Mirroring with 4D 2004.3

By Kent Wilbur, Manager Information Systems, 4D, Inc.

TN 05-38

Introduction

バージョン **2004** でバックアップモジュールが統合された当初は、**4D Backup** のミラーリングが実装されていませんでしたが、バージョン **2004.3*** でミラーリングコマンドが追加されたことにより、**4D Server** アプリケーションでミラーリングを実行できるようになりました。この **Tech Note** では、新しいコマンドとミラーリングのセットアップ方法についての説明をしています。

この **Tech Note** の目的は、ミラーリングコマンドの使用例とミラーリングの設置例を示すことであり、完璧なソリューションを提供することではありません。特定の目的に最適なアプリケーションを作成するには、必要に応じてコードを変更する必要があります。

ミラーリングの手法に関連して他の話題、例えば設定の **XML** 書き出し方法や、**4D Client** からの設定ファイル更新などについても簡単に取り上げています。

What is mirroring?

運用されているデータベースとまったく同じデータベースを平行して動作させることをミラーリングといい、メインサーバが何らかの理由で動作しなくなった場合にもっとも早く状況を回復するための手段として知られています。ミラーリングサーバは、定期的にメインサーバと同期をとっており、不測の事態が生じた際には、最後に同期をとられた状態まで更新されているので、メインサーバのログファイルが利用できる状態であれば、それを利用してログを統合すればすぐにメインサーバの後を受けて動作することができます。

バックアップからの復元とは異なり、ミラーリングサーバを使用する場合は、数分後にバックアップサーバが利用できる状態になるため、短時間でデータベースを回復できるのがミラーリングの特徴です。

---

* コマンドが追加されたのはバージョン **2004.2** ですが、**2004.3** でマイナーバグフィックスが施されています。

Does the mirrored database have requirements?

バージョン **2003** 以前では、ミラーリングサーバに条件があり、マシンが正確に対応している必要がありました。つまり、ハードドライブのパーテーション、**OS** などが一致していなければなりませんでした。**4D 2004.3** ではそのような制限がなくなり、事実上クロスプラットフォームでミラーリングをすることさえできるようになりました。唯一の条件は、ミラーサーバが勝手にデータを変えてはいけないという点です。自動的にデータを更新するようなコードを **On Startup** に記述したり、間違ってログインした人がデータを変更するようなことがないように注意して下さい。

Log files

バージョン **2004** のミラーリングも、ログファイルを利用するという点は、従来の方法と同じです。ミラーを設置するには、バックアップを実行して新しいログファイルを作成します。データベースを終了し、すべてのファイルをミラーマシンにコピーしてから両方のサーバを起動すれば、一方をミラーサーバとして使用することができます。

メインサーバでは、定期的にログファイルが新しく作成されるので、使用を終えたログファイルをミラーに送信し、ログの統合を実行すれば、ミラーを更新することができます。ミラーサーバを最新の状態に保つには、この過程が連続して実行されるようにします。

Log file numbering scheme

バックアップの結果として生成されるバックアップファイルおよびログバックアップファイルには、固有の番号が付いています。例えば、「**Mydatabase[0739].4BK**」というバックアップファイル名と「**MyDatabase[0738].4BL**」というログバックアップファイル名が使用されます。

ミラーリングを実行する場合、セグメント化されたログファイルが使用されます。通常のログファイルと区別するため、これらのログファイルの名前には「**MyDatabase[0739-0001].4BL**」、「**MyDatabase[0739-0002].4BL**」というようにセグメント番号が含まれます。統合のためにミラーサーバに送られるのは、これらのクローズドセグメントログファイルです。

New log file function

New log file 関数は、カレントログファイル**(.4DL)**を閉じて、ログバックアップファイル**(.4BL)**に名前を変更し、新しいカレントログファイルを作成します。通常の **BACKUP** コマンドを実行した場合にも同じ動作が起きますが、**New log file** 関数では実際のバックアップが行なわれない

点が異なっています。新しいログファイルが作成されると、閉じられて名前を変更されたファイルのフルパス名が返されます。そのファイル名には、最後に実行されたバックアップの番号とセグメント番号が含まれることになります。

New log file 関数は、**4D Server** でのみ実行することができます。**4D Server** 以外で実行した場合は、エラー**1412** が返されます。カレントログファイルが存在しない場合は、エラー**1403** が返されます。

