して、ライトウェイトポーラーをデバイスアプリケーションに統合します。

i 注記

SQL Central を使用してリモートデータベースを管理し、その後、サーバ起動リモートタスク (SIRT) を使用して Push 通知を実装できます。

関連情報

[ライトウェイトポーリング API \[89 ページ\]](#)

[ライトウェイトポーリングオプションの設定 \[22 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

1.2.5 ゲートウェイと Carrier

ゲートウェイと Carrier は、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトです。ゲートウェイにはサーバ起動同期に対するメッセージ送信方法が、Carrier にはサーバ起動同期で使用される通信業者に関する情報が含まれます。

このセクションの内容:

[ライトウェイトポーラーの代替手段としてのゲートウェイ \[26 ページ\]](#)

ゲートウェイは、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトで、システム起動同期用のメッセージの送信方法に関する情報が含まれます。ライトウェイトポーラーの代わりに使用でき、継続したネットワーク接続を必要とします。

[デバイストラッキングゲートウェイ \[27 ページ\]](#)

デバイストラッキングを使用することにより、Mobile Link サーバでは、Push 要求のリモート ID 情報を使用してデバイスを追跡できます。デバイストラッキングゲートウェイは、自動追跡 IP アドレス、電話番号、公衆無線ネットワークプロバイダ ID を利用して SYNC ゲートウェイ、UDP ゲートウェイ、SMTP ゲートウェイを介して Push 通知を配信します。

1.2.5.1 ライトウェイトポーラーの代替手段としてのゲートウェイ

ゲートウェイは、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトで、システム起動同期用のメッセージの送信方法に関する情報が含まれます。ライトウェイトポーラーの代わりに使用でき、継続したネットワーク接続を必要とします。

ゲートウェイのプロパティは、Mobile Link サーバで設定します。1 つの Mobile Link サーバに複数のゲートウェイを設定できます。

サポートされているゲートウェイ

Mobile Link サーバでは、次のゲートウェイがサポートされています。

SYNC ゲートウェイ

SYNC ゲートウェイは TCP/IP ベースのゲートウェイです。Push 通知は、Mobile Link による同期と同じプロトコルを使用して送信されます。

デフォルトの SYNC ゲートウェイの名前は、Default-SYNC です。通常、デフォルトのゲートウェイ設定を変更する必要はありません。

UDP ゲートウェイ

UDP ゲートウェイは、UDP ゲートウェイを介して Push 通知を送信します。

デフォルトの UDP ゲートウェイの名前は、Default-UDP です。通常、デフォルトのゲートウェイ設定を変更する必要はありません。Mobile Link Listener は、Push 通知を受信するときにデフォルトで UDP を使用します。

SMTP ゲートウェイ

SMTP ゲートウェイは、通信業者の電子メールから SMS への変換サービスを使用して Push 通知を送信します。

デフォルトの SMTP ゲートウェイの名前は、Default-SMTP です。

デバイストラッキングゲートウェイ

サポートされているゲートウェイ以外に、Push 通知を送信する最適なゲートウェイを自動的に選択するデバイストラッキングゲートウェイを設定できます。デフォルトのデバイストラッキングゲートウェイは、Default-DeviceTracker です。ライトウェイトポーラーを使用したくない場合、このゲートウェイを使用してください。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[デバイストラッキングゲートウェイ \[27 ページ\]](#)

[SYNC ゲートウェイプロパティ \[57 ページ\]](#)

[UDP ゲートウェイプロパティ \[58 ページ\]](#)

[SMTP ゲートウェイプロパティ \[56 ページ\]](#)

1.2.5.2 デバイストラッキングゲートウェイ

デバイストラッキングを使用することにより、Mobile Link サーバでは、Push 要求のリモート ID 情報を使用してデバイスを追跡できます。デバイストラッキングゲートウェイは、自動追跡 IP アドレス、電話番号、公衆無線ネットワークプロバイダ ID を利用して SYNC ゲートウェイ、UDP ゲートウェイ、SMTP ゲートウェイを介して Push 通知を配信します。

ゲートウェイは、最初に SYNC ゲートウェイを使用してデバイスへの接続を試みます。配信が失敗した場合は、UDP ゲートウェイ、続いて SMTP ゲートウェイが使用されます。この機能は、デバイスのアドレスを変更する場合に便利です。

デバイストラッキングゲートウェイには、最大で 3 つの従属ゲートウェイ (1 つの SYNC と 1 つの SMTP と 1 つの UDP) を持つことができます。Push 通知は、Mobile Link Listener から送信されたデバイストラッキング情報に基づいて、いずれかの従属ゲートウェイへ自動的にルーティングされます。従属ゲートウェイを有効にすると、デバイスアドレスの変更は、Mobile Link サーバによって自動的に管理されます。アドレスが変更されると、Mobile Link Listener は統合データベースと同期して、ml_device_address システムテーブル内のトラッキング情報を更新します。

9.0.1 以降のほとんどの Mobile Link Listener は、デバイストラッキングをサポートしています。デバイストラッキングをサポートしない Mobile Link Listener を使用している場合、トラッキング情報を提供することで、デバイストラッキングゲートウェイを使用することもできます。

このセクションの内容:

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

9.0.0 Mobile Link Listener のデバイストラッキングを手動で設定するためのシステムプロシージャがいくつかあります。これらのシステムプロシージャは、統合データベース上の Mobile Link システムテーブル ml_device、ml_device_address、ml_listening を更新します。

[デバイストラッキングゲートウェイ設定のクイックスタート \[30 ページ\]](#)

次の手順では、デバイストラッキングゲートウェイを設定する方法の概要を説明します。

[Carrier と Carrier 設定 \[31 ページ\]](#)

Carrier は、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトで、サーバ起動同期で使用される通信業者に関する情報が含まれます。

関連情報

[ライトウェイトポーラーの代替手段としてのゲートウェイ \[26 ページ\]](#)

[Carrier と Carrier 設定 \[31 ページ\]](#)

1.2.5.2.1 9.0.0 Mobile Link Listener のデバイストラッキングの設定

9.0.0 Mobile Link Listener のデバイストラッキングを手動で設定するためのシステムプロシージャがいくつかあります。これらのシステムプロシージャは、統合データベース上の Mobile Link システムテーブル ml_device、ml_device_address、ml_listening を更新します。

コンテキスト

SQL Anywhere 9.0.0 以前で実行している Mobile Link Listener を使用している場合にのみ、デバイストラッキングがサポートされている必要があります。その他すべての Windows デバイス用 Mobile Link Listener では、デバイストラッキングがサポートされています。

手動で設定するデバイストラッキングでは、ネットワークアドレス情報を提供しないで Mobile Link ユーザ名によって受信者をアドレス指定できますが、情報が変更されている場合はこれを Mobile Link によって自動的に更新することはできません。ユーザ自身が手動で変更する必要があります。電子メールアドレスは変更されることが少ないので、この方法は SMTP ゲートウェイで特に便利です。

UDP ゲートウェイでは、再接続のたびに IP アドレスが変更される場合、静的エントリに依存することはできません。この問題を解決するには、IP アドレスではなくホスト名をアドレス指定します。ただし、このソリューションでは、DNS サーバテーブルの更新速度が低下するため、Push 通知が誤配信される可能性があります。システムプロシージャを設定して、システムテーブルをプログラムによって更新することもできます。

手順

1. 各デバイスに対して、ml_device システムテーブルにデバイスレコードを追加します。次に例を示します。

```
CALL ml_set_device(  
  'myWindowsMobile',  
  'MobiLink Listeners for myWindowsMobile - 9.0.1',  
  '1',  
  'not used',  
  'y',  
  'manually entered by administrator'  
);
```

最初のパラメータである **myWindowsMobile** は、ユーザ定義のユニークなデバイス名です。2 番目のパラメータには、Mobile Link Listener バージョンに関するオプションの注釈が含まれています。3 番目のパラメータは、Mobile Link Listener のバージョンを指定します。SQL Anywhere 9.0.0 Mobile Link Listener の場合は **0** を、9.0.0 以降の Windows 用の Mobile Link Listener の場合は **2** を使用します。4 番目のパラメータは、オプションのデバイス情報を指定します。5 番目のパラメータは、デバイストラッキングを無視するかどうかを指定します。最後のパラメータには、このエントリに関するオプションの注釈が含まれています。

2. 各デバイスに対して、ml_device_address システムテーブルにアドレスレコードを追加します。次に例を示します。

```
CALL ml_set_device_address(  
  'myWindowsMobile',  
  'ROGERS AT&T',  
  '55511234567',  
  'y',  
  'y',  
  'manually entered by administrator'  
);
```

最初のパラメータである **myWindowsMobile** は、ユーザ定義のユニークなデバイス名です。2 番目のパラメータはネットワークプロバイダ ID で、Carrier プロパティ network_provider_id と一致している必要があります。3 番目のパラメータは、UDP の IP アドレスです。4 番目のパラメータは、Push 通知の送信用にこのエントリをアクティブにするかどうかを設定します。5 番目のパラメータは、デバイストラッキングを無視するかどうかを指定します。最後のパラメータには、このエントリに関するオプションの注釈が含まれています。

3. 各リモートデータベースに対して、追加した各デバイスの ml_listening システムテーブルに受信者レコードを追加します。これは、デバイスを Mobile Link ユーザ名にマッピングします。次に例を示します。

```
CALL ml_set_listening(  
  'myULDB',  
  'myWindowsMobile',  
  'y',  
  'y',  
  'manually entered by administrator'  
);
```

最初のパラメータは Mobile Link ユーザ名です。2 番目のパラメータは、ユーザ定義のユニークなデバイス名です。3 番目のパラメータは、デバイストラッキングのアドレス指定用にこのエントリをアクティブにするかどうかを設定します。4 番目のパラメータは、デバイストラッキングを無視するかどうかを指定します。最後のパラメータには、このエントリに関するオプションの注釈が含まれています。

結果

指定したデバイスがデバイストラッキングを行うよう設定されます。

関連情報

[ml_set_device システムプロシージャ \[99 ページ\]](#)

[ml_set_listening システムプロシージャ \[102 ページ\]](#)

[ml_set_device_address システムプロシージャ \[100 ページ\]](#)

[Carrier プロパティ \[59 ページ\]](#)

1.2.5.2.2 デバイストラッキングゲートウェイ設定のクイックスタート

次の手順では、デバイストラッキングゲートウェイを設定する方法の概要を説明します。

1. 必要に応じて、SYNC ゲートウェイ、UDP ゲートウェイ、または SMTP ゲートウェイを設定します。
Mobile Link サーバを起動すると、これらのゲートウェイがデフォルト設定を使用して設定されています。

i 注記

SMTP ゲートウェイの場合、Carrier の設定が必要です。

2. 次の条件に従って新しい Notifier を作成し、request_cursor イベントを設定します。
 - ゲートウェイ名は、使用するデバイストラッキングゲートウェイの名前にしてください。デフォルトのゲートウェイ名は、Default-DeviceTracker です。この名前は、結果セットの最初のカラムで指定されています。
 - アドレス名には、デバイスのリモート ID を設定してください。dblsn -t+ オプションを使用して、Mobile Link サーバにリモート ID を登録します。この名前は、結果セットの 4 番目のカラムで指定されています。
3. Mobile Link の ml_user システムテーブルに Mobile Link Listener 名を追加します。
デフォルトの Mobile Link Listener 名は、`device_name-dblsn` (`device_name` はデバイス名) です。
Mobile Link Listener を実行して、Mobile Link Listener メッセージウィンドウ内のデバイス名を確認します。または、`dblsn -e` オプションを使用してデバイス名を設定するか、`dblsn -u` オプションを使用して別の Mobile Link Listener 名を設定できます。
4. 必要なオプションを指定して Mobile Link Listener を起動します。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Carrier と Carrier 設定 \[31 ページ\]](#)

[request_cursor イベント \[44 ページ\]](#)

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

[-u dblsn オプション \[78 ページ\]](#)

[-e dblsn オプション \[67 ページ\]](#)

1.2.5.2.3 Carrier と Carrier 設定

Carrier は、Mobile Link システムテーブルまたは Notifier プロパティファイルに保存される Mobile Link オブジェクトで、サーバ起動同期で使用される通信業者に関する情報が含まれます。

Notifier では有効な電子メールアドレスを作成する必要があるため、SMTP ゲートウェイを使用して Push 通知を送信するには、無線通信事業者を設定してください。また、従属 SMTP ゲートウェイが有効なデバイストラッキングゲートウェイを使用する場合にも、無線通信事業者を設定してください。

ネットワークプロバイダ ID や SMS 電子メールのプレフィクスなど、Carrier のプロパティは、Mobile Link サーバで設定します。複数の Carrier サービスに対応するには、Mobile Link サーバで複数の Carrier を設定します。

送信者の構文

Push 通知は、Mobile Link Listener で受信され、メッセージフィルタリング用に前処理されると、複数のキーワードに分割されます。メッセージの *sender* キーワードは電子メールアドレスです。この電子メールアドレスは、デバイスで生成され、無線通信事業者によって異なります。

sender 構文は、次のフォーマットになります。

```
sender = sms_email_user_prefix phone-number@sms_email_domain
```

i 注記

`sms_email_user_prefix` と `phone-number` の間には、スペースを入れません。

`sms_email_user_prefix` 値と `sms_email_domain` 値は Carrier プロパティです。Mobile Link サーバで設定してください。`phone-number` 値は、`ml_device_address` システムテーブルの `address` カラムから取得されます。

送信者の構文を指定するには、Carrier サービスを使用するデバイスで Mobile Link Listener を実行します。メッセージのロギングを有効にし、`dblsh -m and -v` オプションを使用して冗長レベルを 2 に設定します。Mobile Link Listener のロード後に、メッセージログを確認します。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Carrier プロパティ \[59 ページ\]](#)

[高度: メッセージ構文 \[104 ページ\]](#)

1.3 サーバ起動同期の Mobile Link サーバ設定

サーバ側設定は、Notifier プロパティ、ゲートウェイプロパティ、Carrier プロパティ、Notifier イベントで構成されます。

これらの設定を行うには、次のいずれかの方法を使用します。

- SQL Central
- Notifier 設定ファイル
- ml_add_property システムプロシージャ

SQL Central と ml_add_property システムプロシージャを使用する方法では、ml_property システムテーブルにイベントと設定が追加されます。

i 注記

サーバ側設定を変更しても、Mobile Link サーバの稼働中は変更が有効になりません。新しい設定を適用するには、Mobile Link サーバを停止して再度起動する必要があります。

ml_property システムテーブルでサーバ側設定を行ってあるときに Notifier 設定ファイルを使用する場合は、システムテーブル設定が必ず最初にロードされ、その次にファイル設定がロードされます。Notifier 設定ファイルによって既存のサーバ側設定が上書きされますが、この変更は統合データベースに永続的には適用されません。

このセクションの内容:

[ml_add_property システムプロシージャを使用したサーバ側の設定 \[33 ページ\]](#)

SQL Anywhere 統合データベースのサーバ側設定を行うには、ml_add_property システムプロシージャを使用します。これらのプロパティとイベントは、Interactive SQL を使用して設定できます。

[SQL Central を使用した Notifier、ゲートウェイ、または Carrier の設定 \[34 ページ\]](#)

SQL Central には、プロパティとイベントを変更するためのグラフィカルユーザインタフェースが用意されています。SQL Central を使用すると、複数の Notifier、ゲートウェイ、Carrier を設定できます。

[Notifier 設定ファイルを使用したサーバ側の設定 \[36 ページ\]](#)

サーバ側設定は、Notifier 設定ファイルに格納できます。このファイルを使用すると、複数の Notifier、ゲートウェイ、Carrier を設定できます。

[Notifier イベント \[39 ページ\]](#)

イベントは、Notifier が Mobile Link Listener をポーリングするたびに起動されます。イベントが起動すると、そのイベントに対応する SQL スクリプトが実行されます。SQL スクリプトは、下記のいずれの Notifier イベントに組み込むことができます。スクリプトの実行は任意ですが、request_cursor ポーリングイベントを記述する必要があります。

[共通プロパティ \[52 ページ\]](#)

共通プロパティは、Notifier、ゲートウェイ、Carrier で共有されます。共通プロパティは、すべてオプションです。

[Notifier プロパティ \[52 ページ\]](#)

Notifier プロパティを使用すると、Notifier の動作を変更できます。Notifier のプロパティは、すべてオプションです。

[ゲートウェイプロパティ \[54 ページ\]](#)

デフォルトでは、Mobile Link サーバを起動すると、あらかじめ定義されている 4 つのゲートウェイが作成されます。これらのゲートウェイは、統合データベース用の Mobile Link 設定スクリプトを実行したときにインストールされます。

[Carrier プロパティ \[59 ページ\]](#)

Carrier プロパティを使用すると、無線通信事業者設定の動作を変更できます。この設定では、電話番号自動追跡のマッピングや電子メールアドレスに対するネットワークプロバイダのマッピングに関する情報を提供します。Carrier プロパティはすべてオプションで、SMTP ゲートウェイを使用している場合にのみ必須です。

1.3.1 ml_add_property システムプロシージャを使用したサーバ側の設定

SQL Anywhere 統合データベースのサーバ側設定を行うには、ml_add_property システムプロシージャを使用します。これらのプロパティとイベントは、Interactive SQL を使用して設定できます。

i 注記

Notifier、ゲートウェイ、Carrier に名前を付ける場合は、ANSI 標準を使用してください。

共通プロパティ構文

```
CALL ml_add_property('SIS', '', 'Property', Value);
```

Notifier プロパティとイベントの構文

```
CALL ml_add_property('SIS', 'Notifier(NotifierName)', 'Event-or-Property', Value);
```

ゲートウェイプロパティ構文

```
CALL ml_add_property('SIS', 'DeviceTracker(DeviceTrackerName)', 'Property', Value);
```

```
CALL ml_add_property('SIS', 'SMTP(SMTPName)', 'Property', Value);
```

```
CALL ml_add_property('SIS', 'UDP(UDPName)', 'Property', Value);
```

```
CALL ml_add_property('SIS', 'SYNC(SYNCName)', 'Property', Value);
```

Carrier プロパティ構文

```
CALL ml_add_property('SIS', 'Carrier(CarrierName)', 'Property', Value);
```

1.3.2 SQL Central を使用した Notifier、ゲートウェイ、または Carrier の設定

SQL Central には、プロパティとイベントを変更するためのグラフィカルユーザインターフェースが用意されています。SQL Central を使用すると、複数の Notifier、ゲートウェイ、Carrier を設定できます。

コンテキスト

Notifier、ゲートウェイ、Carrier に名前を付ける場合は、ANSI 標準を使用してください。

SQL Central を使用してサーバ側設定を実行すると、mlsrv17 -notifier オプションを使用するときに、コマンドラインで Notifier 設定ファイルを指定する必要がありません。

手順

1. SQL Central において、統合データベース用に Mobile Link プロジェクトをまだ作成していない場合は、Mobile Link プラグインを使用して作成します。
2. **▶ ビュー ▶ フォルダ** をクリックします。
3. 左ウィンドウ枠で *Mobile Link 17* を展開し、Mobile Link プロジェクト名、**統合データベース**、統合データベース名の順に展開し、**通知** を選択します。
右ウィンドウ枠に、使用可能な Notifier、ゲートウェイ、Carrier がすべて表示されます。
4. 新しい Notifier、ゲートウェイ、Carrier を作成します。
 - 新しい Notifier を作成するには、右ウィンドウ枠の *Notifier* タブをクリックし、**▶ ファイル ▶ 新規 ▶ Notifier** を選択します。
 - 新しいゲートウェイを作成するには、右ウィンドウ枠の *ゲートウェイ* タブをクリックし、**▶ ファイル ▶ 新規 ▶ ゲートウェイ** をクリックします。
 - 新しい Carrier を作成するには、右ウィンドウ枠の *Carrier* タブをクリックし、**▶ ファイル ▶ 新規 ▶ Carrier** をクリックします。
5. 設定する Notifier、ゲートウェイ、または Carrier を選択します。
 - Notifier プロパティまたはイベントを設定するには、右ウィンドウ枠の *Notifier* タブをクリックし、設定する Notifier を選択します。
 - ゲートウェイプロパティを設定するには、右ウィンドウ枠の *ゲートウェイ* タブをクリックし、設定するゲートウェイを選択します。

- Carrier プロパティを設定するには、右ウィンドウ枠の *Carrier* タブをクリックし、設定する Carrier を選択します。

▶ **ファイル** ▶ **プロパティ** ▶ をクリックします。

選択した Notifier、ゲートウェイ、または Carrier に適用可能なすべての設定を調整できるウィンドウが表示されます。

6. **OK** をクリックします。

結果

Notifier、ゲートウェイまたは Carrier が設定され、使用できるようになります。

このセクションの内容:

[Notifier 設定ファイルからのサーバ側設定のインポート \[35 ページ\]](#)

サーバ側設定を ml_property_table にインポートするには、Notifier 設定ファイルを使用します。

[Notifier 設定ファイルへのサーバ側設定のエクスポート \[36 ページ\]](#)

サーバ側設定は、ml_property テーブルから Notifier 設定ファイルにエクスポートできます。設定をエクスポートすると、複数のバージョンのサーバ側設定を作成し、mlsrv17-notifier オプションを使用して異なるバージョンをロードできます。

1.3.2.1 Notifier 設定ファイルからのサーバ側設定のインポート

サーバ側設定を ml_property_table にインポートするには、Notifier 設定ファイルを使用します。

手順

1. SQL Central において、統合データベース用に Mobile Link プロジェクトをまだ作成していない場合は、Mobile Link ブラウザを使用して作成します。
2. ▶ **ビュー** ▶ **フォルダ** ▶ をクリックします。
3. 左ウィンドウ枠で *Mobile Link 17* を展開し、Mobile Link プロジェクト名、**統合データベース**、統合データベース名の順に展開し、**通知** を選択します。
4. ▶ **ファイル** ▶ **インポートの設定** ▶ をクリックし、ウィザードの指示に従います。

結果

設定が、Notifier 設定ファイルから ml_property_table にインポートされます。

1.3.2 Notifier 設定ファイルへのサーバ側設定のエクスポート

サーバ側設定は、ml_property テーブルから Notifier 設定ファイルにエクスポートできます。設定をエクスポートすると、複数のバージョンのサーバ側設定を作成し、mlsrv17 -notifier オプションを使用して異なるバージョンをロードできます。

手順

1. SQL Central において、統合データベース用に Mobile Link プロジェクトをまだ作成していない場合は、Mobile Link プラグインを使用して作成します。
2. ▶ ビュー ▶ フォルダ ▶ をクリックします。
3. 左ウィンドウ枠で *Mobile Link 17* を展開し、Mobile Link プロジェクト名、統合データベース、統合データベース名の順に展開し、通知を選択します。
4. ▶ ファイル ▶ 設定のエクスポート ▶ をクリックし、ウィザードの指示に従います。

結果

指定した設定が Notifier 設定ファイルへエクスポートされます。

1.3.3 Notifier 設定ファイルを使用したサーバ側の設定

サーバ側設定は、Notifier 設定ファイルに格納できます。このファイルを使用すると、複数の Notifier、ゲートウェイ、Carrier を設定できます。

i 注記

Notifier、ゲートウェイ、Carrier に名前を付ける場合は、ANSI 標準を使用してください。

Notifier 設定ファイルの作成と設定

Notifier 設定ファイルは、テキストエディタを使用して作成したり、SQL Central からエクスポートしたプロパティとイベントの設定から生成したりできます。

一般的な Notifier 設定ファイルのレイアウトを参照するには、`%SQLANY%SAMP17%MobiLink%template.Notifier` テンプレートファイルを開きます。このテンプレートファイルでは、サーバ側プロパティとイベントを設定するための例を提供しません。

必要な設定が完了したら Notifier 設定ファイルを保存し、サーバ側プロパティとイベントを Mobile Link サーバにロードします。

共通プロパティ構文

```
Property = Value
```

Notifier イベント構文

```
Notifier(NotifierName).Event = ¥
# Replace this text with SQL script.           ¥
# Be sure to put a backslash (\) at           ¥
# the end of every line of code               ¥
# if your event requires multiple             ¥
# lines of text.
```

Notifier プロパティ構文

```
Notifier(NotifierName).Property = Value
```

ゲートウェイプロパティ構文

```
# For Device tracking gateways:
DeviceTracker(DeviceTrackerName).Property = Value
# For SMTP gateways:
SMTP(SMTPName).Property = Value
# For SYNC gateways:
SYNC(SYNCName).Property = Value
# For UDP gateways:
UDP(UDPName).Property = Value
```

Carrier プロパティ構文

```
Carrier(CarrierName).Property = Value
```

Notifier 設定ファイルのロード

Notifier 設定ファイルを Mobile Link サーバにロードするには、コマンドラインから `-notifier` オプションを指定して `mlsrv17` を実行します。たとえば、CarDealer.Notifier 設定ファイルに定義されたサーバ側設定を使用するには、次のコマンドを実行します。

```
mlsrv17 ... -notifier "c:¥CarDealer.Notifier"
```

ファイルを指定しない場合は、デフォルトで `config.Notifier` ファイルがロードされます。

i 注記

デフォルトの SYNC ゲートウェイを使用する場合、サーバ側設定を Notifier 設定ファイルには保存できません。別の方法を使用して、この設定を `ml_property` システムテーブルに格納する必要があります。

エスケープシーケンスの使用

円記号 (¥) はエスケープ文字です。Notifier 設定ファイルで使用できる一般的なエスケープシーケンスのリストは、次のとおりです。

エスケープシーケンス	説明
¥b	バックスペース
¥t	タブ
¥n	改行
¥r	キャリッジリターン
¥"	二重引用符 (")
¥'	一重引用符 (')
¥¥	円記号 (¥)
¥e	エスケープ

Unicode のエスケープシーケンス形式は `¥uXXXX`、ASCII のエスケープシーケンス形式は `¥xxx` です。ここで、各 `x` は 16 進数字を表します。

複数行のテキストが必要なプロパティまたはイベントを編集する場合は、1 つの円記号 (¥) を各行の最後に追加します。

関連情報

[SQL Central を使用した Notifier、ゲートウェイ、または Carrier の設定 \[34 ページ\]](#)

1.3.4 Notifier イベント

イベントは、Notifier が Mobile Link Listener をポーリングするたびに起動されます。イベントが起動すると、そのイベントに対応する SQL スクリプトが実行されます。SQL スクリプトは、下記のいずれの Notifier イベントに組み込むことができます。スクリプトの実行は任意ですが、request_cursor ポーリングイベントを記述する必要があります。

Notifier イベントは、ポーリングイベント、接続イベント、非同期イベントの 3 つに分類されます。ポーリングイベントは、Notifier が統合データベースをチェックするたびに起動し、begin_poll イベントと end_poll イベントの間に発生するすべてのイベントが含まれます。接続イベントは、Notifier のデータベース接続中に起動します。非同期イベントは、同期処理中の任意の時点で起動する可能性があります。

特に指定しないかぎり、Notifier イベントはお奨めする方法のいずれかを使用して設定できます。

Mobile Link Listener が Notifier をポーリングすると、これらのイベントが次の順序で起動します。

```
Fire begin_connection event
For each poll (
  Fire begin_poll event
  Fire shutdown_query event
  Fire request_cursor event
  For all requests expired before required confirmation (
    Fire error_handler event
  )
  Fire request_delete event
  Fire end_poll event
)
Fire end_connection event
```

このセクションの内容:

[ポーリング中のイベント \[39 ページ\]](#)

ポーリングイベントは、Notifier が統合データベースをチェックするたびに起動する Notifier イベントに分類されます。これらのイベントには、begin_poll イベントと end_poll イベントの間に発生するすべてのイベントが含まれます。

[接続イベント \[47 ページ\]](#)

接続イベントは、Notifier データベースの接続時に起動する Notifier イベントに分類されます。

[非同期イベント \[48 ページ\]](#)

非同期イベントは、同期処理中の任意の時点で起動する可能性のある Notifier イベントに分類されます。

1.3.4.1 ポーリング中のイベント

ポーリングイベントは、Notifier が統合データベースをチェックするたびに起動する Notifier イベントに分類されます。これらのイベントには、begin_poll イベントと end_poll イベントの間に発生するすべてのイベントが含まれます。

このセクションの内容:

[begin_poll イベント \[40 ページ\]](#)

このポーリングイベントは SQL スクリプトを受け入れ、Notifier が統合データベースをチェックして Push 要求があるかどうかを確認する前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

[end_poll イベント \[41 ページ\]](#)

このポーリングイベントは SQL スクリプトを受け入れ、Notifier が統合データベースをチェックして Push 要求があるかどうかを確認した後に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

[error_handler イベント \[41 ページ\]](#)

転送に失敗した場合や転送が確認されなかった場合を示すには、このイベントを設定します。たとえば、転送に失敗した場合、このイベントを使用すると、監査テーブルにローを挿入したり、Push 通知を送信したりできます。

[request_cursor イベント \[44 ページ\]](#)

このポーリングイベントは SQL スクリプトを受け入れ、Push 要求を検出すると起動されます。このイベントは設定が必要です。

[request_delete イベント \[46 ページ\]](#)

このポーリングイベントは SQL スクリプトを受け入れ、Push 要求の削除の必要性が検出されるとクリーンアップ処理を実行するために起動されます。

[shutdown_query イベント \[46 ページ\]](#)

このポーリングイベントは SQL スクリプトを受け入れ、begin_poll イベントの後に起動されます。戻り値は Notifier の停止ステータスを示します。デフォルトでは、値は NULL であるため、このイベントは起動されません。

1.3.4.1.1 begin_poll イベント

このポーリングイベントは SQL スクリプトを受け入れ、Notifier が統合データベースをチェックして Push 要求があるかどうかを確認する前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

例

この例では、Notifier A という名前の Notifier で使用する Push 要求を作成します。SQL 文を使用して、PushRequest という名前のテーブルにローを挿入します。このテーブルの各ローは、1つのアドレスに送信するメッセージを表しています。WHERE 句によって、PushRequest テーブルに挿入される Push 要求が決まります。

ml_add_property システムプロシージャを SQL Anywhere 統合データベースで使用するには、次のコマンドを実行します。

```
ml_add_property(
  'SIS',
  'Notifier(Notifier A)',
  'begin_poll',
  'INSERT INTO PushRequest
    (gateway, mluser, subject, content)
  SELECT 'MyGateway', DISTINCT mluser, 'sync',
    stream_param
  FROM MLUserExtra, mluser_union, Dealer
  WHERE MLUserExtra.mluser = mluser_union.name
  AND (push_sync_status = 'waiting for request'
    OR datediff( hour, last_status_change, now() ) > 12 )
  AND ( mluser_union.publication_name is NULL
    OR mluser_union.publication_name = 'FullSync' )
  AND Dealer.last_modified > mluser_union.last_sync_time'
);
```


関連情報

[Push 要求 \[9 ページ\]](#)

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.4.1.2 end_poll イベント

このポーリングイベントは SQL スクリプトを受け入れ、Notifier が統合データベースをチェックして Push 要求があるかどうかを確認した後に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

このイベントを使用すると、テーブルのクリーンアップを実行したり、ポーリングの結果をログに記録できます。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.4.1.3 error_handler イベント

転送に失敗した場合や転送が確認されなかった場合を示すには、このイベントを設定します。たとえば、転送に失敗した場合、このイベントを使用すると、監査テーブルにローを挿入したり、Push 通知を送信したりできます。

次の表に、error_handler イベントを使用して取得できるパラメータの詳細を示します。

スクリプトパラメータ	タイプ	説明
request_option (out)	Integer	エラーハンドラが戻った後に Notifier が Push 要求に対して実行する処理を制御します。出力は、次のいずれかの値になります。 <ul style="list-style-type: none">0: エラーコードに基づいてデフォルトアクションを実行し、エラーを記録します。1: 何もしません。2: request_delete イベントを実行します。3: セカンダリゲートウェイへの配信を試行します。

スクリプトパラメータ	タイプ	説明
error_code (in)	Integer	<p>エラーコードには、次のいずれかの値を使用します。</p> <ul style="list-style-type: none"> • -1: 確認が成功したあと、要求はタイムアウトされました。 • -8: 配信試行中にエラーが発生しました。
request_id (in)	Integer	要求を識別します。
gateway (in)	varchar	Push 要求に関連付けられているゲートウェイを指定します。
address (in)	varchar	<p>Push 要求に関連付けられているアドレスを指定します。</p> <p>セキュリティ上の理由から、オプションの Notifier エラーハンドラが呼び出されると、次のいずれかに該当しない、通知の件名または内容の文字がアスタリスク (*) に置き換わります。</p> <ul style="list-style-type: none"> • 英数字 • ピリオド • コロン • マイナス記号 • プラス記号 • アンダースコア <p>通知で送信される値は、元の値と同じです。</p>
subject (in)	varchar	<p>Push 要求に関連付けられている件名を指定します。</p> <p>セキュリティ上の理由から、オプションの Notifier エラーハンドラが呼び出されると、次のいずれかに該当しない、通知の件名または内容の文字がアスタリスク (*) に置き換わります。</p> <ul style="list-style-type: none"> • 英数字 • ピリオド • コロン • マイナス記号 • プラス記号 • アンダースコア <p>通知で送信される値は、元の値と同じです。</p>

スクリプトパラメータ	タイプ	説明
content (in)	varchar	<p>Push 要求に関連付けられている内容を指定します。</p> <p>セキュリティ上の理由から、オプションの Notifier エラーハンドラが呼び出されると、次のいずれかに該当しない、通知の件名または内容の文字がアスタリスク (*) に置き換わります。</p> <ul style="list-style-type: none"> • 英数字 • ピリオド • コロン • マイナス記号 • プラス記号 • アンダースコア <p>通知で送信される値は、元の値と同じです。</p>

i 注記

このイベントにはシステムプロシージャの使用が必要です。SQL Central を使用して、このイベントを直接設定することはできません。

例

次の例では、CustomError というテーブルを作成し、CustomErrorHandler というストアドプロシージャを使用してエラーをテーブルに記録します。出力パラメータ Notifier_opcode は常に 0 で、デフォルトの Notifier 処理が使用されます。

```

CREATE TABLE CustomError(
  error_code integer,
  request_id integer,
  gateway varchar(255),
  address varchar(255),
  subject varchar(255),
  content varchar(255),
  occurAt timestamp not null default timestamp
);
CREATE PROCEDURE CustomErrorHandler(
  out @Notifier_opcode integer,
  in @error_code integer,
  in @request_id integer,
  in @gateway varchar(255),
  in @address varchar(255),
  in @subject varchar(255),
  in @content varchar(255)
)
BEGIN
  INSERT INTO CustomError(
    error_code,
    request_id,
    gateway,
    address,
    subject,
    content)
  VALUES (
    @error_code,

```

```
@request_id,  
@gateway,  
@address,  
@subject,  
@content  
);  
SET @Notifier_opcode = 0;  
END
```

ml_add_property システムプロシージャを SQL Anywhere 統合データベースで使用するには、次のコマンドを実行します。

```
call ml_add_property(  
  'SIS',  
  'Notifier(myNotifier)',  
  'error_handler',  
  'call CustomErrorHandler(?, ?, ?, ?, ?, ?, ?)');
```

または、Notifier 設定ファイルに次の行を追加しても、このイベントを起動できます。

```
Notifier(myNotifier).error_handler = call CustomErrorHandler(?, ?, ?, ?, ?, ?, ?)
```

mlsrv17 -notifier オプションを使用してファイルを実行します。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Notifier 設定ファイルを使用したサーバ側の設定 \[36 ページ\]](#)

1.3.4.1.4 request_cursor イベント

このポーリングイベントは SQL スクリプトを受け入れ、Push 要求を検出すると起動されます。このイベントは設定が必要です。

ライトウェイトポーラーを使用する場合の Push 要求のフェッチ (推奨)

このイベントで結果セットに最大 3 つのカラムが含まれる場合、Notifier はサーバとデバイスの間に永続的な接続がないことと、デバイスが Notifier をポーリングしてから Push 通知を送信する必要があることを確認します。Notifier は、結果セットをキャッシュしてから Push 通知を送信します。Mobile Link サーバでは、ポーリングキーによってデバイスを識別します。ポーリングキーは、デバイスが Notifier をポーリングするたびにデバイスが送信します。

このイベントの結果セットには、次のカラムが指定した順序で含まれている必要があります。

- Poll key

- Subject (オプション)
- Content (オプション)

ゲートウェイを使用する場合の Push 通知のフェッチ

このイベントで結果セットに 3 つを超えるカラムが含まれる場合、Notifier はサーバとデバイス間に永続的な接続が存在することを確認し、Push 要求が検出されたときにゲートウェイを使用して Push 通知を送信します。

このイベントの結果セットには、次のカラムが指定した順序で含まれている必要があります。

- Request ID (オプション)
- Gateway
- Subject
- Content
- Address
- Resend interval (オプション)
- Time to live (オプション)

例

次の例では、ml_add_property システムプロシージャを使用して、Simple という名前のカスタム Notifier 用の request_cursor イベントスクリプトを作成します。SELECT 文では、Notifier に PushRequest という名前のテーブルから Push 要求を検出するように指示します。

```
CALL ml_add_property('SIS', 'Notifier(Simple)', 'request_cursor',  
  'SELECT poll_key,  
    subject,  
    content  
  FROM PushRequest'  
);
```

スクリプトに WHERE 句を追加して、送信済みの要求をフィルタします。たとえば、要求を挿入した時刻を追跡する Push 要求カラムを追加して、このイベントで WHERE 句を使用すると、ユーザが最後に同期を行った時刻よりも前に挿入された要求をフィルタできます。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Push 要求の要件 \[9 ページ\]](#)

1.3.4.1.5 request_delete イベント

このポーリングイベントは SQL スクリプトを受け入れ、Push 要求の削除の必要性が検出されるとクリーンアップ処理を実行するために起動されます。

このイベントはパラメータとして request ID を受け入れ、request ID ごとに実行されます。request_cursor イベントには、request_delete イベントを使用するための request ID カラムが含まれている必要があります。指定したパラメータまたは疑問符 (?) を使用すると、request ID を参照できます。別のプロセスや end_poll イベントなどのイベントにクリーンアップ処理を割り当ててある場合、このイベントはオプションです。

Notifier では、DELETE 文を使用して、次の形式の Push 要求を削除できます。

暗黙的に除外

この Push 要求は、以前発生したが、request_cursor イベントから取得された現在の要求セットにはありません。

確認済み

配信が確認された Push 要求です。

失効

この Push 要求は、resend 属性と現在の時刻に基づき、有効期限が切れています。resend 属性のない要求は、次の要求に表示された場合でも、有効期限が切れていると見なされます。

request_delete イベントを使用すると、有効期限が切れた要求または暗黙的に除外された要求を削除できなくなります。たとえば、`%SQLANYSAMP17%MobiLink¥SIS_CarDealer` ディレクトリの CarDealer サンプルでは、request_delete イベントを使用して、PushRequest テーブルのステータスフィールドを 'processed' に設定しています。

```
UPDATE PushRequest SET status='processed' WHERE req_id = ?
```

このサンプルの begin_poll イベントでは、最後の同期時間を利用して、処理済みの Push 要求を削除する前にリモートデバイスが最新状態であるかどうかをチェックしています。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.4.1.6 shutdown_query イベント

このポーリングイベントは SQL スクリプトを受け入れ、begin_poll イベントの後に起動されます。戻り値は Notifier の停止ステータスを示します。デフォルトでは、値は NULL であるため、このイベントは起動されません。

Notifier を停止するには、"yes" を返すように SQL スクリプトを設定します。それ以外の場合は、"no" を返すように設定します。Notifier が停止した場合、end_poll イベントは起動されません。

停止ステータスをテーブルに格納している場合は、end_connection イベントを使用してステータスをリセットします。

例

次の例では、ml_add_property システムプロシージャを使用して、Simple という名前のカスタム Notifier 用の shutdown_query イベントスクリプトを作成します。SELECT 文によって、tooManyNotifierErrors メソッドから true が返された場合に停止するよう Notifier に通知しています。

```
CALL ml_add_property('SIS', 'Notifier(Simple)', 'shutdown_query',
  'SELECT
    IF tooManyNotifierErrors() THEN
      'yes'
    ELSE
      'no'
    ENDIF'
);
```

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[end_connection イベント \[48 ページ\]](#)

1.3.4.2 接続イベント

接続イベントは、Notifier データベースの接続時に起動する Notifier イベントに分類されます。

このセクションの内容:

[begin_connection イベント \[47 ページ\]](#)

このイベントは SQL スクリプトを受け入れ、Notifier が統合データベースに接続した後、Push 要求をチェックする前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

[end_connection イベント \[48 ページ\]](#)

このイベントは SQL スクリプトを受け入れ、Notifier が統合データベースから切断する直前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

1.3.4.2.1 begin_connection イベント

このイベントは SQL スクリプトを受け入れ、Notifier が統合データベースに接続した後、Push 要求をチェックする前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

このイベントを使用すると、テンポラリテーブルまたは変数を作成できます。このイベントを使用して、独立性レベルを変更しないでください。独立性レベルを指定するには、isolation プロパティを使用します。

統合データベースへの接続が失われると、Notifier は再接続した直後にこのイベントを再実行します。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[Notifier プロパティ \[52 ページ\]](#)

1.3.4.2.2 end_connection イベント

このイベントは SQL スクリプトを受け入れ、Notifier が統合データベースから切断する直前に起動されます。デフォルトでは、値は NULL であるため、このイベントは起動されません。

このイベントを使用すると、SQL 変数やテンポラリテーブルなどの一時的な記憶領域をクリーンアップできます。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.4.3 非同期イベント

非同期イベントは、同期処理中の任意の時点で起動する可能性のある Notifier イベントに分類されます。

このセクションの内容:

[confirmation_handler イベント \[48 ページ\]](#)

Mobile Link Listener がアップロードした配信確認情報を処理するには、このイベントを設定します。ステータスパラメータが 0 を返す場合、request_id で識別された Push 要求は、remote_device パラメータで識別された Mobile Link Listener によって正常に受信されています。