New log file が実行されている間は、**4D Server** のすべてのプロセスが停止します。

Transfer the log file segment

作成されたログファイルが閉じられると、それはミラーサーバに送られます。転送の方式については、共有ネットワークボリュームを使用する、**4D Internet Commands** で **FTP** ファイル転送メカニズムを構築して **4D Open** で転送されたファイルの統合をミラーに指示する、などの方法が考えられます。サンプルでは **4D Open** ではなく **SOAP** でログファイルの転送と統合を実行することにしました。**SOAP** を使用すれば、簡単に統合の成否をメインサーバに返信することができるからです。

INTEGRATE LOG FILE

INTEGRATE LOG FILE コマンドは、渡されたログファイルを使用してログの統合を開始します。カレントログファイルは閉じられ、渡されたログファイルのエントリーが読み取られてデータファイルに統合されます。渡されたログファイルはカレントログファイルになり、ミラーリングの場合は、データベースのシンクロナイズが完了します。

INTEGRATE LOG FILE コマンドは、**4D Server** でのみ実行することができます。**4D Server** 以外で実行した場合は、エラー**1412** が返されます。カレントログファイルが存在しない場合は、エラー**1403** が返されます。

ログバックアップファイルとバックアップファイルには、それぞれの順序を示す内部的なシーケンス番号が付けられています。シーケンス番号は、ファイル名だけではなくファイルの中にも書き込まれており、たとえファイル名を変更されたとしても、ログバックアップファイルは正しいバックアップファイルに対してのみ適用されるようになっています。ログバックアップファイルとバックアップファイルの番号が対応しない場合、エラー**1410** が発生します。

ERROR HANDLING

ある種のエラーは軽微であり、次回に予定されたミラーリングの実施を阻むものではありませんが、中には致命的なエラーもあります。いかなる場合においても、適切なエラーハンドリングを実施することは重要です。**ON ERR CALL** をまったく使用していないなら、ミラーリング中にエラーが発生するとプロセスが終了し、通知や対処もすることなく未完のままタスクが終わることになります。次回のミラーリングが正しく実行されない可能性もあります。ミラーリングコマンドを使用する場合、**ON ERR CALL** の使用は必須です。

The example database

サンプルデータベースでは、簡単なミラーリングを実践しています。実行にはバージョン **2004.2** 以降**(2004.3** 推奨**)**の **4D Server** がふたつ必要です。ミラーリング、ログファイルの転送と統合、エラーハンドリングおよび回復は専用の設定ファイルで管理されています。

Creating the example database mirror

ミラーリングを実行するためには、有効なログファイルがなければなりません。その上でデータベースを別のサーバマシンに複製します。ミラーリングサーバをセットアップする際には、次の手順を正確に踏む必要があります。

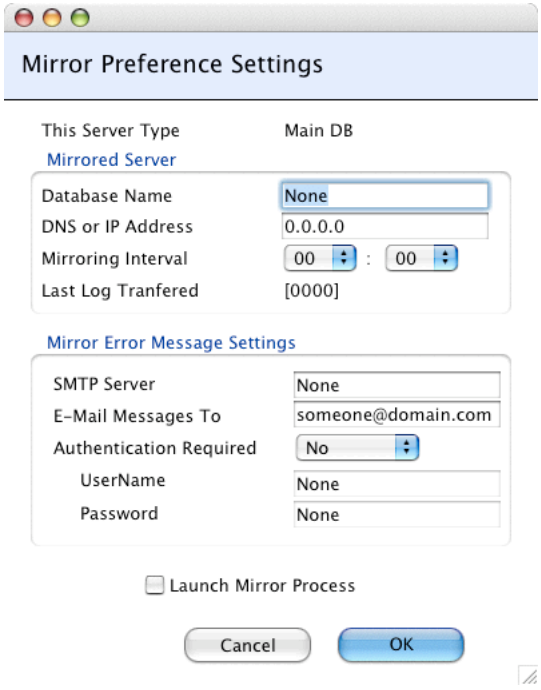
メインデータベースを起動します。**(MainDB** を選択して設定をして下さい。**)**
バックアップを実行します。**(**バックアップフォルダに **Plugins** も含めると良いでしょう。**)**
バックアップの環境設定を開いて新しいログファイルを開始します。
データベースを終了します。
ストラクチャ、データ、バックアップ、ログを含む全ファイルを別マシンにコピーします。
Preferences フォルダの中にある **Mirror** フォルダを破棄します。
ミラーデータベースを起動します。**(Mirror** を選択して下さい。**)**
メインデータベースを起動します。