1.3.4.3.1 confirmation_handler イベント

Mobile Link Listener がアップロードした配信確認情報を処理するには、このイベントを設定します。ステータスパラメータが 0 を返す場合、request_id で識別された Push 要求は、remote_device パラメータで識別された Mobile Link Listener によって正常に受信されています。

request_option パラメータを使用すると、配信確認への応答としてアクションを開始できます。request_option が 0 の場合、confirmation_handler イベントはデフォルトのアクションを開始します。つまり、request_delete イベントが実行されて、元の Push 要求が削除されます。配信確認を送信するデバイスが request_id で識別されたデバイスと一致しない場合、デフォルトのアクションでは、元の Push 要求がセカンダリゲートウェイを使用して送信されます。

i 注記

Mobile Link Listener が配信確認情報をアップロードできるようにするには、`dblsn -x` オプションを使用します。配信確認は必要だが IP 追跡は不要な場合は、`dblsn -ni` オプションを使用します。

i 注記

このイベントにはシステムプロシージャの使用が必要です。SQL Central を使用する方法では、このイベントを直接設定できません。

`confirmation_handler` イベントを使用して、次のパラメータを取得できます。

スクリプトパラメータ	タイプ	説明
<code>request_option (out)</code>	Integer	<p>ハンドラが戻った後に Notifier が要求に対して実行する処理を制御します。次の値が返されます。</p> <ul style="list-style-type: none">• <code>0</code>: <code>status</code> パラメータの値に基づいてデフォルトの Notifier アクションを実行します。応答デバイスがターゲットデバイスであることを <code>status</code> が示している場合、Notifier は要求を削除します。そうでない場合は、Notifier はセカンダリゲートウェイへの配信を試行します。• <code>1</code>: 何もしません。• <code>2</code>: <code>Notifier.request_delete</code> を実行します。• <code>3</code>: セカンダリゲートウェイへの配信を試行します。
<code>status (in)</code>	Integer	<p>状況の概要。ステータスは、開発時に不適切なフィルタやハンドラ属性などの問題の識別に使用できます。次の値が返されます。</p> <ul style="list-style-type: none">• <code>0</code>: 受信され、確認されました。• <code>-2</code>: 正しい応答相手でしたが、メッセージは拒否されました。• <code>-3</code>: 正しい応答相手で、メッセージは受け入れられましたが、アクションは失敗しました。• <code>-4</code>: 間違った応答相手でしたが、メッセージは受け入れられました。• <code>-5</code>: 間違った応答相手で、メッセージは拒否されました。• <code>-6</code>: 間違った応答相手でした。メッセージは受け入れられ、アクションは正常に終了しました。• <code>-7</code>: 間違った応答相手でした。メッセージは受け入れられましたが、アクションは失敗しました。

スクリプトパラメータ	タイプ	説明
request_id (in)	Integer	request ID。request_cursor イベントには、confirmation_handler イベントを使用するための request ID カラムが含まれている必要があります。
remote_code (in)	Integer	Mobile Link Listener からレポートされた概要です。次の値が返されます。 <ul style="list-style-type: none"> • 1: メッセージは受け入れられました。 • 2: メッセージは拒否されました。 • 3: メッセージは受け入れられ、アクションは正常に終了しました。 • 4: メッセージは受け入れられ、アクションは失敗しました。
remote_device (in)	varchar	応答 Mobile Link Listener のデバイス名です。
remote_mluser (in)	varchar	応答 Mobile Link Listener の Mobile Link ユーザ名です。
remote_action_return (in)	varchar	リモートアクションのリターンコードです。
remote_action (in)	varchar	アクションコマンド用に予約済みです。
gateway (in)	varchar	要求に関連付けられているゲートウェイです。
address (in)	varchar	要求に関連付けられているアドレスです。
subject (in)	varchar	要求に関連付けられている件名です。
content (in)	varchar	要求に関連付けられている内容です。

例

次の例では、CustomConfirmation というテーブルを作成し、CustomConfirmationHandler という名前のストアプロシージャを使用して確認をログ記録します。出力パラメータ request_option は常に 0 に設定され、デフォルト Notifier 処理が使用されます。

```
CREATE TABLE CustomConfirmation(
  error_code integer,
  request_id integer,
  remote_code integer,
  remote_device varchar(128),
  remote_mluser varchar(128),
  remote_action_return varchar(128),
  remote_action varchar(128),
  gateway varchar(255),
  address varchar(255),
  subject varchar(255),
  content varchar(255),
  occurAt timestamp not null default timestamp
);
CREATE PROCEDURE CustomConfirmationHandler(
  out @request_option integer,
  in @error_code integer,
  in @request_id integer,
  in @remote_code integer,
```

```

in @remote_device varchar(128),
in @remote_mluser varchar(128),
in @remote_action_return varchar(128),
in @remote_action varchar(128),
in @gateway varchar(255),
in @address varchar(255),
in @subject varchar(255),
in @content varchar(255)
)
BEGIN
INSERT INTO CustomConfirmation(
error_code,
request_id,
remote_code,
remote_device,
remote_mluser,
remote_action_return,
remote_action,
gateway,
address,
subject,
content)
VALUES (
@error_code,
@request_id,
@remote_code,
@remote_device,
@remote_mluser,
@remote_action_return,
@remote_action,
@gateway,
@address,
@subject,
@content
);
SET @request_option = 0;
END

```

ml_add_property システムプロシージャを SQL Anywhere 統合データベースで使用するには、次のコマンドを実行します。

```

call ml_add_property(
'SIS',
'Notifier(myNotifier)',
'confirmation_handler',
'call CustomConfirmation(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)');

```

または、Notifier 設定ファイルに次の行を追加して、このイベントを呼び出すこともできます。

```

Notifier(myNotifier).confirmation_handler = call
CustomConfirmation(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

```

mlsrv17 -notifier オプションを使用してファイルを実行します。

関連情報

[Notifier イベント \[39 ページ\]](#)

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

[ゲートウェイプロパティ \[54 ページ\]](#)

Notifier 設定ファイルを使用したサーバ側の設定 [36 ページ]

Windows デバイス用の Mobile Link Listener オプション [62 ページ]

1.3.5 共通プロパティ

共通プロパティは、Notifier、ゲートウェイ、Carrier で共有されます。共通プロパティは、すべてオプションです。

プロパティ	値	説明
verbosity	{0 1 2 3}	Notifier、ゲートウェイ、Carrier の冗長性レベルを指定します。次の値を使用できます。 <ul style="list-style-type: none">• 0:トレーシングなし• 1:起動、シャットダウン、プロパティのトレーシング• 2:通知を表示• 3:完全レベルのトレーシング デフォルト値は 0 です。

1.3.6 Notifier プロパティ

Notifier プロパティを使用すると、Notifier の動作を変更できます。Notifier のプロパティは、すべてオプションです。

プロパティ	値	説明
connect_string	<code>connection_string</code>	データベースへの接続に使用されるデフォルトの接続動作を上書きします。デフォルト値は <code>com.sap.ml.script.ServerContext</code> です。この値では、mlsrv17 コマンドラインで指定された接続文字列を使用します。 別のデータベースに接続するときに通知ロジックとデータを同期データから分離するのに便利です。ほとんどの展開ではこのプロパティを設定しません。
enable	{yes no}	Notifier を有効にするかどうかを指定します。-notifier mlsrv17 オプションを実行すると、有効な Notifier がすべて起動します。

プロパティ	値	説明
gui	{yes no}	<p>Notifier の動作中に Notifier ウィンドウを表示するかどうかを指定します。デフォルト値は <code>yes</code> です。</p> <p>この Notifier ウィンドウを使用すると、ポーリング間隔を一時的に変更したり、すぐにポーリングを実行したりできます。また、Mobile Link サーバを停止せずに Notifier を停止するために使用することも可能です。一度停止すると、Mobile Link サーバを停止して再度起動しないと、Notifier を再度起動できません。</p>
isolation	{0 1 2 3}	<p>Notifier のデータベース接続の独立性レベルを指定します。次の値を使用できます。</p> <ul style="list-style-type: none"> • <code>0</code>: コミットされない読み出し • <code>1</code>: コミットされた読み出し • <code>2</code>: 繰り返し可能読み出し • <code>3</code>: 直列化可能。 <p>デフォルト値は <code>1</code> です。レベルが高くなると競合が増加しますが、パフォーマンスが逆に低下することがあります。独立性レベルを <code>0</code> に設定すると、コミットされていないデータ（ロールバックする可能性のあるデータ）を読み出すことができます。</p>
poll_every	number{s m h}	<p>確認がタイムアウトになるまでに待機する時間を指定します。次に、使用可能な時間単位のリストを示します。</p> <ul style="list-style-type: none"> • <code>s</code>: 秒単位。 • <code>m</code>: 分単位。 • <code>h</code>: 時間単位。 <p>デフォルト値は <code>1m</code> です。時間単位は、<code>HHhMMmSSs</code> の形式で組み合わせることができます。時間単位が指定されていない場合、時間は秒単位で測定されます。</p>

プロパティ	値	説明
shared_database_connection	{yes no}	<p>Notifier がデータベース接続を共有するかどうかを指定します。デフォルト値は <i>no</i> です。Notifier が接続を共有できるのは、その独立性レベルが同じ場合のみです。</p> <p>パフォーマンスに悪影響を及ぼさずにリソースを節約するには、<i>yes</i> を指定します。状況によっては、接続を共有できないことがあります。たとえば、アプリケーションが Notifier 間でユニークでない SQL 変数名を使用している場合などが該当します。</p>

1.3.7 ゲートウェイプロパティ

デフォルトでは、Mobile Link サーバを起動すると、あらかじめ定義されている 4 つのゲートウェイが作成されます。これらのゲートウェイは、統合データベース用の Mobile Link 設定スクリプトを実行したときにインストールされます。

デフォルトのゲートウェイの名前は、次のとおりです。

- Default-DeviceTracker ゲートウェイ
- Default-SYNC ゲートウェイ
- Default-UDP ゲートウェイ
- Default-SMTP ゲートウェイ

デフォルトゲートウェイを削除したり、名前を変更したりしないでください。代わりに、名前の異なる追加のゲートウェイを作成できます。

DefaultSYNC と DefaultUDP で定義されているプロパティを変更する必要はありませんが、DefaultSYNC ゲートウェイには、SMTP サーバ情報を指定する必要があります。デフォルトのゲートウェイを使用する必要がありますが、必要に応じて代替設定を使用できます。

このセクションの内容:

[デバイストラッキングゲートウェイプロパティ \[55 ページ\]](#)

デバイストラッキングゲートウェイプロパティを使用すると、デバイストラッキングゲートウェイの動作を変更できます。デバイストラッキングゲートウェイプロパティは、すべてオプションです。

[SMTP ゲートウェイプロパティ \[56 ページ\]](#)

SMTP ゲートウェイプロパティを使用すると、SMTP ゲートウェイの動作を変更できます。サーバのプロパティは必須ですが、他の SMTP ゲートウェイプロパティは、すべてオプションです。

[SYNC ゲートウェイプロパティ \[57 ページ\]](#)

SYNC ゲートウェイプロパティを使用すると、SYNC ゲートウェイの動作を変更できます。SYNC ゲートウェイプロパティは、すべてオプションです。

[UDP ゲートウェイプロパティ \[58 ページ\]](#)

UDP ゲートウェイプロパティを使用すると、IP アドレスやポート番号など、UDP ゲートウェイの動作を変更できます。UDP ゲートウェイプロパティは、すべてオプションです。

1.3.7.1 デバイストラッキングゲートウェイプロパティ

デバイストラッキングゲートウェイプロパティを使用すると、デバイストラッキングゲートウェイの動作を変更できます。デバイストラッキングゲートウェイプロパティは、すべてオプションです。

プロパティ	値	説明
confirm_action	{ yes no }	確認が配信時にこのゲートウェイを通じて送信されるかどうかを指定します。デフォルト値は no です。
confirm_delivery	{ yes no }	Mobile Link Listener がメッセージを受信する統合データベースを確認するかどうかを指定します。デフォルト値は yes です。Mobile Link Listener は、-x Mobile Link Listener オプションを指定して起動する必要があります。
description	description_text	ゲートウェイに関する説明です。
enable	{ yes no }	デバイストラッキングゲートウェイを使用するかどうかを指定します。
smtp_gateway	smtp_gateway_name	SMTP 従属ゲートウェイの名前を指定します。デフォルト値は DefaultSMTP です。デバイストラッキングゲートウェイが使用できる SMTP ゲートウェイは、1 つのみです。このゲートウェイは有効にしておく必要があります。
sync_gateway	sync_gateway_name	SYNC 従属ゲートウェイの名前を指定します。デフォルト値は DefaultSYNC です。デバイストラッキングゲートウェイが使用できる SYNC ゲートウェイは、1 つのみです。このゲートウェイは有効にしておく必要があります。
udp_gateway	udp_gateway_name	UDP 従属ゲートウェイの名前を指定します。デフォルト値は DefaultUDP です。デバイストラッキングゲートウェイが使用できる UDP ゲートウェイは、1 つのみです。このゲートウェイは有効にしておく必要があります。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.7.2 SMTP ゲートウェイプロパティ

SMTP ゲートウェイプロパティを使用すると、SMTP ゲートウェイの動作を変更できます。サーバのプロパティは必須ですが、他の SMTP ゲートウェイプロパティは、すべてオプションです。

プロパティ	値	説明
confirm_action	{ <i>yes</i> <i>no</i> }	確認が配信時にこのゲートウェイを通じて送信されるかどうかを指定します。デフォルト値は <i>no</i> です。
confirm_delivery	{ <i>yes</i> <i>no</i> }	このゲートウェイが配信を確認するかどうかを指定します。デフォルト値は <i>no</i> です。
confirm_timeout	number{ <i>s</i> <i>m</i> <i>h</i> }	確認がタイムアウトになるまでに待機する時間を指定します。次に、使用可能な時間単位のリストを示します。 <ul style="list-style-type: none">• <i>s</i>: 秒単位。• <i>m</i>: 分単位。• <i>h</i>: 時間単位。 デフォルト値は <i>1m</i> です。時間単位は、 <i>HHhMMmSSs</i> の形式で組み合わせることができます。時間単位が指定されていない場合、時間は秒単位で測定されます。
description	<i>description_text</i>	ゲートウェイに関する説明です。
enable	{ <i>yes</i> <i>no</i> }	SYNC ゲートウェイを使用するかどうかを指定します。
Listeners_are_900	{ <i>yes</i> <i>no</i> }	すべての Mobile Link Listener が SQL Anywhere 9.0.0 クライアントであるかどうかを指定します。デフォルト値は <i>no</i> です。SQL Anywhere 9.0.1 以降のクライアントについては、この値を <i>no</i> のままにします。
password	<i>password</i>	SMTP サービスのパスワードを指定します。一部のサービスでは必須です。
sender	<i>SMTP_address</i>	SMTP Push 通知の送信側アドレスを指定します。デフォルト値は <i>anonymous</i> です。
server	<i>IP_address_or_hostname</i>	メッセージを Mobile Link Listener に送信するために使用する SMTP サーバの IP アドレスまたはホスト名を指定します。デフォルト値は <i>mail</i> です。
user	<i>username</i>	SMTP サービスのユーザ名を指定します。一部のサービスでは必須です。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.7.3 SYNC ゲートウェイプロパティ

SYNC ゲートウェイプロパティを使用すると、SYNC ゲートウェイの動作を変更できます。SYNC ゲートウェイプロパティは、すべてオプションです。

プロパティ	値	説明
confirm_action	{ yes no }	確認が配信時にこのゲートウェイを通じて送信されるかどうかを指定します。デフォルト値は no です。
confirm_delivery	{ yes no }	このゲートウェイが配信を確認するかどうかを指定します。デフォルト値は no です。
confirm_timeout	number{ s m h }	確認がタイムアウトになるまでに待機する時間を指定します。次に、使用可能な時間単位のリストを示します。 <ul style="list-style-type: none">• s: 秒単位。• m: 分単位。• h: 時間単位。 デフォルト値は 1m です。時間単位は、HHhMMmSSs の形式で組み合わせることができます。時間単位が指定されていない場合、時間は秒単位で測定されます。
description	description_text	ゲートウェイに関する説明です。
enable	{ yes no }	SYNC ゲートウェイを使用するかどうかを指定します。
Listeners_are_900	{ yes no }	すべての Mobile Link Listener が SQL Anywhere 9.0.0 クライアントであるかどうかを指定します。デフォルト値は no です。SQL Anywhere 9.0.1 以降のクライアントについては、この値を no のままにします。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.7.4 UDP ゲートウェイプロパティ

UDP ゲートウェイプロパティを使用すると、IP アドレスやポート番号など、UDP ゲートウェイの動作を変更できます。UDP ゲートウェイプロパティは、すべてオプションです。

プロパティ	値	説明
confirm_action	{ yes no }	確認が配信時にこのゲートウェイを通じて送信されるかどうかを指定します。デフォルト値は <i>no</i> です。
confirm_delivery	{ yes no }	このゲートウェイが配信を確認するかどうかを指定します。デフォルト値は <i>yes</i> です。
confirm_timeout	number{ s m h }	確認がタイムアウトになるまでに待機する時間を指定します。次に、使用可能な時間単位のリストを示します。 <ul style="list-style-type: none">• <i>s</i>: 秒単位。• <i>m</i>: 分単位。• <i>h</i>: 時間単位。 デフォルト値は <i>1m</i> です。時間単位は、 <i>HHhMMmSSs</i> の形式で組み合わせることができます。時間単位が指定されていない場合、時間は秒単位で測定されます。
description	description_text	ゲートウェイに関する説明です。
enable	{ yes no }	UDP ゲートウェイを使用するかどうかを指定します。
Listeners_are_900	{ yes no }	すべての Mobile Link Listener が SQL Anywhere 9.0.0 クライアントであるかどうかを指定します。デフォルト値は <i>no</i> です。SQL Anywhere 9.0.1 以降のクライアントについては、この値を <i>no</i> のままにします。
Listener_port	port_number	リモートデバイスが UDP パケットの送信に使用するポートを指定します。デフォルト値は <i>5001</i> です。
sender	IP_address_or_hostname	マルチホームのホストの場合にのみ使用します。送信者の IP アドレスまたはホスト名を指定します。デフォルト値は <i>localhost</i> です。
sender_port	port_number	UDP パケットの送信に使用するポート番号を指定します。デフォルトでは、オペレーティングシステムによって空きポート番号がランダムに割り当てられます。

関連情報

[サーバ起動同期の Mobile Link サーバ設定 \[32 ページ\]](#)

1.3.8 Carrier プロパティ

Carrier プロパティを使用すると、無線通信事業者設定の動作を変更できます。この設定では、電話番号自動追跡のマッピングや電子メールアドレスに対するネットワークプロバイダのマッピングに関する情報を提供します。Carrier プロパティはすべてオプションで、SMTP ゲートウェイを使用している場合にのみ必須です。

プロパティ	値	説明
enable	{ yes no }	Carrier を使用するかどうかを指定します。
description	description_text	Carrier に関する説明です。
network_provider_id	id_text	ネットワークプロバイダ ID を指定します。Windows Mobile Phone Edition で SMS を使用するには、このプロパティを _generic_ に設定します。
sms_email_domain	domain_name	Carrier のドメイン名を指定します。
sms_email_user_prefix	prefix_name	電子メールアドレスで使用されるプレフィックスを指定します。

1.4 Windows デバイス用の Mobile Link Listener ユーティリティ (dblsn)

Mobile Link Listener ユーティリティにより、Windows デバイスは Push 通知を受領し、アクションを開始できます。

構文

```
dblsn [ options ] -/ message-handler [ -/ message-handler... ]
```

```
message-handler :  
[ polling-option;... ] [ filter;... ] action; [ option;... ]
```

```
polling-option :  
[ ;poll_connect = string ]  
[ ;poll_notifier = string ]  
[ ;poll_key = string ]  
[ ;poll_every = number ]
```

```
option :  
[ ;continue = yes ]  
[ ;confirm_action = yes ]  
[ ;confirm_delivery = no ]  
[ ;maydial = no ]
```

```
filter :  
[ subject = string ]  
[ content = string ]  
[ message = string | message_start = string ]  
[ sender = string ]
```

```

action :
  action = command[;altaction = command ]

command :
  START program [ program-arguments ]
  | RUN program [ program-arguments ]
  | POST window-message TO { window-class-name | window-title }
  | tcpip-socket-action
  | DBLSN FULL SHUTDOWN

tcpip-socket-action :
  SOCKET port=app-port
  [ ;host=app-host ]
  [ ;sendText=text1 ]
  [ ;recvText= text2 [ ;timeout=num-sec ] ]
window-message : string | message-id

```

備考

dblsn の -l メッセージハンドラは、セミコロンで区切られた name-equal-value ペアです。name-equal-value ペアの文字列値は一重引用符で囲む必要があります。そうしないと、文字列値にセミコロンが含まれる場合に dblsn は起動に失敗します。

このセクションの内容:

[Listener の配備のセキュリティ保護 \[61 ページ\]](#)

Listener は外部通知を受信すると、その通知からの情報を指定しながらアプリケーションを起動することができます。外部通知を使用すると、理論上、好ましくない結果をもたらす有害なデータが注入されるおそれがあります。Listener の配備のセキュリティ保護には注意が必要です。

[Windows デバイス用の受信ライブラリ \[61 ページ\]](#)

Mobile Link Listener には、UDP 受信ライブラリ `lsn_udp17.dll` が付属しており、デフォルトではこのライブラリがロードされます。

[Windows デバイス用の Mobile Link Listener オプション \[62 ページ\]](#)

次のオプションは、Mobile Link Listener の設定に使用できます。

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

次のキーワードを使用すると、dblsn -l オプションを使用して作成されたメッセージハンドラを設定できます。

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

アクションは、新しいメッセージハンドラを設定する場合に指定されます。フィルタ条件が満たされると、アクションが開始されます。アクションが失敗した場合は、代替アクションが開始されます。アクションは、`action` キーワードを使用して定義されます。代替アクションは、`altaction` キーワードを使用して定義されます。

[Windows デバイス用の Mobile Link Listener action 変数 \[87 ページ\]](#)

アクションまたはフィルタでは、次の action 変数を使用できます。action 変数は、メッセージハンドラを開始する前に値に置き換えられます。

1.4.1 Listener の配備のセキュリティ保護

Listener は外部通知を受信すると、その通知からの情報を指定しながらアプリケーションを起動することができます。外部通知を使用すると、理論上、好ましくない結果をもたらす有害なデータが注入されるおそれがあります。Listener の配備のセキュリティ保護には注意が必要です。

次の推奨事項を実施して、Listener のセキュリティ保護を行うことをおすすめします。

- `dblsn -x` オプションを使用し、TLS ベースのプロトコル (HTTPS など) を指定することにより、サーバ (Notifier) を検証しネットワーク通信をセキュリティ保護します。
- SMS または UDP リスナは、セキュリティ保護されていないので、使用しません。デフォルトでは、SMS と UDP は両方とも表示されます。
- `dblsn -l` オプションを介して設定されたアクションはすべて、無効な入力を拒否するか、または任意の入力が受信される場合に悪影響を与えないように保証される必要があります。
- `dblsn -l` オプションを介したアクションの指定では、非常に強力なアプリケーションまたはごく一般的なアプリケーション (`cmd.com` など) を呼び出しません。本当に必要なことを実行するだけであり、無効な入力は拒否する専用のアプリケーションを配備および設定します。
- メッセージフィルタを使用してアクションの呼び出しを制限します。
- 機能を指定するアクション変数の使用は回避します。

関連情報

[メッセージフィルタ \[19 ページ\]](#)

[-x dblsn オプション \[80 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[-l dblsn オプション \[69 ページ\]](#)

1.4.2 Windows デバイス用の受信ライブラリ

Mobile Link Listener には、UDP 受信ライブラリ `lsn_udp17.dll` が付属しており、デフォルトではこのライブラリがロードされます。

SMTP ゲートウェイを使用する場合は、SMTP 受信ライブラリを指定する必要があります。ライブラリは、`dblsn -d` オプションを使用して指定できます。ライブラリのオプションは、`dblsn -a` オプションを使用して指定できます。

UDP (`lsn_udp17.dll`)

UDP 受信ライブラリでサポートされているオプションのリストは、次のとおりです。

オプション	説明
<code>Port=port-number</code>	このオプションでは、受信するポート番号を指定します。デフォルトのポートは <code>5001</code> です。
<code>Timeout=seconds</code>	このオプションでは、UDP 受信ポートでの読み込み処理の最大ブロック時間を指定します。この値は、UDP 受信スレッドのポーリング間隔より小さくしてください。デフォルトは <code>0</code> です。
<code>ShowSenderPort</code>	このオプションは、 <code>\$sender action</code> 変数のすべての出現箇所送信側ポート番号を示します。デフォルトでは、ポート番号は非表示です。このオプションを指定すると、ポート番号が <code>:port-number</code> の構文で送信側アドレスの最後に追加されます。
<code>HideWSAErrorBox</code>	ソケット操作でのエラーを示すエラーウィンドウを表示しません。
<code>CodePage=number</code>	マルチバイト文字は、この番号に基づいて Unicode に変換されます。このオプションが適用されるのは、Windows Mobile のみです。

関連情報

[-d dbln オプション \[66 ページ\]](#)

[-a dbln オプション \[65 ページ\]](#)

1.4.3 Windows デバイス用の Mobile Link Listener オプション

次のオプションは、Mobile Link Listener の設定に使用できます。

オプション	説明
<code>@{ variable filename }</code>	指定された環境変数またはテキストファイルからの Mobile Link Listener オプションを適用します。
<code>-a value</code>	受信ライブラリの 1 つのライブラリオプションを指定します。
<code>-d filename</code>	受信ライブラリを指定します。
<code>-e device-name</code>	デバイス名を指定します。
<code>-f string</code>	デバイスに関する追加の情報を指定します。
<code>-g seconds</code>	IP トラッカーのポーリング間隔を指定します。
<code>-i seconds</code>	SMTP 接続のポーリング間隔を指定します。
<code>-l "keyword=value;..."</code>	メッセージハンドラの定義と作成を行います。
<code>-ls</code>	SMS 受信を有効にします。Windows Mobile の場合にのみ有効です。
<code>-lu</code>	UDP 受信を有効にします。
<code>-m</code>	メッセージのロギングを有効にします。

オプション	説明
<code>-ni</code>	IP 追跡を無効にします。
<code>-Oprefix</code>	ファイルに出力のログを取ります。
<code>-OSbytes</code>	メッセージログファイルの最大サイズを指定します。
<code>-Ofilename</code>	ファイルをトランケートし、そのファイルに出力を記録します。
<code>-p</code>	アイドル状態になったときにデバイスを自動的に停止できます。
<code>-pc {+ -}</code>	永続的な接続を有効または無効にします。
<code>-q</code>	Mobile Link Listener をクワイエットモードで実行します。
<code>-qi</code>	dblsn アイコンとメッセージウィンドウを非表示にします。
<code>-rfilename</code>	メッセージフィルタの応答アクションに関わるリモートデータベースを指定します。
<code>-SVscript-version</code>	認証に使用されるスクリプトバージョンを指定します。
<code>-t {+ -} name</code>	リモートデータベースのリモート ID の登録または登録解除を行います。
<code>-tSsession-name(session-option=option-value[;...])</code>	Mobile Link Listener のトレースセッションを設定します。
<code>-Uusername</code>	Mobile Link ユーザ名を指定します。
<code>-v {0 1 2 3}</code>	メッセージログの冗長性レベルを指定します。
<code>-Wpassword</code>	Mobile Link パスワードを指定します。
<code>-x {http https tcpip} [(protocol-option=value;...)]</code>	ネットワークプロトコルと、Mobile Link サーバのプロトコルオプションを指定します。
<code>-Ynewpassword</code>	新しい Mobile Link パスワードを指定します。

このセクションの内容:

[@data dblsn オプション \[65 ページ\]](#)

指定された環境変数またはテキストファイルからの Mobile Link Listener オプションを適用します。

[-a dblsn オプション \[65 ページ\]](#)

受信ライブラリの 1 つのライブラリオプションを指定します。

[-d dblsn オプション \[66 ページ\]](#)

受信ライブラリを指定します。

[-e dblsn オプション \[67 ページ\]](#)

デバイス名を指定します。

[-f dblsn オプション \[68 ページ\]](#)

デバイスに関する追加の情報を指定します。

[-gi dblsn オプション \[68 ページ\]](#)

IPトラッカーのポーリング間隔を指定します。

[-i dblsn オプション \[68 ページ\]](#)

SMTP 接続のポーリング間隔を指定します。

- l dblsn オプション [69 ページ]
メッセージハンドラの定義と作成を行います。
- ls dblsn オプション [70 ページ]
SMS 受信を有効にします。SMS リスナはセキュリティ保護されていないので、避けてください。
- lu dblsn オプション [70 ページ]
UDP 受信を有効にします。UDP リスナはセキュリティ保護されていないので、避けてください。
- m dblsn オプション [71 ページ]
メッセージのロギングを有効にします。
- ni dblsn オプション [71 ページ]
IP 追跡を無効にします。
- o dblsn オプション [72 ページ]
ファイルに出力のログを取ります。
- os dblsn オプション [72 ページ]
メッセージログファイルの最大サイズを指定します。
- ot dblsn オプション [72 ページ]
ファイルをトランケートし、そのファイルに出力を記録します。
- p dblsn オプション [73 ページ]
アイドル状態になったときにデバイスを自動的に停止できます。
- pc dblsn オプション [73 ページ]
永続的な接続を有効または無効にします。
- q dblsn オプション [74 ページ]
Mobile Link Listener をクワイエットモードで実行します。
- qi dblsn オプション [74 ページ]
dblsn アイコンとメッセージウィンドウを非表示にします。
- r dblsn オプション [75 ページ]
メッセージフィルタの応答アクションに関わるリモートデータベースを指定します。
- sv dblsn オプション [75 ページ]
認証に使用されるスクリプトバージョンを指定します。
- t dblsn オプション [76 ページ]
リモートデータベースのリモート ID の登録または登録解除を行います。
- ts dblsn オプション [76 ページ]
Mobile Link Listener のトレースセッションを設定します。
- u dblsn オプション [78 ページ]
Mobile Link Listener の Mobile Link ユーザ名を指定します。
- v dblsn オプション [78 ページ]
メッセージログの冗長性レベルを指定します。
- w dblsn オプション [79 ページ]
Mobile Link パスワードを指定します。
- x dblsn オプション [80 ページ]
ネットワークプロトコルと、Mobile Link サーバのプロトコルオプションを指定します。

[-y dblsn オプション \[80 ページ\]](#)

新しい Mobile Link パスワードを指定します。

1.4.3.1 @data dblsn オプション

指定された環境変数またはテキストファイルからの Mobile Link Listener オプションを適用します。

構文

```
dblsn @data ...
```

備考

デフォルトでは、パラメータなしで Mobile Link Listener を実行した場合の設定ファイルは `dblsn.txt` です。

同じ名前を持つファイルと環境変数が存在する場合は、環境変数が使用されます。

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

例

次の例では、`dblsnoptions` 環境変数からコマンドラインオプションを読み込みます。

```
dblsn @dblsnoptions
```

次の例では、同じ名前の環境変数がないことを前提として、`mydblsn.txt` からコマンドラインオプションを読み込みます。

```
dblsn @mydblsn.txt
```

1.4.3.2 -a dblsn オプション

受信ライブラリの 1 つのライブラリオプションを指定します。

構文

```
dblsn -a value ...
```

備考

デフォルトでは、ライブラリを指定しない場合、Mobile Link Listener は `lsn_udp17.dll` を使用します。他のライブラリまたは追加のライブラリを指定するには、`-d` オプションを使用します。

使用可能なライブラリオプションをすべて表示するには、`?` 値を使用します。

追加のライブラリオプションを設定するには、`-a` オプションを複数回使用します。

例

次の例では、ポートオプションを指定し、`lsn_udp17.dll` 受信ライブラリの `ShowSenderPort` オプションを宣言しています。

```
dblsn -d lsn_udp17.dll -a port=1234 -a ShowSenderPort
```

次の例では、2 つの異なるライブラリのポートオプションを指定しています。

```
dblsn -d lsn_udp17.dll -a port=1234 -d maac750.dll -a port=2345
```

次の例では、デフォルトのライブラリで使用できるライブラリオプションをすべて表示します。

```
dblsn -a ?
```

関連情報

[Windows デバイス用の受信ライブラリ \[61 ページ\]](#)

[-d dblsn オプション \[66 ページ\]](#)

1.4.3.3 -d dblsn オプション

受信ライブラリを指定します。

構文

```
dblsn -d filename ...
```

備考

デフォルトでは、Mobile Link Listener は `lsn_udp17.dll` 受信ライブラリを使用します。

複数の媒体での受信を可能にするマルチチャネル受信を有効にするには、`-d` オプションを複数回使用します。

例

次の例では、`maac750.dll` 受信ライブラリを指定しています。

```
dblsn -d maac750.dll
```

関連情報

[Windows デバイス用の受信ライブラリ \[61 ページ\]](#)

1.4.3.4 -e dblsn オプション

デバイス名を指定します。

構文

```
dblsn -e device-name ...
```

備考

デフォルトでは、デバイス名はオペレーティングシステムから自動的に生成されます。

Mobile Link サーバに接続するときは、すべてのデバイス名がユニークであることを確認してください。

デバイス名は次のものに限ってください。

- 英数字
- ピリオド
- コロン
- マイナス記号
- プラス記号
- アンダースコア

デバイス名は Mobile Link Listener ウィンドウで参照できます。

1.4.3.5 -f dblsn オプション

デバイスに関する追加の情報を指定します。

構文

```
dblsn -f string ...
```

備考

デフォルトでは、この情報はデバイス上で実行されているオペレーティングシステムのバージョン番号です。

1.4.3.6 -gi dblsn オプション

IPトラッカーのポーリング間隔を指定します。

構文

```
dblsn -gi number ...
```

備考

デフォルトでは、IPトラッカーは 60 秒ごとにポーリングします。

1.4.3.7 -i dblsn オプション

SMTP 接続のポーリング間隔を指定します。

構文

```
dblsn -i number ...
```

備考

-i オプションでは、Mobile Link Listener がメッセージをチェックする頻度を指定します。

SMTP 接続の場合、デフォルト値は 30 秒です。UDP 接続の場合、Mobile Link Listener はすぐに接続します。

-i オプションは、-d オプションで指定された受信ライブラリごとに 1 回使用できます。

例

次の例では、2 つの異なるライブラリのポーリング間隔を指定しています。

```
dblsn -d lsn_udp17.dll -i 60 -d maac750.dll -i 45
```

関連情報

[-d dblsn オプション \[66 ページ\]](#)

1.4.3.8 -l dblsn オプション

メッセージハンドラの定義と作成を行います。

構文

```
dblsn -l "keyword=value;..." ...
```

備考

Push 通知の追加のメッセージハンドラを定義するには、-l オプションを複数回使用します。メッセージハンドラは、指定した順序で処理されます。

関連情報

[メッセージハンドラ \[18 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

1.4.3.9 -ls dblsn オプション

SMS 受信を有効にします。SMS リスナはセキュリティ保護されていないので、避けてください。

構文

```
dblsn -ls ...
```

備考

デフォルトでは、SMS リスニングが表示されます。

関連情報

[Listener の配備のセキュリティ保護 \[61 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[-x dblsn オプション \[80 ページ\]](#)

1.4.3.10 -lu dblsn オプション

UDP 受信を有効にします。UDP リスナはセキュリティ保護されていないので、避けてください。

構文

```
dblsn -lu ...
```

備考

デフォルトでは、UDP リスニングが表示されます。

関連情報


[Listener の配備のセキュリティ保護 \[61 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[-x dblsn オプション \[80 ページ\]](#)

1.4.3.11 -m dblsn オプション

メッセージのロギングを有効にします。

 構文

```
dblsn -m ...
```

備考

デフォルトでは、メッセージのロギングはオフです。

1.4.3.12 -ni dblsn オプション

IP 追跡を無効にします。

 構文

```
dblsn -ni ...
```

備考

デフォルトでは、IP 追跡は有効です。

このオプションでは、配信確認は停止しません。

-ni オプションと -x オプションを一緒に使用すると、UDP アドレスのトラッキングが無効になります。この機能は、デバイストラッキングで UDP アドレスの更新を除外する場合に役立ちます。

関連情報

[-x dblsn オプション \[80 ページ\]](#)

1.4.3.13 -o dblsn オプション

ファイルに出力のログを取ります。

構文

```
dblsn -o filename ...
```

備考

デフォルトでは、出力は Mobile Link Listener ウィンドウに表示されます。

関連情報

[-ot dblsn オプション \[72 ページ\]](#)

1.4.3.14 -os dblsn オプション

メッセージログファイルの最大サイズを指定します。

構文

```
dblsn -os bytes ...
```

備考

デフォルトでは、最大サイズは無制限となります。最小のサイズ制限は 10000 です。

1.4.3.15 -ot dblsn オプション

ファイルをトランケートし、そのファイルに出力を記録します。

構文

```
dblsn -ot filename ...
```


備考

ファイルの内容が削除されてから、出力が記録されます。

関連情報

[-o dblsn オプション \[72 ページ\]](#)

1.4.3.16 -p dblsn オプション

アイドル状態になったときにデバイスを自動的に停止できます。

構文

```
dblsn -p ...
```

備考

デフォルトでは、Mobile Link Listener がデバイスの停止を防止します。

このオプションが適用されるのは、Windows Mobile のみです。

1.4.3.17 -pc dblsn オプション

永続的な接続を有効または無効にします。

構文

```
dblsn -pc { + | - } ...
```

備考

デフォルトでは、永続的接続は有効です。- フラグを指定すると、永続的接続が無効になります。+ フラグを指定すると、有効になります。

永続的接続を無効にすると、Mobile Link Listener は Push 通知を受信できなくなりますが、短命に終わった永続的接続がデバイストラッキングと確認用に有効になります。

永続的接続が切断された場合、Mobile Link Listener は継続的に再接続を試みます。

例

次の例では、永続的接続を無効にします。

```
dblsn -pc-
```

1.4.3.18 -q dblsn オプション

Mobile Link Listener をクワイエットモードで実行します。

構文

```
dblsn -q ...
```

備考

-q オプションを指定すると、Mobile Link Listener ウィンドウが最小化されます。デフォルトでは、Mobile Link Listener ウィンドウが表示されます。

1.4.3.19 -qi dblsn オプション

dblsn アイコンとメッセージウィンドウを非表示にします。

構文

```
dblsn -qi ...
```

備考

このオプションでは、dblsn の実行が視覚的に確認できないようにします。起動時のエラーのウィンドウは開く場合があります。-o で指定したログファイルを使用してエラーを診断できます。

関連情報

[-o dblsn オプション \[72 ページ\]](#)

1.4.3.20 -r dblsn オプション

メッセージフィルタの応答アクションに関わるリモートデータベースを指定します。

構文

```
dblsn -r filename ...
```

備考

`filename` には、RID ファイルのフルパスを指定する必要があります。このファイルは、最初に同期した後に `dbmlsync` によって自動的に作成されます。データベースファイルと同じロケーションと名前が使用されます。Ultra Light データベースの場合は、`filename` をデータベース名と同じにする必要があります。

`-r` オプションを適用すると、メッセージハンドラで `$remote_id action` 変数を使用して、RID ファイルのリモート ID を参照できます。デフォルトでは、リモート ID には GUID が使用されます。

複数のデータベースを識別するには、`-r` オプションを複数回使用します。

関連情報

[メッセージフィルタ \[19 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

1.4.3.21 -sv dblsn オプション

認証に使用されるスクリプトバージョンを指定します。

構文

```
dblsn -sv script-version ...
```

備考

デフォルトでは、ml_global サーバスクリプトバージョンが定義されていると、Mobile Link Listener はこのスクリプトバージョンを使用します。

1.4.3.22 -t dblsn オプション

リモートデータベースのリモート ID の登録または登録解除を行います。

構文

```
dblsln -t { + | - } name ...
```

備考

+ フラグを指定すると、リモート ID が登録されます。- フラグを指定すると、ID の登録が解除されます。

登録を行うと、Mobile Link Listener はそのリモート ID を参照して Push 通知を指定できます。

デバイストラッキング情報のアップロードに成功すると、登録された ID がサーバの ml_listening システムテーブルに保持されます。ID の登録が必要となるのは一度のみです。

複数の ID を登録または登録解除するには、-t オプションを複数回使用します。複数の ID を登録すると、複数のリモートデータベースに Push 通知を指定する場合に役立ちます。

関連情報

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

1.4.3.23 -ts dblsn オプション

Mobile Link Listener のトレースセッションを設定します。

構文

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

セッション名は **logging** にする必要があります。

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

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

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

備考

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

例

次に、-ts オプションの例を示します。

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

1.4.3.24 -u dblsn オプション

Mobile Link Listener の Mobile Link ユーザ名を指定します。

構文

```
dblsn -u username ...
```

備考

デフォルトでは、ユーザ名は `device-name-dblsn` です。`device-name` には、デバイスの名前を指定します。デバイス名は、`-e` オプションを使用して指定できます。

Mobile Link Listener では、ユーザ名を使用して Mobile Link サーバに接続し、デバイストラッキング、確認、永続的接続を行います。

ユーザ名には、Mobile Link サーバに登録されているユニークな Mobile Link ユーザ名を使用する必要があります。この名前は、統合データベースの `ml_user` システムテーブルに存在します。

関連情報

[-e dblsn オプション \[67 ページ\]](#)

[-w dblsn オプション \[79 ページ\]](#)

1.4.3.25 -v dblsn オプション

メッセージログの冗長性レベルを指定します。

構文

```
dblsn -v { 0 | 1 | 2 | 3 } ...
```

備考

デフォルトでは、冗長性レベルは 0 に設定されています。

次の表に、使用可能な冗長性レベル値の概要を示します。

冗長性レベル	説明
0	冗長性はオフです。
1	受信ライブラリメッセージ、基本的なアクショントレース段階、コマンドラインオプションを表示します。
2	レベル 1 の冗長性メッセージと、詳細なアクションのトレース段階を表示します。
3	レベル 2 の冗長性メッセージ、ポーリングステータス、受信ステータスを表示します。

メッセージログに Push 通知を出力するには、`-m` オプションを使用する必要があります。

関連情報

[-m dblsn オプション \[71 ページ\]](#)

1.4.3.26 -w dblsn オプション

Mobile Link パスワードを指定します。

構文

```
dblsn -w password ...
```

備考

パスワードは、関連する Mobile Link ユーザ名のもとで Mobile Link サーバに登録されている必要があります。

Mobile Link Listener では、パスワードを使用して Mobile Link サーバに接続し、デバイストラッキング、確認、永続的接続を行います。

関連情報

[-u dblsn オプション \[78 ページ\]](#)

1.4.3.27 -x dblsn オプション

ネットワークプロトコルと、Mobile Link サーバのプロトコルオプションを指定します。

構文

```
dblsn -x { http | https | tcpip } [ (protocol-option=value;...) ] ...
```

備考

Mobile Link サーバへの接続は、Mobile Link Listener がデバイストラッキング情報や配信確認を統合データベースに送信するために必要です。Mobile Link サーバのロケーションを指定するには、*host* プロトコルオプションを使用します。