何らかの理由でミラーが割れてしまった場合には、データをバックアップするところまで戻ってミラーを再構築しなくてはなりません。

The mirror preference settings

ふたつのデータベースはどこまでもそっくりなので、見分ける方法が必要です。当然のことな

がら、データそのものにそれを含めることはできません。ミラーでなくなってしまうからです。サンプルでは、外部の **XML** ファイルに設定を記述することで、ふたつのサーバを区別しています。メイン側には様々な設定項目がありますが、ミラー側にはそれがミラーであるということだけが情報として設定されます。したがって、新しいミラーを構築する場合は設定ファイルを破棄し、起動後に **Mirror** を選択すれば、それだけでミラーの設定が完了することになります。



サンプルデータベースでは、メインデータベースのデータベース名を項目として要求しています。データベース名は起動中のサーバとの接続を確立するために使用されます。**New log file** を使用するよりも前に、まずサーバに接続ができるということが肝心です。

次にミラーリングサーバの **IP** アドレスを指定します。あるいは **DNS** ルックアップで利用可能なホスト名でも構いません。

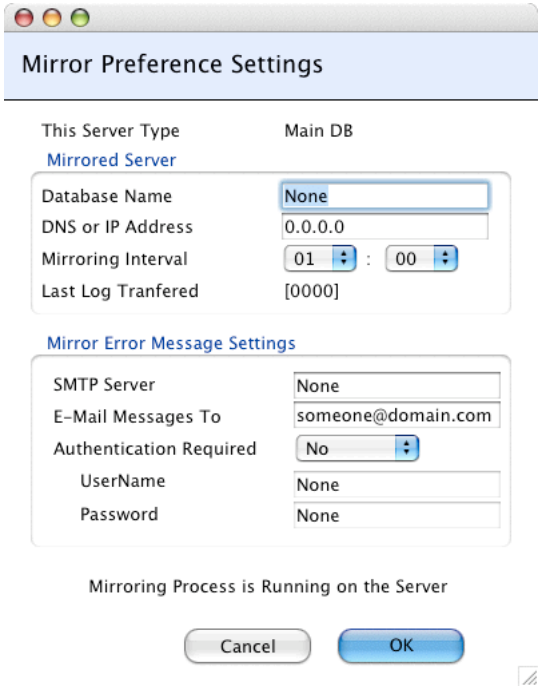
ミラーリングの間隔は、時間と分で設定します。**00hours00minutes** の場合はミラーリングが停止します。バックアップモジュールのような曜日によるスケジュールを使用したいのであれば、それは自作する必要があるでしょう。

最後の **Last Log Transfered** は、書き込み禁止の項目で、情報を提供する目的で配置されています。ここには統合に成功した最後のログが表示されており、修復の際に役立てることができ

ます。

エラーメッセージを電子メールで受け取る場合は、**SMTP** サーバと電子メールアドレスを入力します。デフォルト設定のままであれば、エラーメールは送信されません。

一番したのチェックボックスは、ミラーリングプロセスの実行状態を示しています。ミラーリングの間隔さえ設定してあれば、チェックボックスを有効にすることによって、ミラーリングプロセスを起動できます。ミラーリングプロセスが起動中であれば、更新された設定値が反映されます。



Mirror preference settings – Server side

ミラーリングの設定ファイルは、サーバに保存されます。**4D** には、バックアップその他の設定を保存するための **Preferences** フォルダが用意されているので、ミラーリングの設定 **XML** ファイルも同じ場所で管理することにしました。

ファイルの場所を変更するのであれば、メソッド **Mirror_tMirrorPath** を修正して下さい。パスは必要に応じて **Macintosh** 形式に変換されるので、ここには **Windows** で形式で記述しておきます。

Mirror_tMirrorPath

設定ファイルの読み込み、またファイルが存在しない場合の新規作成は再帰メソッド **Mirror_HandleMirrorPreferences** の中ですべて処理されています。再帰メソッドは慣れないとかえって見にくいように感じるかもしれませんが、まずはファイルが存在しないと想定してコード **Case** 文の「**load**」ステートメントへジャンプしてみて下さい。しばらく進むと再び自分をコールしているので、今度はメソッドの「**create**」ステートメントにジャンプします。

Mirror_HandleMirrorPreferences

Mirror_tMirrorPath から返されたミラーの場所は、**GEN_tFormatPathname** によってプラットフォームに適した形式に整形されます。パスは **GEN_tFolderPathnames** でフォルダのパスとファイル名に分断されますが、**Test path name** で調べた結果 **Preferences** フォルダがなければ作成します。同じく **Test path name** で設定ファイルの有無も調べ、もしなければ **Mirror_HandleMirrorPreferences(**自分**)**をコールして作成します。