-x オプションを指定すると、サーバのアドレスが変更された場合に、デバイスで統合データベースを更新できます。

1.4.3.28 -y dblsn オプション

新しい Mobile Link パスワードを指定します。

構文

```
dblsn -y newpassword ...
```

備考

リモートデバイスによるパスワードの変更が認証システムで許可されていない場合は、-y オプションを利用できません。

1.4.4 Windows デバイス用の Mobile Link Listener キーワード

次のキーワードを使用すると、`dblsn -l` オプションを使用して作成されたメッセージハンドラを設定できます。

フィルタのキーワード

Push 通知内のメッセージをフィルタするには、次のキーワードを使用します。

キーワードの構文	説明
<code>subject=subject-string</code>	件名のテキストが <code>subject-string</code> と同等の場合にメッセージをフィルタします。
<code>content=content-string</code>	内容のテキストが <code>content-string</code> と同等の場合にメッセージをフィルタします。
<code>message=message-string</code>	メッセージ全体のテキストが <code>message-string</code> と同等の場合にメッセージをフィルタします。
<code>message_start=message-string</code>	メッセージが <code>message-string</code> で始まる場合にメッセージをフィルタします。
<code>sender=sender-string</code>	メッセージの送信者が <code>sender-string</code> の場合にメッセージをフィルタします。

アクションのキーワード

フィルタ条件が満たされている場合にアクションを開始するには、次のキーワードを使用します。

キーワードの構文	説明
<code>action=command</code>	アクションコマンドを指定します。
<code>altaction=command</code>	アクションコマンドが失敗した場合に開始される代替アクションコマンドを指定します。

ポーリングのオプション

ライトウェイトポーラーを設定するには、次のオプションを使用します。

キーワードの構文	説明
<code>poll_connect={ http https tcpip }[(protocol-option=value;...)]</code>	サーバへの接続に必要なライトウェイトネットワークプロトコルオプションを指定します。デフォルト値は、 <code>dblsn -x</code> オプションから継承されます。
<code>poll_notifier=Notifier-string</code>	Push 要求を処理する Notifier の名前を指定します。必須。
<code>poll_key=key-string</code>	Notifier に Mobile Link Listener 自体を示すための Listener の名前を指定します。この値は、ユニークにする必要があります。必須。
<code>poll_every=seconds-number</code>	Mobile Link Listener がサーバをポーリングする頻度を指定します。間隔は秒単位で測定されます。デフォルト値は、Mobile Link サーバから自動的に取得されます。

オプション

次のオプションを使用すると、メッセージハンドラの動作を設定できます。

キーワードの構文	説明
<code>continue=[yes no]</code>	最初の一致を検出した後に Mobile Link Listener が受信を継続するかどうかを指定します。デフォルト値は <code>no</code> です。 <code>yes</code> 値を使用すると、複数のフィルタを指定するときに、1つのメッセージによって複数のアクションが開始される場合に役立ちます。
<code>confirm_action=[yes no]</code>	フィルタがアクションを確認するかどうかを指定します。デフォルト値は <code>yes</code> です。
<code>confirm_delivery=[yes no]</code>	<p>フィルタが条件付きのメッセージ配信を確認するかどうかを指定します。デフォルト値は <code>yes</code> です。したがって、最初のフィルタがメッセージを受け入れたときに配信確認が送信されます。</p> <p>メッセージの確認が必要で、フィルタがメッセージを受け入れた場合にのみ、配信を確認できます。指定したゲートウェイに、<code>yes</code> に設定された <code>confirm_delivery</code> キーワード値が定義されている場合は、メッセージに確認が必要です。<code>no</code> 値は、複数のフィルタが同一メッセージを受け入れる場合に、どのフィルタが配信確認をするかを細かく制御するために使用できます。</p> <p>Mobile Link Listener がアップロードした配信確認情報を処理するには、<code>confirmation_handler</code> イベントを使用します。</p>
<code>maydial=[yes no]</code>	アクションがモデムにダイヤル接続するパーミッションがあるかどうかを指定します。デフォルト値は <code>yes</code> です。 <code>no</code> 値を指定すると、Mobile Link Listener はアクションの前にモデムを解放します。

関連情報

[メッセージハンドラ \[18 ページ\]](#)

[-x dblsn オプション \[80 ページ\]](#)

[confirmation_handler イベント \[48 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

1.4.5 Windows デバイス用の Mobile Link Listener アクションコマンド

アクションは、新しいメッセージハンドラを設定する場合に指定されます。フィルタ条件が満たされると、アクションが開始されます。アクションが失敗した場合は、代替アクションが開始されます。アクションは、*action* キーワードを使用して定義されます。代替アクションは、*altaction* キーワードを使用して定義されます。

アクションコマンドのリストを次に示します。

コマンド	説明
<code>STARTprogramarglist</code>	バックグラウンドで Mobile Link Listener が実行されているときにプログラムを開始します。
<code>RUNprogramarglist</code>	Mobile Link Listener を一時停止してプログラムを実行します。
<code>POSTwindowmessage id to windowclass windowtitle</code>	ウィンドウクラスにウィンドウメッセージを送信します。
<code>SOCKET port=windowname[:host=hostname] [:sendText=text][:recvText=text[:timeout=seconds]]</code>	TCP/IP 接続を使用して、アプリケーションにメッセージを送信します。
<code>DBLSN FULL SHUTDOWN</code>	強制的に Mobile Link Listener をシャットダウンします。

action キーワードまたは *altaction* キーワードごとに指定できるアクションは 1 つのみです。1 つのアクションで複数のタスクを実行する場合、複数のコマンドを含むバッチファイルを作成し、*RUN* アクションコマンドを使用してファイルを実行します。

このセクションの内容:

[START アクションコマンド \[84 ページ\]](#)

バックグラウンドで Mobile Link Listener が実行されているときにプログラムを開始します。

[RUN アクションコマンド \[84 ページ\]](#)

Mobile Link Listener を一時停止してプログラムを実行します。

[POST アクションコマンド \[85 ページ\]](#)

ウィンドウクラスにウィンドウメッセージを送信します。

[SOCKET アクションコマンド \[86 ページ\]](#)

TCP/IP 接続を使用して、アプリケーションにメッセージを送信します。

[DBLSN FULL SHUTDOWN アクションコマンド \[87 ページ\]](#)

強制的に Mobile Link Listener をシャットダウンします。

関連情報

[アクションの開始 \[20 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

1.4.5.1 START アクションコマンド

バックグラウンドで Mobile Link Listener が実行されているときにプログラムを開始します。

構文

```
action='START program arglist'
```

備考

プログラムを起動すると、Mobile Link Listener は Push 通知の受信を再開します。

Mobile Link Listener はプログラムの終了を待機しません。したがって、アクションコマンドの実行に失敗したか、または指定したプログラムを起動できないかどうかのみを判定できます。

例

次の例では、コマンドラインオプションをいくつか使用して dbmlsync を起動します。オプションの一部は、\$content action 変数を使用してメッセージから取得されます。

```
dblsn -l "action='start dbmlsync.exe @dbmlsync.txt -n  
$content -wc dbmlsync_$content -e sch=INFINITE';"
```

1.4.5.2 RUN アクションコマンド

Mobile Link Listener を一時停止してプログラムを実行します。

構文

```
action='RUN program arglist'
```

備考

Mobile Link Listener は、プログラムの終了を待機してから受信を再開します。

プログラムを実行すると、Mobile Link Listener がプログラムを起動できない場合またはプログラムが 0 以外のリターンコードを返した場合に、Mobile Link Listener はプログラムの実行に失敗したかどうかを判定します。

例

次の例では、コマンドラインオプションをいくつか使用して dbmlsync を実行します。オプションの一部は、\$content 変数を使用してメッセージから取得されます。

```
dblsn -l "action='run dbmlsync.exe @dbmlsync.txt -n $content';"
```

1.4.5.3 POST アクションコマンド

ウィンドウクラスにウィンドウメッセージを送信します。

構文

```
action='POST windowmessage | id to windowclass | windowtitle'
```

備考

POST コマンドを使用すると、ウィンドウメッセージを使用するアプリケーションに通知できます。

ウィンドウメッセージは、メッセージの内容またはウィンドウメッセージ ID (存在する場合) によって識別できます。

ウィンドウクラスは、クラス名またはウィンドウタイトルによって識別できます。名前で識別する場合は、-wc dbmlsync オプションを使用すると、ウィンドウクラス名を指定できます。ウィンドウタイトルでウィンドウクラスを識別する場合は、最上位レベルのウィンドウのみでウィンドウクラスを参照できます。

ウィンドウメッセージまたはウィンドウクラス名に、スペースや句読点などの英数字以外の文字が含まれる場合は、テキストを一重引用符 (') で囲みます。また、エスケープ文字にも一重引用符を使用します。したがって、ウィンドウメッセージまたはウィンドウクラスに一重引用符が含まれる場合は、2 つの一重引用符 (") を使用して引用符を参照してください。

POST には次が有効です。

10 進 id による送信

例:post 999 to <wc|wt>

16 進 id による送信

例:post 0x3E7 to <wc|wt>

登録されたメッセージ名による送信

例:post myRegisteredMsgName to <wc|wt>

例

一重引用符が含まれるウィンドウとメッセージの使用法を示すため、次の例では mike's_message ウィンドウメッセージを mike's_class ウィンドウクラスに送信します。

```
dblsn -l "action='post mike's_message to mike's_class';"
```

次の例では、ウィンドウメッセージ dbas_synchronize を、クラス名 dbmlsync_FullSync で登録された dbmlsync インスタンスに送信します。

```
dblsn -l "action='post dbas_synchronize to dbmlsync_FullSync';"
```

1.4.5.4 SOCKET アクションコマンド

TCP/IP 接続を使用して、アプリケーションにメッセージを送信します。

構文

```
action='SOCKET port=windowname [;host=hostname] [;sendText=text] [;recvText=text [;timeout=seconds]]'
```

備考

SOCKET コマンドを使用して、実行中のアプリケーションに動的な情報を渡し、Java アプリケーションと Microsoft Visual Basic アプリケーションにメッセージを組み込みます。どちらの言語でも、カスタムウィンドウメッセージ機能はサポートされていません。また、Microsoft eEmbedded Visual Basic では、コマンドラインパラメータがサポートされていません。

ソケットに接続するには、ポートとホストを指定する必要があります。メッセージの入力には sendText を使用します。

アプリケーションが sendText の受信に成功したことを確認するときにメッセージを表示するには、recvText を使用します。recvText を使用すると、タイムアウト制限を指定できます。Mobile Link Listener が接続できない場合、受信確認を送信できない場合、またはタイムアウト制限内に確認を受信できない場合は、アクションの実行に失敗します。

例

次の例では、ポート 12345 で受信しているローカルアプリケーションに、\$sender=\$message で文字列を転送します。Mobile Link Listener では、確認としてアプリケーションが 5 秒以内に "beeperAck" を送信することが予期されます。

```
dblsn -l "action='socket port=12345;
sendText=$sender=$message;
recvText=beeperAck;
timeout=5'"
```

1.4.5.5 DBLSN FULL SHUTDOWN アクションコマンド

強制的に Mobile Link Listener をシャットダウンします。

構文

```
action='DBLSN FULL SHUTDOWN'
```

備考

停止すると、Mobile Link Listener は Push 通知の処理を停止し、デバイストラッキング情報の同期を停止します。サーバ起動同期を続行するには、Mobile Link Listener を再度起動する必要があります。

1.4.6 Windows デバイス用の Mobile Link Listener action 変数

アクションまたはフィルタでは、次の action 変数を使用できます。action 変数は、メッセージハンドラを開始する前に値に置き換えられます。

action 変数は、ドル記号 (\$) で始まります。エスケープ文字もドル記号であるため、1 つのドル記号をブレーステキストとして指定するには、2 つのドル記号 (\$\$) を使用します。

変数	説明
\$subject	メッセージの件名。
\$content	メッセージの内容。
\$message	件名、内容、送信者を含むメッセージ全体。
\$message_start	message_start フィルタキーワードで指定された、メッセージの先頭の一部。この変数を使用できるのは、message_start フィルタキーワードを指定した場合のみです。
\$message_end	message_start フィルタキーワードで除外された、メッセージの一部。この変数を使用できるのは、message_start フィルタキーワードを指定した場合のみです。
\$ml_connect	dblsn -x オプションによって指定された Mobile Link ネットワークプロトコルオプション。デフォルトは、空の文字列です。
\$ml_user	dblsn -u オプションによって指定された Mobile Link ユーザ名。デフォルト名は <code>device-name-dblsn</code> です。
\$ml_password	dblsn -w オプションによって指定される Mobile Link パスワード、または -y を使用した場合は新しい Mobile Link パスワード。
\$priority	この変数の意味は、carrier ライブラリに依存します。
\$request_id	Push 要求で指定された要求 ID。

変数	説明
\$remote_id	リモート ID。この変数は、dbsln -r オプションが指定された場合にだけ使用されます。
\$sender	メッセージの送信者。
\$type	この変数の意味は、carrier ライブラリに依存します。
\$year	この変数の意味は、carrier ライブラリに依存します。
\$month	この変数の意味は、carrier ライブラリに依存します。値は 1 ~ 12 までです。
\$day	この変数の意味は、carrier ライブラリに依存します。値は 1 ~ 31 までです。
\$hour	この変数の意味は、carrier ライブラリに依存します。値は 0 ~ 23 までです。
\$minute	この変数の意味は、carrier ライブラリに依存します。値は 0 ~ 59 までです。
\$second	この変数の意味は、carrier ライブラリに依存します。値は 0 ~ 59 までです。
\$best_adapter_mac	dbsln -x オプションによって指定された Mobile Link サーバに到達するための最善の NIC の MAC アドレス。最善のルートが NIC を経由しない場合、この変数の値は空文字列になります。
\$best_adapter_name	dbsln -x オプションによって指定された Mobile Link サーバに到達するための最善の NIC のアダプタ名。最善のルートが NIC を経由しない場合、この変数の値は空文字列になります。
\$best_ip	dbsln -x オプションによって指定された Mobile Link サーバに到達するための最善の IP インタフェースの IP アドレス。サーバが到達不能な場合、この変数の値は 0.0.0.0 になります。
\$best_network_name	dbsln -x オプションによって指定された Mobile Link サーバに到達するための最善のプロファイルの RAS またはダイヤルアッププロファイル名。最善のルートが RAS またはダイヤルアップ接続を経由しない場合、この変数の値は空文字列になります。
\$adapters	アクティブなネットワークアダプタ名のリストで、それぞれパイプ記号 () で分割します。
\$network_names	接続 RAS エントリ名のリストで、それぞれパイプ記号 () で分割します。RAS エントリ名は、ダイヤルアップネットワーク (DUN) のダイヤルアップエントリ名と呼ばれる場合もあります。
\$poll_connect	poll_connect ポーリングキーワードによって指定された Mobile Link ネットワークプロトコルオプション。デフォルトは、空の文字列です。
\$poll_notifier	poll_notifier ポーリングキーワードによって指定された Notifier の名前。
\$poll_key	poll_key ポーリングキーワードによって指定されたポーリングキー。
\$poll_every	poll_every ポーリングキーワードによって指定されたポーリング頻度。

例

次の例では、\$message_end action 変数を使用して、同期するパブリケーションを特定しています。

```
dblsn -l "message_start=start-of-message;action='run dbmlsync.exe -c ... -n $message_end'"
```

関連情報

[Push 要求 \[9 ページ\]](#)

[高度: フィルタとしてのリモート ID \[23 ページ\]](#)

[-x dblsn オプション \[80 ページ\]](#)

[-l dblsn オプション \[69 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[Windows デバイス用の Mobile Link Listener アクションコマンド \[83 ページ\]](#)

1.5 ライトウェイトポーリング API

ライトウェイトポーリング API は、ご使用のデバイスアプリケーションに統合できるプログラミングインタフェースです。ライトウェイトポーリング API には、サーバのポーリングに必要なメソッドが含まれています。

必要なファイル

すべてのディレクトリは、%SQLANY17% を基準とした相対ディレクトリです。次に、ライトウェイトポーリング API のコンパイルに必要なファイルのリストを示します。

ファイル名またはロケーション	説明
Bin32¥mllplib17.dll	ライトウェイトポーリング API ランタイムダイナミックライブラリ
SDK¥Lib¥x86¥mllplib17.lib および SDK¥Lib ¥x64¥mllplib17.lib	ライトウェイトポーリング API ランタイムインポートライブラリ
SDK¥Include¥mllplib.h	ライトウェイトポーリング API ヘッダファイル

C の SIS_CarDealer_LP_API サンプルアプリケーションは、%SQLANYSAMPI7%¥MobiLink ¥SIS_CarDealer_LP_API にあります。

API メンバ

メソッド	説明
MLLightPoller クラス	ライトウェイトポーラーオブジェクトを表します。
MLLPCreatePoller メソッド	MLLightPoller のインスタンスを作成します。
MLLPDestroyPoller メソッド	MLLightPoller のインスタンスを破棄します。

このセクションの内容:

[MLLightPoller クラス \[90 ページ\]](#)

ライトウェイトポーラーオブジェクトを表します。

[MLLPCreatePoller メソッド \[94 ページ\]](#)

MLLightPoller のインスタンスを作成します。

[MLLPDestroyPoller メソッド \[95 ページ\]](#)

MLLightPoller のインスタンスを破棄します。

1.5.1 MLLightPoller クラス

ライトウェイトポーラーオブジェクトを表します。

構文

```
public class MLLightPoller
```

メンバー

名前	説明
Poll メソッド	Notifier にキャッシュの Push 要求をチェックさせ、サーバをポーリングします。
SetConnectInfo メソッド	Mobile Link クライアントのストリームタイプとネットワークプロトコルオプションを設定します。
auth_status 列挙体	可能な認証ステータスコードを指定します。
return_code 列挙体	可能なリターンコードを指定します。

例

```
MLLightPoller * poller = MLLCreatePoller();
```

このセクションの内容:

[Poll メソッド \[91 ページ\]](#)

Notifier にキャッシュの Push 要求をチェックさせ、サーバをポーリングします。

[SetConnectInfo メソッド \[92 ページ\]](#)

Mobile Link クライアントのストリームタイプとネットワークプロトコルオプションを設定します。

[auth_status 列挙体 \[93 ページ\]](#)

可能な認証ステータスコードを指定します。

[return_code 列挙体 \[94 ページ\]](#)

可能なリターンコードを指定します。

1.5.1.1 Poll メソッド

Notifier にキャッシュの Push 要求をチェックさせ、サーバをポーリングします。

構文

```
public virtual return_code MLLightPoller::Poll(  
    const char * notifier,  
    const char * key,  
    char * subject = 0,  
    size_t * subjectSize = 0,  
    char * content = 0,  
    size_t * contentSize = 0  
)
```

パラメータ

Notifier

Notifier の名前。

key

Mobile Link Listener を識別するポーリングキーの名前。

subject

メッセージの件名を受け取るバッファ (NULL で終了)。

subjectSize

IN: 件名バッファのサイズ

OUT: 受信した件名のサイズ。ゼロが NULL 件名を示す NULL ターミネータを含みます。

content

メッセージの内容を受け取るバッファ (NULL で終了)。

contentSize

IN: 内容バッファのサイズ

OUT: 受信した内容のサイズ。ゼロが NULL 内容を示す NULL ターミネータを含みます。

戻り値

return_code 列挙体にリストされているコードの 1 つ。

備考

Mobile Link Listener は Notifier に接続し、Notifier がキャッシュで指定されたポーリングキー宛での Push 通知をチェックした後に、切断します。

関連情報

[return_code 列挙体 \[94 ページ\]](#)

1.5.1.2 SetConnectInfo メソッド

Mobile Link クライアントのストリームタイプとネットワークプロトコルオプションを設定します。

構文

```
public virtual return_code MLLightPoller::SetConnectInfo(  
    const char * streamName,  
    const char * streamParams  
);
```

パラメータ

streamName

使用するネットワークプロトコル。使用可能な値は、tcpip、http、https、または tls です。

streamParams

セミコロンで区切ったリスト形式のプロトコルオプション文字列。

戻り値

return_code 列挙体にリストされているコードの1つ。

例

```
poller_ret = poller->SetConnectInfo("http", "host=localhost;port=80;")
```

関連情報

[return_code 列挙体 \[94 ページ\]](#)

1.5.1.3 auth_status 列挙体

可能な認証ステータスコードを指定します。

構文

```
public typedef enum MLLightPoller::auth_status;
```

メンバー

メンバー名	説明
AUTH_EXPIRED	認証の有効期限が切れています。
AUTH_IN_USE	ユーザ名はすでに認証されています。
AUTH_INVALID	認証が無効です。
AUTH_UNKNOWN	認証が不明です。
AUTH_VALID	認証が有効です。
AUTH_VALID_BUT_EXPIRES_SOON	認証は有効ですが、まもなく失効します。

1.5.1.4 return_code 列挙体

可能なリターンコードを指定します。

構文

```
public typedef enum MLLightPoller::return_code;
```

メンバー

名前	説明
AUTH_FAILED	Mobile Link サーバへの接続が認証されませんでした。
BAD_STREAM_NAME	認識されないストリーム名。
BAD_STREAM_PARAM	ストリームパラメータを解析できませんでした。
COMMUNICATION_ERROR	通信エラーにより失敗しました。
CONNECT_FAILED	Mobile Link サーバに接続できませんでした。
CONTENT_OVERFLOW	内容バッファが小さすぎます。
KEY_NOT_FOUND	指定した Notifier から指定したキーの Push 通知がありません。
NYI	内部でのみ使用。
OK	メソッドが正常に実行されました。
SUBJECT_OVERFLOW	件名バッファが小さすぎます。