GEN_tFormatPathname

GEN_tFolderPathnames

XML 文書である設定ファイルは、**DOM** コマンドで内容を読み取っています。**DOM** は、**XML** ストラクチャをすべてメモリに展開するコマンドです。一連の **DOM** コマンドを実行した結果として次のような **XML** ファイルが書き出されます。

Mirror preference settings – Client side

XML 文書であるミラーリング設定ファイルは、サーバマシンに存在するので、クライアントからこれを更新するには、ある程度のテクニックが必要です。直接ファイルを開いて内容を変更するのは大変ですし、**Execute on server** でダイアログを表示してサーバマシンで編集するというのも良い方法ではありません。

むしろ **GET PROCESS VARIABLE** や **SET PROCESS VARIABLE** によるプロセス間通信を使用したほうが便利です。どちらのコマンドともプロセス **ID** とプロセス変数名、渡す値、あるいは値を受け取るためのローカル変数を特定して使用します。プロセス **ID** が負の数であれば、それはサーバのプロセスです。値を定めたところで **Execute on server** でファイルを更新することができます。

必要な操作はすべてフォーメソッドの中で処理されています。

Form Method: Mirror_Preferences in the [zDialogs] table

The Mirroring process – Main Server side

実際のミラーリングはメインサーバで管理します。基本的にいってミラーリングプロセスは、スケジュールされた時に新しいログファイルを作成してミラーリングを実行するまでスリープしているプロセスです。**P_MirrorProcess** プロセスは、時間の計算しかしていないといっても過言ではありません。プロセスを遅らせる処理をし、途中でミラーリングの設定が変更された場合には、起きて次の実行スケジュールを再計算し、再びプロセスを遅らせる処理をしています。ミラーリングの実行タイミングは日付/タイムスタンプで管理されており、真夜中をまたがる際の計算に注意が必要です。

P_MirrorProcess

深刻なエラーが発生すると、次回のミラーリングに支障をきたす場合があります。サンプルデータベースでは、大事をとって、そのような場合にはミラーリングプロセスを停止するようにしています。

ミラーリングを実行するのは **Mirror_LSendLogFile** メソッドです。はじめにミラーリングサー

バが稼働していることを確かめ、そうでなければ新しいログファイルを作成しません。

Mirror_LSendLogFile

エラーが発生した場合は **ERROR_HandleMirrorError** メソッドでトラップされ、エラーの内容を特定するメッセージが変数に代入されます。

ミラーリングサーバが稼働していれば、新しいログファイルが作成されることになります。結果として閉じられたばかりのログファイルセグメントは **BLOB** に変換されて **SOAP** で送信されます。**(**もちろん、**SOAP** 以外の方法を使用しても構いません。**)**プロキシメソッド **proxy_SOAPHandleEvents** は、ウィザードによって自動生成されたものをほとんどそのまま使用しています。

ERROR_HandleMirrorError

The Mirroring process – Mirrored Server side

ミラーリングサーバは、基本的に。何のプロセスも実行せずに待っているだけです。ストアドプロシージャを実行する場合は、ミラーリングサーバでプロセスを実行しないような策を取るべきです。

SOAP コールが **SOAP** エントリーメソッドである **SOAP_MirrorHandleEvents** をヒットすることによってミラーリングサーバは動き出します。すべての処理が成功すると **1** が返されます。

ミラーリングはふたつの段階に分けることができます。つまり、ミラーリングデータベースが存在し、名前が合っていることを確かめること、そしてログを統合することです。

ログ統合の段階でエラーが発生した場合には、エラーがトラップされ、**SOAP** フォルトがメインサーバに返されます。

SOAP_MirrorHandleEvents

Keeping users out of the mirrored database

ミラーリングサーバは、メインサーバ同様、オンラインで実行されるものです。適切な処置を

施していないと、誤ってユーザがログインしてしまう可能性があります。

データベースの公開名を変えておけば、間違えて誰かがアクセスすることはある程度防げるかもしれません。しかしもっと確実なのは、**4D Client** のポート番号を変更してしまうことです。そうすれば、少なくとも普通の **4D Client** が誤ってアクセスすることはありません。ただし、ミラーがメインサーバの代行をするときには番号を再設定する必要があります。

Recovering from a disaster - small

ここでいう小規模とは、メインサーバが使用不能で、すべてのデータが消失し、バックアップは可能なものの、復旧には大変な時間がかかるというような場合です。

メインサーバのハードドライブにアクセスができ、いまだ統合されていないログファイルが入手できたのであれば、復旧は比較的簡単です。

1ログファイルを **4D Client** マシンのどれかにフォルダごと移動します。ログファイルを統合するまでは、ミラーリングサーバをメインサーバの代わりに利用することができません。**INTEGRATE LOG FILE** はサーバでしか実行できないので、ログファイルの統合をするためには **4D Client** を使用する必要があります。

2ログファイルをコピーされたマシンの **4D Client** でミラーリングサーバに接続します。場合によっては、サーバかクライアントのいずれかでポート番号を変更する必要があるかもしれません。

3ユーザモードで **E_RestoreDataFromMirrorLogs** メソッドを実行します。

4統合するログファイルを選択します。

5場合によっては、サーバのポート番号を変更する必要があるかもしれません。

6ミラーリングサーバを終了します。

7Preferences の **Mirror** フォルダを破棄します。

8サーバを再起動し、今度は **Main DB** として設定をします。

ログファイルが回復できなかったのであれば、そのデータの統合を諦め、ステップ **5** から始めることになります。

Recovering from a disaster - major

ワーストケースシナリオは、メインサーバとミラーリングサーバの両方でデータファイルが破損してしまった場合で、そのような状況では、バックアップから復旧するしかありません。どちらのデータベースを使用しても構いませんが、バックアップにはログファイルセグメントが何も含まれていないことになります。

1最後のバックアップからデータベースを復旧します。

2すべてのログファイルを **4D Client** マシンのどれかにフォルダごと移動します。

3ログファイルをコピーされたマシンの **4D Client** を起動します。

4ユーザモードで **E_RestoreDataFromMirrorLogs** メソッドを実行します。

5統合するログファイルを選択します。

6統合の後、バックアップを実行し、最初のときと同じようにミラーリングサーバをセットアップします。

注記**:** コードはカレントログファイルと閉じられたログファイルの区別をしないため、場合によっては、最後のログファイルを別個に統合する必要があるかもしれません。

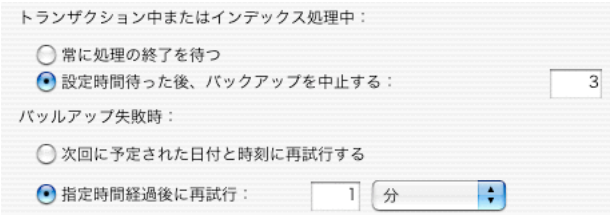
クライアント**/**サーバであっても、サーバ同士の場合と同じように **SOAP** メソッドでログを統合すると便利です。データベースの公開名と **IP** アドレスはミラーリングの設定 **XML** ファイルには含まれてはいないので、インタープロセス通信でから調べてから **SOAP** に渡します。

E_RestoreDataFromMirrorLogs

New log file function and critical operations

New log file 関数は、トランザクションやインデックスなどの動作(クリティカルオペレーション)に対して相互に排他的です。つまりクライアントあるいはストアドプロシージャがトランザクション中に新しいログファイルを作成することができず、新しいログファイルを作成中にトランザクションを始める事もできません。

データベースがトランザクションを使用している場合、ユーザが誰もいない時間帯を狙ってミラーリングを実行するのが得策であるといえます。トランザクション中に **New log file** が実行されたとしても、トランザクションが完了するまである程度の時間、ログファイルの作成を延期するようにバックアップの設定を変更することもできます。



「設定時間待った後、バックアップを中止する」の設定は、決して **0** 分にはしないで下さい。**0** 分にするとトランザクションが完了するまでバックアップはタイムアウトしません。**New log file** が失敗した場合のエラーコード **1411** であるのはクリティカルオペレーションが原因です。サンプルデータベースに含めることができませんでしたが、その場合の対応としてはスケジュールの再調整あるいはその日のミラーリングを省略、という選択肢があるでしょう。

Summary

新しいコマンドは、従来の方法に比べて柔軟性に優れており、ユーザフレンドリーなインタフェースに統合することができます。バージョン **2004.2** の時点で、これらの機能は実装されましたが、マイナーバグフィックスがあったため、バージョン **2004.3** 以降の使用が推奨されています。ちなみにサンプルデータベースは問題のバグの影響は受けません。