1.5.2 MLLPCreatePoller メソッド

MLLightPoller のインスタンスを作成します。

構文

```
extern MLLightPoller * MLLPCreatePoller()
```

戻り値

新しい MLLightPoller オブジェクト

 例

```
poller = MLLPCreatePoller();
```

関連情報

[MLLPDestroyPoller メソッド \[95 ページ\]](#)

1.5.3 MLLPDestroyPoller メソッド

MLLightPoller のインスタンスを破棄します。

 構文

```
extern void MLLPDestroyPoller(  
    MLLightPoller * poller  
)
```

パラメータ

poller

破棄する MLLightPoller。

備考

このメソッドには、NULL MLLightPoller オブジェクトを指定できます。

 例

```
MLLPDestroyPoller (poller);
```

関連情報

[MLLPCreatePoller メソッド \[94 ページ\]](#)

1.6 サーバ起動同期のシステムプロシージャ

サーバ起動同期のシステムプロシージャは、Mobile Link システムテーブル内のローの追加と削除を実行します。

i 注記

これらのシステムプロシージャは、デバイストラッキングに使用します。自動デバイストラッキングをサポートするリモートデバイスを使用する場合、これらのシステムプロシージャを使用する必要はありません。自動デバイストラッキングをサポートしないリモートデバイスを使用する場合、これらのシステムプロシージャを使用して、手動のデバイストラッキングを設定できます。

このセクションの内容:

[ml_delete_device システムプロシージャ \[97 ページ\]](#)

手動でデバイストラッキングを設定している場合、リモートデバイスに関するすべての情報を削除します。

[ml_delete_device_address システムプロシージャ \[97 ページ\]](#)

手動でデバイストラッキングを設定している場合、デバイスのアドレスを削除します。

[ml_delete_listening システムプロシージャ \[98 ページ\]](#)

手動でデバイストラッキングを設定している場合、Mobile Link ユーザとリモートデバイス間のマッピングを削除します。

[ml_set_device システムプロシージャ \[99 ページ\]](#)

手動でデバイストラッキングを設定している場合、リモートデバイスに関する情報を追加または変更します。
ml_device テーブル内のローを追加または更新します。

[ml_set_device_address システムプロシージャ \[100 ページ\]](#)

手動でデバイストラッキングを設定している場合、リモートデバイスアドレスに関する情報を追加または変更します。
ml_device_address テーブル内のローを追加または更新します。

[ml_set_listening システムプロシージャ \[102 ページ\]](#)

手動でデバイストラッキングを設定している場合、Mobile Link ユーザとリモートデバイス間のマッピングを追加または変更します。ml_listening テーブル内のローを追加または更新します。

[ml_set_sis_sync_state システムプロシージャ \[103 ページ\]](#)

Mobile Link 同期ステータスを ml_sis_sync_state システムテーブルに記録します。

関連情報

[デバイストラッキングゲートウェイ \[27 ページ\]](#)

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

1.6.1 ml_delete_device システムプロシージャ

手動でデバイストラッキングを設定している場合、リモートデバイスに関するすべての情報を削除します。

パラメータ

項目	パラメータ	説明
1	@device	VARCHAR(255)。デバイス名。

備考

この機能は、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

デバイスレコードとこれを参照するすべての関連レコードを削除します。

```
CALL ml_delete_device('myOldDevice');
```

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

1.6.2 ml_delete_device_address システムプロシージャ

手動でデバイストラッキングを設定している場合、デバイスのアドレスを削除します。

パラメータ

項目	パラメータ	説明
1	@device	VARCHAR(255)
2	@medium	VARCHAR(255)

備考

このシステムプロシージャは、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

アドレスレコードを削除します。

```
CALL ml_delete_device_address('myWindowsMobile', 'ROGERS AT&T');
```

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

1.6.3 ml_delete_listening システムプロシージャ

手動でデバイストラッキングを設定している場合、Mobile Link ユーザとリモートデバイス間のマッピングを削除します。

パラメータ

項目	パラメータ	説明
1	@name	VARCHAR(128)

備考

このシステムプロシージャは、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

受信者レコードを削除します。

```
CALL ml_delete_listening('myULDB');
```

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

1.6.4 ml_set_device システムプロシージャ

手動でデバイストラッキングを設定している場合、リモートデバイスに関する情報を追加または変更します。ml_device テーブル内のローを追加または更新します。

パラメータ

項目	パラメータ	説明
1	@device	VARCHAR(255)。ユーザが定義したユニークなデバイス名。
2	@listener_version	VARCHAR(128)。Mobile Link Listener のバージョンに関するオプションの注釈。
3	@listener_protocol	INTEGER。バージョン 9.0.0 の場合は 0、または 9.0.0 以降の Windows デバイス用 MobileLink Listener の場合は 2 を使用します。
4	@info	VARCHAR(255)。オプションのデバイス情報。
5	@ignore_tracking	VARCHAR(1)。トラッキングを無視し、手動で入力したデータがトラッキングによって書きされないようにする場合、y に設定します。
6	@source	VARCHAR(255)。このレコードのソースにあるオプションの注釈。

備考

システムプロシージャ ml_set_device、ml_set_device_address、ml_set_listening は、Mobile Link システムテーブル ml_device、ml_device_address、ml_listening にある情報を変更することで自動デバイストラッキングを無効にするのに使用します。

このシステムプロシージャは、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

各デバイスについて、デバイスレコードを追加します。

```
CALL ml_set_device(  
  'myWindowsMobile',  
  'MobiLink Listeners for myWindowsMobile - 9.0.1',  
  '1',  
  'not used',  
  'y',  
  'manually entered by administrator'  
);
```

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

[ml_set_device_address システムプロシージャ \[100 ページ\]](#)

[ml_set_listening システムプロシージャ \[102 ページ\]](#)

1.6.5 ml_set_device_address システムプロシージャ

手動でデバイストラッキングを設定している場合、リモートデバイスアドレスに関する情報を追加または変更します。
ml_device_address テーブル内のローを追加または更新します。

パラメータ

項目	パラメータ	説明
1	@device	VARCHAR(255)。既存のデバイス名。
2	@medium	VARCHAR(255)。ネットワークプロバイダ ID (Carrier の network_provider_id プロパティと一致する必要があります)。
3	@address	VARCHAR(255)。SMS 対応デバイスの電話番号。
4	@active	VARCHAR(1)。Push 通知の送信に使用するためにこのレコードをアクティブにする場合は、y に設定します。

項目	パラメータ	説明
5	@ignore_tracking	VARCHAR(1)。トラッキングを無視し、手動で入力したデータがトラッキングによって上書きされないようにする場合、y に設定します。
6	@source	VARCHAR(255)。このレコードのソースにあるオプションの注釈。

備考

システムプロシージャ ml_set_device、ml_set_device_address、ml_set_listening は、Mobile Link システムテーブル ml_device、ml_device_address、ml_listening にある情報を変更することで自動デバイストラッキングを無効にするのに使用します。

このシステムプロシージャは、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

各デバイスについて、デバイスのアドレスレコードを追加します。

```
CALL ml_set_device_address(
  'myWindowsMobile',
  'ROGERS AT&T',
  '3211234567',
  'y',
  'y',
  'manually entered by administrator'
);
```

関連情報

[9.0.0 Mobile Link Listener のデバイストラッキングの設定 \[28 ページ\]](#)

[ml_set_device システムプロシージャ \[99 ページ\]](#)

[ml_set_listening システムプロシージャ \[102 ページ\]](#)

1.6.6 ml_set_listening システムプロシージャ

手動でデバイストラッキングを設定している場合、Mobile Link ユーザとリモートデバイス間のマッピングを追加または変更します。ml_listening テーブル内のローを追加または更新します。

パラメータ

項目	パラメータ	説明
1	@name	VARCHAR(128)。Mobile Link ユーザ名です。
2	@device	VARCHAR(255)。既存のデバイス名です。
3	@listening	VARCHAR(1)。DeviceTracker のアドレス設定に使用するためにこのレコードをアクティブにする場合、y に設定します。
4	@ignore_tracking	VARCHAR(1)。トラッキングを無視し、手動で入力したデータがトラッキングによって上書きされないようにする場合、y に設定します。
5	@source	VARCHAR(255)。このレコードのソースにあるオプションの注釈。

備考

システムプロシージャ ml_set_device、ml_set_device_address、ml_set_listening は、Mobile Link システムテーブル ml_device、ml_device_address、ml_listening にある情報を変更することで自動デバイストラッキングを無効にするのに使用します。

このシステムプロシージャは、デバイストラッキングを手動で設定する場合にだけ役立ちます。

例

各リモートデータベースについて、デバイスの受信者レコードを追加します。これは、デバイスを Mobile Link ユーザ名にマッピングします。

```
CALL ml_set_listening(  
    'myULDB',  
    'myWindowsMobile',  
    'y',  
    'y',  
    'manually entered by administrator'  
);
```

関連情報

9.0.0 Mobile Link Listener のデバイストラッキングの設定 [28 ページ]

ml_set_device システムプロシージャ [99 ページ]

ml_set_device_address システムプロシージャ [100 ページ]

1.6.7 ml_set_sis_sync_state システムプロシージャ

Mobile Link 同期ステータスを ml_sis_sync_state システムテーブルに記録します。

パラメータ

項目	パラメータ	説明
1	@remote_id	VARCHAR(128)
2	@subscription_id	VARCHAR(128)
3	@publication_name	VARCHAR(128)
4	@user_name	VARCHAR(128)
5	@last_upload	TIMESTAMP
6	@last_download	TIMESTAMP

備考

publication_nonblocking_download_ack イベントで ml_set_sis_sync_state システムプロシージャを呼び出すと、ユーザは ml_sis_sync_state テーブルを参照する request_cursor イベントを作成できます。

例

次のスクリプトでは、publication_nonblocking_download_ack イベントスクリプトを指定して、同期ステータスを記録しています。

```
CALL ml_set_sis_sync_state(  
  {ml s.remote_id},  
  NULL,  
  {ml s.publication_name},  
  {ml s.username},  
  NULL,  
  {ml s.last_publication_download}  
);
```

1.7 高度: メッセージ構文

ライトウェイトポーリング (デフォルト)、UDP ゲートウェイ、SYNC ゲートウェイには、次のメッセージ構文が適用されます。

```
message = [subject]content
```

SMTP ゲートウェイを使用して送信されるメッセージは、次のいずれかの構文構造をしています。

- `message = sender[subject]content`
- `message = sender(subject)content`
- `message = sender{subject}content`
- `message = sender<subject>content`
- `message = sender'subject'content`
- `message = sender"subject"content`

正しいメッセージ構文と `sender` の電子メールアドレス構文は、各自の無線通信事業者によって異なります。メッセージ構文を判定するには、メッセージのログギングを有効にして Mobile Link Listener を実行します。このとき、冗長性レベルは `dblsn` の `-m` オプションと `-v` オプションを使用して 2 に設定します。最初に Mobile Link Listener を実行したときに、メッセージログには正しい構文が記録されています。

デバイストラッキングゲートウェイを使用する場合、メッセージ構文はメッセージの送信に使用する従属ゲートウェイによって異なります。SMTP 従属ゲートウェイを使用する場合、構文は各自の公衆無線通信事業者によって異なります。

備考

大カッコ、シェブロン、二重引用符、カッコ、一重引用符、角カッコは、内部使用のために予約されています。件名内では使用しないでください。

このセクションの内容:

[高度: sa_send_udp システムプロシージャを使用した Push 通知の送信 \[105 ページ\]](#)

SQL Anywhere 統合データベースで `sa_send_udp` システムプロシージャを使用すると、UDP ゲートウェイ経由でデバイスに Push 通知を送信できます。この方法は、Notifier を使用して Push 通知を送信する方法の代替となる手段です。

関連情報

[ゲートウェイと Carrier \[26 ページ\]](#)

[Windows デバイス用の Mobile Link Listener キーワード \[81 ページ\]](#)

[-m dblsn オプション \[71 ページ\]](#)

[-v dblsn オプション \[78 ページ\]](#)

[Push 要求の使用法 \[12 ページ\]](#)

1.7.1 高度: sa_send_udp システムプロシージャを使用した Push 通知の送信

SQL Anywhere 統合データベースで sa_send_udp システムプロシージャを使用すると、UDP ゲートウェイ経由でデバイスに Push 通知を送信できます。この方法は、Notifier を使用して Push 通知を送信する方法の代替となる手段です。

前提条件

- デバイスに Mobile Link Listener が設定され、Push 通知を受信できるようになります
- デバイスに Microsoft Internet Explorer がインストールされます
- デバイス上で次のコマンドが実行されました

```
dblsn -l "message=RunBrowser;action='START iexplore.exe http://www.sap.com';"
```

- SQL Anywhere 統合データベースが Mobile Link サーバ上で稼働されます

コンテキスト

元のメッセージの最後に **1** を追加して、このメッセージを sa_send_udp システムプロシージャの msg 引数で使用すると、元のメッセージが Mobile Link Listener に送信されます。

手順

1. Interactive SQL を実行し、次などのコマンドを使用し統合データベースに接続します。その際には、consdb-source-name を統合データベースの ODBC 名と置き換えます。

```
dbisql -c "dsn=consdb-source-name"
```

2. 次の文を実行して、Push 通知を送信します。

```
CALL sa_send_udp('device-ip-address', 5001, 'RunBrowser1')
```

最初の引数によって、Push 通知は必ず正しいデバイスに送信されます。device-ip-address は、デバイスの IP アドレスに置き換えます。Mobile Link サーバと同じコンピュータで Mobile Link Listener が実行されている場合は、localhost を使用してください。

2 番目の引数はポート番号です。デフォルトでは、Mobile Link Listener はポート 5001 を使用して UDP を受信します。

3番目の引数は、最後に 1 を追加して送信するメッセージです。予約済みのサーバ起動同期プロトコルである 1 を追加すると、UDP ゲートウェイを使用して *RunBrowser* メッセージがデバイスに送信されます。

結果

システム呼び出しが実行されると、*RunBrowser* メッセージがデバイスに送信され、そのデバイスで Microsoft Internet Explorer が起動して SAP ホームページがロードされます。

1.8 サーバ起動同期チュートリアル

サーバ起動同期の使用方法についての理解を深めるには、次のチュートリアルを使用してください。

1. チュートリアル: ライトウェイトポーリングを使用したサーバ起動同期の設定 [106 ページ]

このチュートリアルでは、サーバ起動同期を使用できるように SQL Anywhere 統合データベースとリモートデータベースを設定する方法について説明します。このチュートリアルは、`%SQLANYSAMP17%`¥MobiLink ¥SIS_CarDealer_LP_DBLSN に配置されているサンプルコードに基づいています。

2. チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 [120 ページ]

このチュートリアルでは、サーバ起動同期を使用できるように SQL Anywhere 統合データベースとリモートデータベースを設定する方法について説明します。このチュートリアルは、`%SQLANYSAMP17%`¥MobiLink ¥SIS_CarDealer に配置されているサンプルコードに基づいています。

1.8.1 チュートリアル: ライトウェイトポーリングを使用したサーバ起動同期の設定

このチュートリアルでは、サーバ起動同期を使用できるように SQL Anywhere 統合データベースとリモートデータベースを設定する方法について説明します。このチュートリアルは、`%SQLANYSAMP17%`¥MobiLink ¥SIS_CarDealer_LP_DBLSN に配置されているサンプルコードに基づいています。

前提条件

Mobile Link イベントスクリプトの基本的な知識が必要です。

次のソフトウェアが必要です。

- SQL Anywhere 17

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

サーバ起動同期の実装サンプルは、`%SQLANYSAMPI7%¥MobiLink` にあります。サーバ起動同期のすべてのサンプルディレクトリ名には、プレフィックスの `SIS_` が付いています。

i 注記

SQL Central を使用してリモートデータベースを管理し、その後、ライトウェイトポーリングを使用するサーバ起動同期への代替機能としてサーバ起動リモートタスク (SIRT) を使用できます。

このチュートリアルでは、次のことを学習します。

- SQL Anywhere 統合データベースをサーバ起動同期用に設定します。
- サーバ側プロパティを設定します。
- サーバ起動同期を要求する Push 要求を発行します。

このセクションの内容:

[レッスン 1: 統合データベースのセットアップ \[108 ページ\]](#)

このレッスンでは、`dbinit` ユーティリティを使用して、同期に必要なスクリプトで

`SIS_CarDealer_LP_DBLSN_CONDB` という名前の統合データベースを作成します。その後、SQL Anywhere 17 ドライバを使用して、`SIS_CarDealer_LP_DBLSN_CONDB` データベース用の ODBC データソースを定義します。

[レッスン 2: データベーススキーマの生成 \[109 ページ\]](#)

このレッスンでは、1 つのデータベーススキーマを生成します。このスキーマには、Dealer テーブル、`non_sync_request` テーブル、`download_cursor` 同期スクリプトが含まれます。このデータベーススキーマは、Push 要求を生成するための稼働条件を満たしています。

[レッスン 3: Mobile Link プロジェクトの作成 \[111 ページ\]](#)

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。

[レッスン 4: Notifier の設定 \[112 ページ\]](#)

このレッスンでは、Notifier イベントを設定して、Notifier が Push 要求を作成する方法と Push 通知をデバイスに送信する方法を定義します。

[レッスン 5: Mobile Link サーバの起動 \[114 ページ\]](#)

このレッスンでは、デバイスに Push 通知を送信できるように Notifier を使用して Mobile Link サーバを起動します。

[レッスン 6: リモートデータベースの設定 \[115 ページ\]](#)

このレッスンでは、SQL Anywhere リモートデータベースを作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。

[レッスン 7: Mobile Link Listener の設定 \[116 ページ\]](#)

このレッスンでは、Mobile Link Listener オプションをテキストファイルに保存してから、コマンドラインでファイル名を指定して `dblsn` を実行し、Mobile Link Listener を設定します。

レッスン 8: Push 要求の発行 [118 ページ]

このレッスンでは、Dealer テーブルに変更を加え、サーバ起動同期を要求します。

レッスン 9: クリーンアップ [119 ページ]

チュートリアルをコンピュータから削除します。

タスクの概要: [サーバ起動同期チュートリアル \[106 ページ\]](#)

次のタスク: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

関連情報

[サーバ起動同期 \[4 ページ\]](#)

1.8.1.1 レッスン 1: 統合データベースのセットアップ

このレッスンでは、`dbinit` ユーティリティを使用して、同期に必要なスクリプトで `SIS_CarDealer_LP_DBLSN_CONDB` という名前の統合データベースを作成します。その後、SQL Anywhere 17 ドライバを使用して、`SIS_CarDealer_LP_DBLSN_CONDB` データベース用の ODBC データソースを定義します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

1. 統合データベースを格納する新しい作業ディレクトリを作成します。このチュートリアルでは、`c:\¥MLsis` を作業ディレクトリとします。
2. `dbinit` ユーティリティを使用して SQL Anywhere 統合データベースを作成し、DBA ユーザ ID を DBA に設定し、パスワードを `sql` に設定します。
3. `c:\¥MLsis` ディレクトリに移動し、次のコマンドを実行します。

```
dbinit -dba DBA,passwd SIS_CarDealer_LP_DBLSN_CONDB
```

4. 統合データベースを開始するには、次のコマンドを実行します。

```
dbsrv17 SIS_CarDealer_LP_DBLSN_CONDB
```

5. ▶ スタート ▶ プログラム ▶ SQL Anywhere17 ▶ 管理ツール ▶ ODBC データソースアドミニストレータ ▶ をクリックします。
6. ユーザ DSN タブをクリックしてから、追加をクリックします。
7. データソースの新規作成ウィンドウで、SQL Anywhere17 をクリックし、完了をクリックします。
8. SQL Anywhere の ODBC 設定ウィンドウで、次の操作を行います。
 - a. ODBC タブをクリックします。
 - b. データソース名フィールドに SIS_CarDealer_LP_DBLSN_CONDB と入力します。
 - c. ログインタブをクリックします。
 - d. ユーザ ID フィールドに、DBA と入力します。
 - e. パスワードフィールドに、passwd と入力します。
 - f. アクションドロップダウンリストから、このコンピュータで稼働しているデータベースに接続を選択します。
 - g. サーバ名フィールドに、SIS_CarDealer_LP_DBLSN_CONDB と入力します。
 - h. OK をクリックします。
9. ODBC データソースアドミニストレータを閉じます。

ODBC データソースアドミニストレータウィンドウで OK をクリックします。

結果

統合データベースが作成され、ODBC データソースが定義されます。

次のステップ

次のレッスンに進みます。

1.8.1.2 レッスン 2: データベーススキーマの生成

このレッスンでは、1つのデータベーススキーマを生成します。このスキーマには、Dealer テーブル、non_sync_request テーブル、download_cursor 同期スクリプトが含まれます。このデータベーススキーマは、Push 要求を生成するための稼働条件を満たしています。

前提条件

このチュートリアルのものでこのレッスンを完了している必要があります。

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

手順

1. ▶ スタート ▶ プログラム ▶ SQL Anywhere17 ▶ 管理ツール ▶ SQL Central ▶ をクリックします。
2. 次のタスクを実行して、統合データベースに接続します。
 - a. ▶ 接続 ▶ SQL Anywhere17 に接続 ▶ をクリックします。
 - b. アクションドロップダウンリストから、ODBC データソースを使用した接続を選択します。
 - c. ODBC データソース名をクリックし、参照をクリックします。
 - d. SIS_CarDealer_LP_DBLSN_CONDB を選択し、OK をクリックします。
 - e. 接続をクリックします。
3. Interactive SQL を使用してデータベースに接続します。

Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

- SQL Central から Interactive SQL を起動するには、SIS_CarDealer_LP_DBLSN_CONDB - DBA データベースを右クリックし、Interactive SQL を開くをクリックします。
- コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

```
dbisql -c "dsn=SIS_CarDealer_LP_DBLSN_CONDB"
```

4. 次の SQL 文を実行し、Dealer テーブルと non_sync_request テーブルを作成して設定します。

```
CREATE TABLE Dealer (
  name          VARCHAR(10) NOT NULL PRIMARY KEY,
  rating        VARCHAR(5),
  last_modified TIMESTAMP DEFAULT TIMESTAMP
)
CREATE TABLE non_sync_request(
  poll_key     VARCHAR(128)
)
```

5. 次の文を使用して、Dealer テーブルにデータを挿入します。

```
INSERT INTO Dealer(name, rating) VALUES ('Audi', 'a');
INSERT INTO Dealer(name, rating) VALUES ('Buick', 'b');
INSERT INTO Dealer(name, rating) VALUES ('Chrysler', 'c');
INSERT INTO Dealer(name, rating) VALUES ('Dodge', 'd');
INSERT INTO Dealer(name, rating) VALUES ('Eagle', 'e');
INSERT INTO Dealer(name, rating) VALUES ('Ford', 'f');
INSERT INTO Dealer(name, rating) VALUES ('Geo', 'g');
INSERT INTO Dealer(name, rating) VALUES ('Honda', 'h');
INSERT INTO Dealer(name, rating) VALUES ('Isuzu', 'I');
COMMIT;
```

6. 次の SQL 文を実行して Mobile Link のシステムテーブルとストアプロシージャを作成します。C:¥Program Files¥SQL Anywhere 17¥ は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

```
READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql"
```

7. 次の SQL スクリプトを実行し、download_cursor 同期スクリプトを指定して ml_sis_sync_state システムテーブルに同期ステータスを記録します。

```
CALL ml_add_table_script(
```

```

    'CarDealer',
    'Dealer',
    'download_cursor',
    'SELECT * FROM Dealer WHERE last_modified >= ?'
);
CALL ml_add_connection_script(
    'CarDealer',
    'publication_nonblocking_download_ack',
    'CALL ml_set_sis_sync_state(
        {ml s.remote_id},
        NULL,
        {ml s.publication_name},
        {ml s.username},
        NULL,
        {ml s.last_publication_download}
    )'
);
CALL ml_add_table_script(
    'CarDealer', 'Dealer', 'download_delete_cursor', '--{ml_ignore}'
);
COMMIT;

```

このスクリプトによって、ダウンロード専用同期を記録するように ml_sis_sync_state が設定されます。同期ステータスを記録すると、request_cursor イベントから ml_sis_sync_state システムテーブルを参照できます。次のレッスンでは request_cursor イベントを指定します。

8. Interactive SQL を閉じます。

結果

データベーススキーマが作成されます。

次のステップ

次のレッスンに進みます。

1.8.1.3 レッスン 3: Mobile Link プロジェクトの作成

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。

前提条件

このチュートリアルこれまでのレッスンを完了している必要があります。

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

手順

1. ▶ スタート ▶ プログラム ▶ SQL Anywhere17 ▶ 管理ツール ▶ SQL Central ▶ をクリックします。
2. ▶ ツール ▶ Mobile Link 17 ▶ 新しいプロジェクト ▶ をクリックします。
3. 名前フィールドに SIS_CarDealer_LP_DBLSN_CONDB_project と入力します。
4. ロケーションフィールドに C:¥MLsis と入力し、次へをクリックします。
5. データベースの表示名フィールドに SIS_CarDealer_LP_DBLSN_CONDB と入力します。
6. 編集をクリックします。汎用 ODBC データベースに接続ウィンドウが表示されます。
7. ユーザ ID フィールドに、DBA と入力します。
8. パスワードフィールドに、passwd と入力します。
9. ODBC データソース名フィールドで、参照をクリックして SIS_CarDealer_LP_DBLSN_CONDB を選択します。
10. OK をクリックし、保存をクリックします。
11. パスワードを記憶オプションをオンにし、次へをクリックします。
12. リモートデータベーススキーマページで、同期させる Dealer テーブルのみを選択します。次へをクリックします。
13. 次へを再度クリックしてから、完了をクリックします。OK をクリックします。

結果

Mobile Link プロジェクトが作成されます。

次のステップ

次のレッスンに進みます。

1.8.1.4 レッスン 4: Notifier の設定

このレッスンでは、Notifier イベントを設定して、Notifier が Push 要求を作成する方法と Push 通知をデバイスに送信する方法を定義します。

前提条件

このチュートリアルこれまでのレッスンを完了している必要があります。

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

コンテキスト

request_cursor イベントスクリプトによって、Push 要求が検出されます。各 Push 要求によって、送信される情報と情報を受信するデバイスが決まります。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**SIS_CarDealer_LP_DBLSN_CONDB_project**、**統合データベース**、**SIS_CarDealer_LP_DBLSN_CONDB - DBA** の順に展開します。
2. **通知**を右クリックし、**新規** > **Notifier** をクリックします。
3. 新しい **Notifier** の名前を指定してください。フィールドに **CarDealerNotifier** と入力します。
4. **完了**をクリックします。
5. 右ウィンドウ枠で **CarDealerNotifier** を選択し、**プロパティ**をクリックします。
6. **イベントタブ**をクリックし、イベントリストから **request_cursor** をクリックします。
7. 表示されたテキストフィールドで次の SQL 文を入力します。

```
SELECT ml_sis_sync_state.remote_id + '.sync' FROM ml_sis_sync_state
WHERE
(
  EXISTS (SELECT 1 FROM Dealer
          WHERE last_modified >= ml_sis_sync_state.last_download)
)
```

8. **OK** をクリックして Notifier イベントを保存します。

結果

Notifier が作成され、設定されます。

次のステップ

次のレッスンに進みます。

関連情報

[request_cursor イベント \[44 ページ\]](#)

[ml_set_sis_sync_state システムプロシージャ \[103 ページ\]](#)

1.8.1.5 レッスン 5: Mobile Link サーバの起動

このレッスンでは、デバイスに Push 通知を送信できるように Notifier を使用して Mobile Link サーバを起動します。

前提条件

このチュートリアル of これまでのレッスンを完了している必要があります。

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

手順

統合データベースに接続します。

次のコマンドを実行します。

```
mlsrv17 -notifier -c "dsn=SIS_CarDealer_LP_DBLSN_CONDB" -o serverOut.txt -v+ -dl -zu  
+ -x tcpip
```

次の表は、このレッスンで使用する mlsrv17 オプションを示します。オプション -o、-v、は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v は運用環境では使用しません。

オプション	説明
-notifier	サーバ起動同期用に有効なすべての Notifier を起動します。
-c	接続文字列を指定します。
-o	メッセージログファイル serverOut.txt を指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとプロトコルオプションを設定します。

結果

Mobile Link サーバメッセージウィンドウが表示されます。Notifier は、デバイスから Push 要求を受信する準備が整ったことを示します。

次のステップ

次のレッスンに進みます。

1.8.1.6 レッスン 6: リモートデータベースの設定

このレッスンでは、SQL Anywhere リモートデータベースを作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。

前提条件

このチュートリアルこれまでのレッスンを完了している必要があります。

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

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

c:¥MLsis ディレクトリから次のコマンドを実行します。

```
dbinit -dba DBA,passwd SIS_CarDealer_LP_DBLSN_REM
```

2. dbsrv17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

次のコマンドを実行します。

```
dbsrv17 SIS_CarDealer_LP_DBLSN_REM
```

3. Interactive SQL を使用して Mobile Link クライアントデータベースに接続します。

次のコマンドを実行します。

```
dbisql -c "SERVER=SIS_CarDealer_LP_DBLSN_REM;UID=DBA;PWD=passwd"
```

4. リモートデータベースに Dealer テーブルを作成します。

Interactive SQL で次の SQL 文を実行します。

```
CREATE TABLE Dealer (  
    name          VARCHAR(10) NOT NULL PRIMARY KEY,  
    rating        VARCHAR(5),  
    last_modified  TIMESTAMP DEFAULT TIMESTAMP  
)  
COMMIT;
```

5. Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

Interactive SQL で次の SQL 文を実行します。

```
CREATE PUBLICATION CarDealer(TABLE DEALER WHERE 0=1)
CREATE SYNCHRONIZATION USER test_mluser OPTION ScriptVersion='CarDealer'
CREATE SYNCHRONIZATION SUBSCRIPTION TO CarDealer FOR test_mluser
SET OPTION public.ml_remote_id = 'remote_id';
COMMIT;
```

結果

SQL Anywhere リモートデータベース、同期パブリケーション、ユーザ、サブスクリプションがすべて作成されます。

次のステップ

次のレッスンに進みます。

1.8.1.7 レッスン 7: Mobile Link Listener の設定

このレッスンでは、Mobile Link Listener オプションをテキストファイルに保存してから、コマンドラインでファイル名を指定して dblsn を実行し、Mobile Link Listener を設定します。

前提条件

このチュートリアル of これまでのレッスンを完了している必要があります。

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

手順

1. 次のコマンドを実行して Mobile Link サーバと同期し、SIS_CarDealer_LP_DBLSN_REM.rid ファイルを作成します。

```
dbmlsync -c "SERVER=SIS_CarDealer_LP_DBLSN_REM;UID=DBA;PWD=passwd" -e sa=on -o
reml.txt -v+
```

Mobile Link Listener は、\$remote_id action 変数を使用してポーリングキーを定義できます。このキーは、Mobile Link サーバでデバイスの識別に使用されます。この変数は、リモート ID ファイル SIS_CarDealer_LP_DBLSN_REM.rid から取得します。このファイルは、Mobile Link サーバと初期同期するときに作成されます。リモート ID ファイルを使用する場合は、Mobile Link サーバと同期する必要があります。

- SQL Anywhere Mobile Link クライアントウィンドウでシャットダウンをクリックします。
- 次の内容のテキストファイルを作成して、Mobile Link Listener コマンドファイルを作成します。

```
# Verbosity level
-v2
# Show notification messages in console and log
-m
# Truncate, then write output to dblsn.txt
-ot dblsn.txt
# Remote ID file (defining the scope of $remote_id)
-r SIS_CarDealer_LP_DBLSN_REM.rid
# Message handlers
# Signal dbmlsync to launch, sync and then shutdown
-l "poll_connect='tcpip(host=localhost)';
    poll_notifier=CarDealerNotifier;
    poll_key=$remote_id.sync;
    action='run dbmlsync.exe -c
SERVER=SIS_CarDealer_LP_DBLSN_REM;UID=DBA;PWD=passwd -e sa=on -o rem1.txt -v+';"
```

- このチュートリアルでは、`c:\¥MLsis` をサーバ側コンポーネントの作業ディレクトリとします。テキストファイルを `mydblsn.txt` という名前でこのディレクトリに保存します。
- Mobile Link Listener を起動します。

コマンドプロンプトで、`c:\¥MLsis` に移動するか、Mobile Link Listener コマンドファイルが保存されているディレクトリに移動します。

次のコマンドを実行して、Mobile Link Listener を起動します。

```
dblsn @mydblsn.txt
```

Mobile Link Listener がスリープ中であることを示す *MobiLink Listener for Windows* ウィンドウが表示されます。

結果

Mobile Link Listener が設定されます。

次のステップ

次のレッスンに進みます。

関連情報

[Listener \[17 ページ\]](#)

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dblsn\) \[59 ページ\]](#)

[@data dblsn オプション \[65 ページ\]](#)

[action 変数 \[21 ページ\]](#)

1.8.1.8 レッスン 8: Push 要求の発行

このレッスンでは、Dealer テーブルに変更を加え、サーバ起動同期を要求します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

コンテキスト

このレッスンでは、統合データベースの Dealer テーブルを変更して、Mobile Link Listener が Push 通知をポーリングするときに情報をリモートデータベースにダウンロードできるようにします。次に、統合データベースにポーリングキー値を挿入して、サーバ起動同期を要求します。Notifier は request_cursor イベントを実行し、non_sync_request テーブル内のポーリングキーを検出して Mobile Link Listener に Push 通知を送信します。Mobile Link Listener が Push 通知を受信すると、Mobile Link データベースと同期してリモートデータベースを更新します。

手順

1. 統合データベースに接続していない場合は、Interactive SQL で接続します。

次のコマンドを実行します。

```
dbisql -c "dsn=SIS_CarDealer_LP_DBLSN_CONDB"
```

2. 次の SQL 文を実行します。

```
UPDATE Dealer  
  SET RATING = 'B' WHERE name = 'Geo';  
COMMIT;
```

3. 同期が発生するまで数秒待ちます。

Mobile Link Listener は、統合データベースをポーリングして Push 通知をダウンロードし、リモートデータベースの Dealer テーブルを更新します。

4. リモートデータベースで Dealer テーブルが更新されたことを確認します。

次の SQL 文を実行します。

```
SELECT * FROM Dealer
```

Geo の評価が B になっている必要があります。

結果

統合データベースに変更が加えられ、サーバ起動同期が開始されます。

次のステップ

次のレッスンに進みます。

関連情報

[Push 要求の使用方法 \[12 ページ\]](#)

1.8.1.9 レッスン 9: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

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

コマンドプロンプトで次のコマンドを入力します。

```
odbcad32
```

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

結果

チュートリアルがコンピュータから削除されます。

次のステップ

次のレッスンに進みます。

1.8.2 チュートリアル: ゲートウェイを使用したサーバ起動同期の設定

このチュートリアルでは、サーバ起動同期を使用できるように SQL Anywhere 統合データベースとリモートデータベースを設定する方法について説明します。このチュートリアルは、`%SQLANYSAMP17%¥MobiLink¥SIS_CarDealer` に配置されているサンプルコードに基づいています。

前提条件

Mobile Link イベントスクリプトの基本的な知識が必要です。

次のソフトウェアが必要です。

- SQL Anywhere17

統合データベースで次のロールおよび権限を持つ必要があります。

- SYS_AUTH_RESOURCE_ROLE 互換ロール
- MONITOR システム権限

リモートデータベースで次のロールおよび権限を持つ必要があります。

- SYS_REPLICATION_ADMIN_ROLE システムロール
- SYS_RUN_REPLICATION_ROLE システムロール

コンテキスト

サーバ起動同期のいくつかの実装サンプルは、`%SQLANYSAMP17%¥MobiLink` にあります。サーバ起動同期のすべてのサンプルディレクトリ名には、プレフィックスの `SIS_` が付いています。

このチュートリアルでは、次のことを学習します。

- SQL Anywhere 統合データベースをサーバ起動同期用に設定します。
- サーバ側プロパティを設定します。
- サーバ起動同期を要求する Push 要求を発行します。

1. [レッスン 1: 統合データベースのセットアップ \[122 ページ\]](#)
このレッスンでは、dbinit ユーティリティを使用して、同期に必要なスクリプトで **MLconsolidated** という名前の統合データベースを作成します。データベース用の ODBC データソースを定義します。
2. [レッスン 2: データベーススキーマの生成 \[123 ページ\]](#)
このレッスンでは、データベーススキーマを生成します。このスキーマには、Dealer テーブルと download_cursor 同期スクリプトが含まれます。テーブルとストアードプロシージャは、サーバ起動同期の Push 要求を生成するために使用されます。
3. [レッスン 3: Push 要求を格納するテーブルの作成 \[125 ページ\]](#)
このレッスンでは、Push 要求を格納する Push 要求テーブルを作成します。Notifier は、Push 要求を検出すると、デバイスにメッセージを送信します。
4. [レッスン 4: Mobile Link プロジェクトの作成 \[126 ページ\]](#)
このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。
5. [レッスン 5: Notifier の設定 \[128 ページ\]](#)
このレッスンでは、Notifier で Push 要求を作成し、要求を Mobile Link Listener に送信し、有効期限が切れた要求を削除する方法を定義する、3 つの Notifier イベントを設定します。
6. [レッスン 6: ゲートウェイと Carrier の設定 \[131 ページ\]](#)
ゲートウェイは、メッセージを送信するためのメカニズムです。
7. [レッスン 7: Mobile Link サーバの起動 \[132 ページ\]](#)
このレッスンでは、デバイスに Push 通知を送信できるように Notifier を使用して Mobile Link サーバを起動します。
8. [レッスン 8: リモートデータベースの設定 \[133 ページ\]](#)
このレッスンでは、SQL Anywhere リモートデータベースを作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。
9. [レッスン 9: Mobile Link Listener の設定 \[135 ページ\]](#)
このレッスンでは、Mobile Link Listener オプションをテキストファイルに保存してから、コマンドラインでファイル名を指定して dblsn を実行し、Mobile Link Listener を設定します。
10. [レッスン 10: Push 要求の発行 \[136 ページ\]](#)
Push 要求を発行するには、直接 PushRequest テーブルを移植するか、Dealer テーブルで変更を加えます。
11. [レッスン 11: クリーンアップ \[138 ページ\]](#)
チュートリアルをコンピュータから削除します。

タスクの概要: [サーバ起動同期チュートリアル \[106 ページ\]](#)

前のタスク: [チュートリアル: ライトウェイトポーリングを使用したサーバ起動同期の設定 \[106 ページ\]](#)

関連情報

[サーバ起動同期 \[4 ページ\]](#)

1.8.2.1 レッスン 1: 統合データベースのセットアップ

このレッスンでは、dbinit ユーティリティを使用して、同期に必要なスクリプトで **MLconsolidated** という名前の統合データベースを作成します。データベース用の ODBC データソースを定義します。

前提条件

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

手順

1. 統合データベースを格納する新しい作業ディレクトリを作成します。
このチュートリアルでは、`c:\MLsis` を作業ディレクトリとします。
2. dbinit ユーティリティを使用して SQL Anywhere 統合データベースを起動します。
3. 次のコマンドを実行します。

```
dbinit -dba DBA,passwd MLconsolidated
```

4. dbsrv17 ユーティリティを使用して統合データベースを起動します。

次のコマンドを実行します。

```
dbsrv17 MLconsolidated
```

5. **▶ スタート ▶ プログラム ▶ SQL Anywhere17 ▶ 管理ツール ▶ ODBC データソースアドミニストレータ** をクリックします。
6. **ユーザ DSN** タブをクリックしてから、**追加** をクリックします。
7. **データソースの新規作成** ウィンドウで、**SQL Anywhere17** をクリックし、**完了** をクリックします。
8. **SQL Anywhere の ODBC 設定** ウィンドウで、次の操作を行います。
 - a. **ODBC** タブをクリックします。
 - b. **データソース名** フィールドに **sis_cons** と入力します。
 - c. **ログイン** タブをクリックします。
 - d. **ユーザ ID** フィールドに、**DBA** と入力します。
 - e. **パスワード** フィールドに、**passwd** と入力します。
 - f. **アクション** ドロップダウンリストから、**このコンピュータで稼働しているデータベースに接続** を選択します。
 - g. **サーバ名** フィールドに、**MLconsolidated** と入力します。
 - h. **OK** をクリックします。
9. ODBC データソースアドミニストレータを閉じます。
ODBC データソースアドミニストレータウィンドウで **OK** をクリックします。

結果

統合データベースが作成され、ODBC データソースが定義されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

次のタスク: [レッスン 2: データベーススキーマの生成 \[123 ページ\]](#)

1.8.2.2 レッスン 2: データベーススキーマの生成

このレッスンでは、データベーススキーマを生成します。このスキーマには、Dealer テーブルと download_cursor 同期スクリプトが含まれます。テーブルとストアプロシージャは、サーバ起動同期の Push 要求を生成するために使用されます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

1. [▶ スタート ▶ プログラム ▶ SQL Anywhere17 ▶ 管理ツール ▶ SQL Central ▶](#)をクリックします。
2. 次のタスクを実行して、統合データベースに接続します。
 - a. [▶ 接続 ▶ SQL Anywhere17 に接続 ▶](#)をクリックします。
 - b. ユーザ ID フィールドに、**DBA** と入力します。
 - c. パスワードフィールドに、**passwd** と入力します。
 - d. アクションドロップダウンリストから、**ODBC データソースを使用した接続**をクリックします。
 - e. **ODBC データソース名**をクリックし、**参照**をクリックします。
 - f. **sis_cons** を選択し、**OK** をクリックします。
 - g. **接続**をクリックします。
3. Interactive SQL を使用してデータベースに接続します。

Interactive SQL は、SQL Central またはコマンドプロンプトから起動できます。

- SQL Central から Interactive SQL を起動するには、**MLconsolidated - DBA** データベースを右クリックし、**Interactive SQL を開く**をクリックします。
- コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

```
dbisql -c "dsn=sis_cons"
```

4. 次の SQL 文を実行し、Dealer テーブルを作成して設定します。

```
CREATE TABLE Dealer (  
    name VARCHAR(10) NOT NULL PRIMARY KEY,  
    rating VARCHAR(5),  
    last_modified TIMESTAMP DEFAULT TIMESTAMP  
)
```

5. 次の文を使用して、Dealer テーブルにデータを挿入します。

```
INSERT INTO Dealer(name, rating) VALUES ('Audi', 'a');  
INSERT INTO Dealer(name, rating) VALUES ('Buick', 'b');  
INSERT INTO Dealer(name, rating) VALUES ('Chrysler', 'c');  
INSERT INTO Dealer(name, rating) VALUES ('Dodge', 'd');  
INSERT INTO Dealer(name, rating) VALUES ('Eagle', 'e');  
INSERT INTO Dealer(name, rating) VALUES ('Ford', 'f');  
INSERT INTO Dealer(name, rating) VALUES ('Geo', 'g');  
INSERT INTO Dealer(name, rating) VALUES ('Honda', 'h');  
INSERT INTO Dealer(name, rating) VALUES ('Isuzu', 'I');  
COMMIT;
```

6. 次の SQL スクリプトを実行して Mobile Link のシステムテーブルとストアードプロシージャを作成します。*C:¥Program Files¥SQL Anywhere 17¥* は、SQL Anywhere17 インストール環境のロケーションに置き換えてください。

```
READ "C:¥Program Files¥SQL Anywhere 17¥MobiLink¥setup¥syncsa.sql"
```

7. 次の SQL スクリプトを実行し、download_cursor 同期スクリプトを指定して同期を記録します。

```
CALL ml_add_table_script(  
    'sis_ver1',  
    'Dealer',  
    'download_cursor',  
    'SELECT * FROM Dealer WHERE last_modified >= ?'  
);  
CALL ml_add_table_script(  
    'sis_ver1', 'Dealer', 'download_delete_cursor', '--{ml_ignore}'  
);  
COMMIT
```

Interactive SQL は閉じないでください。

結果

Dealer テーブルと download_cursor 同期スクリプトを含むデータベーススキーマが生成され、Mobile Link のシステムテーブルとストアードプロシージャがインストールされます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 1: 統合データベースのセットアップ \[122 ページ\]](#)

次のタスク: [レッスン 3: Push 要求を格納するテーブルの作成 \[125 ページ\]](#)

1.8.2.3 レッスン 3: Push 要求を格納するテーブルの作成

このレッスンでは、Push 要求を格納する Push 要求テーブルを作成します。Notifier は、Push 要求を検出すると、デバイスにメッセージを送信します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

1. 前のレッスンで Interactive SQL によってデータベースに接続されているはずです。

データベースに接続していない場合は、SQL Central またはコマンドプロンプトで Interactive SQL を起動します。

- SQL Central から Interactive SQL を起動するには、**MLconsolidated - DBA** データベースを右クリックし、**Interactive SQL を開く**をクリックします。
- コマンドプロンプトで Interactive SQL を起動するには、次のコマンドを実行します。

```
dbisql -c "dsn=sis_cons"
```

2. Interactive SQL で次の SQL 文を実行します。

```
CREATE TABLE PushRequest (  
  req_id INTEGER DEFAULT AUTOINCREMENT PRIMARY KEY,  
  mluser VARCHAR(128),  
  subject VARCHAR(128),  
  content VARCHAR(128),  
  resend_interval VARCHAR(30) DEFAULT '20s',  
  time_to_live VARCHAR(30) DEFAULT '1m',  
  status VARCHAR(128) DEFAULT 'created'  
)  
COMMIT;
```

3. Interactive SQL を閉じます。

結果

Push 要求を格納する Push 要求テーブルが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 2: データベーススキーマの生成 \[123 ページ\]](#)

次のタスク: [レッスン 4: Mobile Link プロジェクトの作成 \[126 ページ\]](#)

関連情報

[Push 要求 \[9 ページ\]](#)

[サーバ起動同期 \[4 ページ\]](#)

[サーバ起動同期のコンポーネント \[6 ページ\]](#)

1.8.2.4 レッスン 4: Mobile Link プロジェクトの作成

このレッスンでは、新しい Mobile Link プロジェクトを作成して、統合データベースに接続します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

1. *SQL Central* で、**ツール** ▶ *Mobile Link17* ▶ **新しいプロジェクト** をクリックします。
2. 名前フィールドに **sis_cons_project** と入力します。
3. ロケーションフィールドに **C:¥MLsis** と入力し、**次へ** をクリックします。
4. データベースの表示名フィールドに **sis_cons** と入力します。
5. **編集** をクリックします。汎用 *ODBC* データベースに接続ウィンドウが表示されます。
6. ユーザ ID フィールドに、**DBA** と入力します。
7. パスワードフィールドに、**passwd** と入力します。
8. *ODBC* データソース名フィールドで、**参照** をクリックして **sis_cons** を選択します。
9. **OK** をクリックし、**保存** をクリックします。
10. **パスワードを記憶オプション** をオンにし、**次へ** をクリックします。
11. **新しいリモートデータベーススキーマページ** でデフォルトをそのまま使用し、**次へ** をクリックします。
12. **次へ** をクリックしてから、**完了** をクリックします。**OK** をクリックします。

結果

Mobile Link プロジェクトが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 3: Push 要求を格納するテーブルの作成 \[125 ページ\]](#)

次のタスク: [レッスン 5: Notifier の設定 \[128 ページ\]](#)

1.8.2.5 レッスン 5: Notifier の設定

このレッスンでは、Notifier で Push 要求を作成し、要求を Mobile Link Listener に送信し、有効期限が切れた要求を削除する方法を定義する、3 つの Notifier イベントを設定します。

前提条件

このチュートリアルの前までのレッスンを完了している必要があります。

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

コンテキスト

Notifier は、統合データベース内の変更を検出し、begin_poll イベントを使用して Push 要求を作成します。この場合、変更が Dealer テーブルで発生し、リモートデータベースが最新でない場合、begin_poll スクリプトは、PushRequest テーブルを設定します。

request_cursor スクリプトは、Push 要求をフェッチします。各 Push 要求により、メッセージで送信される情報、情報を受信するリモートデータベースが決まります。

request_delete Notifier イベントはクリーンアップ処理を指定します。このスクリプトを使用すると、暗黙に除外された要求、期限が切れた要求が自動的に削除されます。

手順

1. SQL Central の左ウィンドウ枠の *Mobile Link 17* で、**sis_cons_project**、**統合データベース**、**sis_cons** の順に展開します。
2. **通知** を右クリックし、**新規** > **Notifier** をクリックします。
3. 新しい **Notifier** の名前を指定してくださいフィールドに **CarDealerNotifier** と入力します。
4. **完了** をクリックします。
5. begin_poll イベントスクリプトを入力します。
 - a. 右ウィンドウ枠で **CarDealerNotifier** を選択し、**ファイル** > **プロパティ** をクリックします。
 - b. **イベントタブ** をクリックします。
 - c. イベントリストから **begin_poll** を選択します。
 - d. 表示されたテキストフィールドで次の SQL 文を入力します。

```
--  
-- Insert the last consolidated database  
-- modification date into @last_modified  
--  
DECLARE @last_modified timestamp;  
SELECT MAX(last_modified) INTO @last_modified FROM Dealer;
```



```

--
-- Delete processed requests if the mluser is up-to-date
--
DELETE FROM PushRequest
  FROM PushRequest AS p, ml_user AS u, ml_subscription AS s
  WHERE p.status = 'processed'
        AND u.name = p.mluser
        AND u.user_id = s.user_id
        AND @last_modified <= GREATER(s.last_upload_time,
s.last_download_time);
--
-- Insert new requests when a device is not up-to-date
--
INSERT INTO PushRequest(mluser, subject, content)
SELECT u.name, 'sync', 'ignored'
  FROM ml_user as u, ml_subscription as s
  WHERE u.name IN (SELECT name FROM ml_listening WHERE listening = 'y')
        AND u.user_id = s.user_id
        AND @last_modified > greater(s.last_upload_time, s.last_download_time)
        AND u.name NOT LIKE '%-dblsn'
        AND NOT EXISTS(SELECT * FROM PushRequest
  WHERE PushRequest.mluser = u.name
        AND PushRequest.subject = 'sync')

```

begin_poll スクリプトの最初の主要セクションでは、デバイスが最新の状態でない場合は、PushRequest テーブルからの処理済み要求は削除されます。

```
@last_modified <= GREATER(s.last_upload_time, s.last_download_time)
```

@last_modified は、統合データベース Dealer テーブルで最大の変更が行われた日です。式 greater(s.last_upload_time, s.last_download_time) は、リモートデータベースの最後の同期時間を表します。

request_delete イベントを使用して、直接 Push 要求を削除することもできます。ただし、この場合に begin_poll イベントを使用すると、リモートデータベースが同期する前に、同期期限が切れた要求や暗黙的に除外された要求が削除されないようにすることができます。

コードの次のセクションは、Dealer テーブルの last_modified カラムに加えられた変更をチェックし、ml_listening テーブルにリストされた最新の状態でないすべてのアクティブ Mobile Link Listener に対して Push 要求を発行します。

```
@last_modified > GREATER(s.last_upload_time, s.last_download_time)
```

PushRequest テーブルが移植されると、begin_poll スクリプトが件名を 'sync' に設定します。

6. request_cursor スクリプトを入力します。
 - a. イベントリストから *request_cursor* をクリックします。
 - b. 表示されたテキストフィールドで次の SQL 文を入力します。

```

SELECT
  p.req_id,
  'Default-DeviceTracker',
  p.subject,
  p.content,
  p.mluser,
  p.resend_interval,
  p.time_to_live
FROM PushRequest AS p

```

PushRequest テーブルによって、request_cursor スクリプトにローが入力されます。

順序と request_cursor 結果セットの値は重要です。たとえば、2 番目のパラメータは、デフォルトのゲートウェイ Default-DeviceTracker を定義します。デバイストラッキングゲートウェイは、ユーザへのアクセス方法を追跡し、UDP または SMTP を自動的に選択してリモートデバイスに接続します。

7. request_delete スクリプトを入力します。
 - a. イベントリストから *request_delete* をクリックします。
 - b. 表示されたテキストフィールドで次の SQL 文を入力します。

```
UPDATE PushRequest SET status='processed' WHERE req_id = ?
```

request_delete スクリプトは、ローを削除するのではなく、PushRequest テーブルのローのステータスを 'processed' に更新します。

8. **適用** をクリックしてから **OK** をクリックして Notifier イベントを保存します。

結果

3 つの Notifier イベントが定義されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 4: Mobile Link プロジェクトの作成 \[126 ページ\]](#)

次のタスク: [レッスン 6: ゲートウェイと Carrier の設定 \[131 ページ\]](#)

関連情報

[デバイストラッキングゲートウェイ \[27 ページ\]](#)

[request_delete イベント \[46 ページ\]](#)

[begin_poll イベント \[40 ページ\]](#)

[request_cursor イベント \[44 ページ\]](#)

[request_delete イベント \[46 ページ\]](#)

1.8.2.6 レッスン 6: ゲートウェイと Carrier の設定

ゲートウェイは、メッセージを送信するためのメカニズムです。

前提条件

このチュートリアル of これまでのレッスンを完了している必要があります。

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

コンテキスト

サポートされるゲートウェイまたはデバイストラッキングゲートウェイを定義できます。デバイストラッキングゲートウェイを指定すると、Mobile Link サーバではクライアントへのアクセス方法を追跡して、最適なゲートウェイを自動的に選択します。

手順

このチュートリアルでは、デフォルトのデバイストラッキングゲートウェイを使用するため、設定は必要ありません。

結果

デフォルトゲートウェイが使用されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 5: Notifier の設定 \[128 ページ\]](#)

次のタスク: [レッスン 7: Mobile Link サーバの起動 \[132 ページ\]](#)

関連情報

[ライトウェイトポーラーの代替手段としてのゲートウェイ \[26 ページ\]](#)

[ゲートウェイと Carrier \[26 ページ\]](#)

[デバイストラッキングゲートウェイプロパティ \[55 ページ\]](#)

1.8.2.7 レッスン 7: Mobile Link サーバの起動

このレッスンでは、デバイスに Push 通知を送信できるように Notifier を使用して Mobile Link サーバを起動します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

統合データベースに接続します。

次のコマンドを実行します。

```
mlsrv17 -notifier -c "dsn=sis_cons" -o serverOut.txt -v+ -dl -zu+ -x tcpip
```

次の表は、このレッスンで使用する mlsrv17 オプションを示します。オプション -o、-v、は、デバッグとトラブルシューティングの情報を提供します。これらのロギングオプションは、開発環境での使用に適しています。パフォーマンス上の理由から、一般的に -v は運用環境では使用しません。

オプション	説明
-notifier	サーバ起動同期用に有効なすべての Notifier を起動します。
-c	接続文字列を指定します。
-o	メッセージログファイル serverOut.txt を指定します。
-v+	ログを取る対象となる情報を指定します。-v+ を使用して、最大冗長ロギングをオンに設定します。
-zu+	自動的に新しいユーザを追加します。
-x	Mobile Link クライアントの通信プロトコルとプロトコルオプションを設定します。

結果

Mobile Link サーバメッセージウィンドウが表示されます。Notifier は、デバイスから Push 要求を受信する準備が整ったことを示します。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 6: ゲートウェイと Carrier の設定 \[131 ページ\]](#)

次のタスク: [レッスン 8: リモートデータベースの設定 \[133 ページ\]](#)

1.8.2.8 レッスン 8: リモートデータベースの設定

このレッスンでは、SQL Anywhere リモートデータベースを作成し、同期パブリケーション、ユーザ、サブスクリプションを作成します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

1. dbinit コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

次のコマンドを実行します。

```
dbinit -dba DBA,passwd remotel
```

2. dbsrv17 コマンドラインユーティリティを使用して、Mobile Link クライアントデータベースを起動します。

次のコマンドを実行します。

```
dbsrv17 remotel
```

- Interactive SQL を使用して Mobile Link クライアントデータベースに接続します。

次のコマンドを実行します。

```
dbisql -c "SERVER=remotel;UID=DBA;PWD=passwd"
```

- Dealer テーブルを作成します。

Interactive SQL で次の SQL 文を実行します。

```
CREATE TABLE Dealer (  
  name          VARCHAR(10) NOT NULL PRIMARY KEY,  
  rating        VARCHAR(5),  
  last_modified  TIMESTAMP DEFAULT TIMESTAMP  
)  
COMMIT;
```

- Mobile Link 同期ユーザ、パブリケーション、サブスクリプションを作成します。

Interactive SQL で次の SQL 文を実行します。

```
CREATE PUBLICATION car_dealer_pub (table Dealer);  
CREATE SYNCHRONIZATION USER sis_user1;  
CREATE SYNCHRONIZATION SUBSCRIPTION  
  TO car_dealer_pub  
  FOR sis_user1  
  OPTION scriptversion='sis_ver1';  
COMMIT;
```

結果

SQL Anywhere リモートデータベース、同期パブリケーション、ユーザ、サブスクリプションが作成されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 7: Mobile Link サーバの起動 \[132 ページ\]](#)

次のタスク: [レッスン 9: Mobile Link Listener の設定 \[135 ページ\]](#)

1.8.2.9 レッスン 9: Mobile Link Listener の設定

このレッスンでは、Mobile Link Listener オプションをテキストファイルに保存してから、コマンドラインでファイル名を指定して dblsn を実行し、Mobile Link Listener を設定します。

前提条件

このチュートリアルの前までのレッスンを完了している必要があります。

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

手順

1. 次の内容のテキストファイルを作成して、Mobile Link Listener コマンドファイルを作成します。

```
#-----  
# Verbosity level  
-v2  
# Show notification messages in console and log  
-m  
# Polling interval, in seconds  
-i 3  
# Truncate, then write output to dblsn.txt  
-ot dblsn.txt  
# MobiLink address and connect parameter for dblsn  
-x "host=localhost"  
# Enable device tracking and specify the MobiLink user name.  
-t+ sis_user1  
# Message handlers  
# Synchronize using dbmlsync  
-l "subject=sync;  
action='start dbmlsync.exe  
-c SERVER=remote1;UID=DBA;PWD=passwd  
-o dbmlsyncOut.txt  
';"
```

2. このチュートリアルでは、`c:\¥MLsis` をサーバ側コンポーネントの作業ディレクトリとします。テキストファイルを `mydblsn.txt` という名前でこのディレクトリに保存します。
3. Mobile Link Listener を起動します。

コマンドプロンプトで、`c:\¥MLsis` に移動するか、Mobile Link Listener コマンドファイルが保存されているディレクトリに移動します。

次のコマンドを実行して、Mobile Link Listener を起動します。

```
dblsn @mydblsn.txt
```

結果

Mobile Link Listener がスリープ中であることを示す *MobiLink Listener for Windows* ウィンドウが表示されます。

トラッキング情報が統合データベースにアップロードされると、Mobile Link サーバメッセージウィンドウに新しいエントリが表示されます。この情報は、Mobile Link Listener と Mobile Link サーバ間の正常な初期通信をリレーします。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 8: リモートデータベースの設定 \[133 ページ\]](#)

次のタスク: [レッスン 10: Push 要求の発行 \[136 ページ\]](#)

関連情報

[Listener \[17 ページ\]](#)

[Windows デバイス用の Mobile Link Listener ユーティリティ \(dbsln\) \[59 ページ\]](#)

[@data dbsln オプション \[65 ページ\]](#)

1.8.2.10 レッスン 10: Push 要求の発行

Push 要求を発行するには、直接 PushRequest テーブルを移植するか、Dealer テーブルで変更を加えます。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

コンテキスト

サーバ起動同期では、直接 PushRequest テーブルを移植するか、Dealer テーブルで変更を加えることで、Push 要求を発行することができます。後者の場合、Notifier の begin_poll スクリプトは、Dealer テーブルの変更を検出して、

PushRequest テーブルを移植します。どちらの場合も、PushRequest テーブルが Notifier の request_cursor スクリプトにローを入力します。これによって、リモートデバイスでメッセージを受信する方法が決まります。

手順

1. 統合データベースに接続していない場合は、Interactive SQL で接続します。

次のコマンドを実行します。

```
dbisql -c "dsn=sis_cons"
```

2. 次の SQL 文を実行します。

```
INSERT INTO PushRequest(mluser, subject, content)
VALUES ('sis_user1', 'sync', 'not used');
COMMIT;
```

3. 同期が発生するまで数秒待ちます。

移植すると、PushRequest テーブルは Notifier の request_cursor スクリプトにローを入力します。request_cursor スクリプトは、メッセージで送信される情報と、情報を受信するリモートデバイスを決定します。

4. dbmlsync が開いたままである場合は、[シャットダウン](#)をクリックし、閉じます。
5. 次の SQL 文を実行し、サーバ起動同期を要求するように、統合データベースの **Dealer** テーブルに変更を加えます。

```
UPDATE Dealer
SET RATING = 'B' WHERE name = 'Geo';
COMMIT;
```

6. 同期が発生するまで数秒待ちます。

この場合、Notifier の begin_poll スクリプトは Dealer テーブルの変更を検出し、PushRequest テーブルを適切に移植します。この場合も、PushRequest テーブルが移植されると、Notifier の request_cursor スクリプトは、メッセージで送信される情報と、情報を受信するリモートデバイスを決定します。

7. リモートデータベースで Dealer テーブルが更新されたことを確認します。

次の SQL 文を実行します。

```
SELECT * FROM Dealer
```

Geo の評価が **B** になっている必要があります。

結果

サーバ起動同期を要求する Push 要求が PushRequest テーブルに直接挿入されます。

次のステップ

次のレッスンに進みます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 9: Mobile Link Listener の設定 \[135 ページ\]](#)

次のタスク: [レッスン 11: クリーンアップ \[138 ページ\]](#)

関連情報

[Push 要求の使用方法 \[12 ページ\]](#)

1.8.2.11 レッスン 11: クリーンアップ

チュートリアルをコンピュータから削除します。

前提条件

このチュートリアルのこれまでのレッスンを完了している必要があります。

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

手順

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

コマンドプロンプトで次のコマンドを入力します。

```
odbcad32
```

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

結果

チュートリアルがコンピュータから削除されます。

タスクの概要: [チュートリアル: ゲートウェイを使用したサーバ起動同期の設定 \[120 ページ\]](#)

前のタスク: [レッスン 10: Push 要求の発行 \[136 ページ\]](#)

1.9 このマニュアルの印刷、再生、および再配布

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

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

ここに記載された情報は事前の通知なしに変更されることがあります。

重要免責事項および法的情報

コードサンプル

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

アクセシビリティ

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

ジェンダーニュートラルな表現

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

インターネットハイパーリンク

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

[go.sap.com/registration/
contact.html](http://go.sap.com/registration/contact.html)

© 2016 SAP SE or an SAP affiliate company. All rights reserved.

本書のいかなる部分も、SAP SE 又は SAP の関連会社の明示的な許可なくして、いかなる形式でも、いかなる目的にも複製又は伝送することはできません。本書に記載された情報は、予告なしに変更されることがあります。SAP SE 及びその頒布業者によって販売される一部のソフトウェア製品には、他のソフトウェアベンダーの専有ソフトウェアコンポーネントが含まれています。製品仕様は、国ごとに変わる場合があります。

これらの文書は、いかなる種類の表明又は保証もなしで、情報提供のみを目的として、SAP SE 又はその関連会社によって提供され、SAP 又はその関連会社は、これら文書に関する誤記脱落等の過失に対する責任を負うものではありません。SAP 又はその関連会社の製品及びサービスに対する唯一の保証は、当該製品及びサービスに伴う明示的な保証がある場合に、これに規定されたものに限られます。本書のいかなる記述も、追加の保証となるものではありません。

本書に記載される SAP 及びその他の SAP の製品やサービス、並びにそれらの個々のロゴは、ドイツ及びその他の国における SAP SE (又は SAP の関連会社) の商標若しくは登録商標です。本書に記載されたその他のすべての製品およびサービス名は、それぞれの企業の商標です。

商標に関する詳細の情報や通知については、<http://www.sap.com/corporate-en/legal/copyright/index.epx> をご覧ください。