Mirroring with 4D 2004.4, PART I

By Kent Wilbur

TN 06-36

PartnerWare

4D は、4D アプリケーションを開発するという目的に限り、このテクニカルノートに収録されているソフトウェアの使用をパートナーに許諾します(パートナーウェア)。パートナーウェアの使用権は、4D で動作するアプリケーションに組み込まれた場合に限られており、単体での使用、あるいは 4D アプリケーション以外に組み込まれた形では認められません。ライセンスは無期限ではなく、開発者が何らかの理由でパートナーではなくなった時点で失効します。パートナーウェアを使用して開発されたアプリケーションの運用は続けられますが、開発者がパートナーではなくなった場合、パートナーウェアによる新たな開発は認められません。4D は、派生する権利を含め、パートナーウェアの全権を留保します。

Introduction

4D 2004.3 では、ミラーリングシステムを想定したコマンドがふたつ追加されました。新しく導入されたミラーリングコマンドは、かなり機能が限られており、改良を求める声が挙がっていました。

筆者が最初のテクニカルノートを公開した後、より柔軟なミラーリングソリューションを求める声が寄せられました。今回のテクニカルノートに収録されているコードは、機能が増え、エラーハンドリングも追加されており、前回のものよりもずっと効果的です。

このテクニカルノートの目的は、2004.4 で改良されたミラーリングコマンドを説明するとともに、より完成度の高いミラーリングのソリューションを提案することです。テクニカルノートに収録されているコードは、以前のコードとは互換性がありません。今回のコードを実装するのであれば、旧テクニカルノートのメソッドは取り除いて置き換えてください。

既存のデータベースにミラーリングソリューションを追加できるように、コンポーネントが提供されています。テクニカルノートおよびコンポーネントのアップデートは、必要に応じて提供される予定です。

新旧のテクニカルノートでは、コードが大幅に異なっています。以前のテクニカルノートを参照しなくても良いように、重要な内容は本テクニカルノートの中で改めて説明されています。

What is mirroring?

ミラーリングとは、実行中のデータベースと平行して実行される複製を設置するというテクニックのことです。ミラーリングの目的は、何らかの理由でメインデータベースが動作を継続できなくなった場合、もっとも短期間で復帰できる方法を備えることにあります。ミラーリングでは、定期的にデータベースの同期が合わせられるため、メインデータベースに障害が生じた場合、ミラーデータベースは、最後に同期が合わせられた時点の状態を保持しています。最後の同期以降の操作を記録したログファイルがメインサーバに残されていれば、そのログファイルに記録された比較的少数のエントリーを統合することにより、障害の生じたメインサーバの後をミラーサーバに継がせることができます。

ミラーサーバは、完全に機能するサーバであり、すでに実行中であるため、データベースの障害発生から機能再開までに要する時間は数分です。バックアップからの復元は必要ありません。ミラーリングは、メインサーバの代替サーバをセットアップできる最速の手段です。

Does the mirrored server database have any special requirements?

ミラーマシン最大のポイントは、絶対に単独でデータファイルのデータを更新してはならないという点です。たとえば、On Startup で自動的にデータを操作するようなコードを記述してはなりません。クライアント接続あるいは Web 接続により、ユーザが不意にミラーサーバにログインするようなこともあってはなりません。それぞれ On Server Open Connection、On Web Connection で対策を施してください。

もうひとつ、データベースは自分自身がミラーなのかメインなのかを判別できなければなりません。この方法はいくつか考えられます。ストラクチャファイルの名前を区別し、On Starup で確認することができます。4D Extensions フォルダに設定ファイルを置く方法もあります。サンプルデータベースでは、Preferences フォルダに置かれた XML ファイルでミラー/メインを判別しています。ミラーであれば、デーアファイルに変更を加えたり、結果的にミラー関係を壊してしまうようなコードは実行されません。

Log files

2003 以前および 2004 バージョンどちらのミラーリングにおいても、ログファイルの使用が鍵を握っている点は変わりません。ミラーを開始するには、バックアップを実行し、ログファイルの使用を開始します。その後、すぐにデータベースを終了して全ファイルをミラーマシンに移します。ミラーマシンのファイル内容は、メインマシンのファイル内容と同じでなければなりません。その後、両方のサーバを再起動してセットアップ完了です。

ミラーを持続させるためには、メインサーバで定期的に新しいログファイルを作成し、閉じられたログファイルを統合のためにミラーに送信しなければなりません。この操作はメインサーバとミラーサーバの同期を合わせるためのものであり、システム起動中は継続して実行します。

Log file numbering scheme

バックアップファイルおよびログファイルには、特有の番号が付けられています。バックアップファイルは MyDatabase[0739].4BK、ログファイルは MyDatabase[0738].4BL のようなファイル名です。

ミラーリングが関係する場合、ミラーサーバに対するログの転送用にセグメント化されたログファイルも作られます。それらのログは、他と区別するため、ファイル名にセグメント番号も付けられています。例：MyDatabase[0739-0001].4BL、MyDatabase[0739-0002].4BL。統合のためミラーサーバに転送されるのはそのようなセグメント化されたログファイルです。

New log file function

New log file 関数は、カレントログファイルを閉じ、名前を変更し、新しいカレントファイルを作成するコマンドです。バックアップ実行後にカレントログファイルがアーカイブされ、新規作成される動作に似ていますが、バックアップを実行しないでそうする点が異なっています。

New log file 関数は、閉じられた上に名前を変更されたカレントログファイルのフルパス名を返します。このファイル名には、前述のようにセグメント番号が含まれます。

New log file 関数は、4D Server でなければ実行できないコマンドです。4D Server で実行されなかった場合、エラー1412 が返され、ログファイルには何も起きません。データベースは、ログファイルを使用している必要があります。ログファイルを使用していなければ、エラー1403 が返されます。

New log file function and critical operations

New log file 関数の動作とトランザクション・インデックス作成は、同時に実行できない関係にあります。クライアントマシン、またはストアドプロシージャがトランザクション中、New log file 関数を実行することはできません。New log file 関数の実行中に新しいトランザクションを開始することもできません。

トランザクション中にバックアップが実行できないのは、2003 以前のバージョンと同じです。それでも 4D 2004.4 では、この問題に顕著な改良が加えられました。New log file 関数は新しいトランザクションの開始を阻むため、トランザクションを試みたクライアントプロセスをフ

リーズさせる恐れがありますが、4D 2004.4 では New log file 関数がトランザクションの完了を待機するタイムアウトを最短 1 秒から設定できるようになり、クライアントプロセスフリーズのリスクを最少限にとどめられるようになりました。

ミラーリングソリューションを実装する場合、デターベースのバックアップ環境設定にも注意を払う必要があります。バックアップの一般設定では、「設定時間待った後、バックアップを中止する」ラジオボタンを選択し、デフォルト待機時間の「3 分」を「1 秒」に変更してください。

設定の変更により、ミラーリングによるクライアントのロックアップが 1 秒しか続かなくなる点に注目してください。New log file 関数の成否は、エラーコードの解析で調べることができます。エラーコードが 1411 であれば、New log file 関数がトランザクションまたはインデックス作成に阻まれたことを意味します。一般的にはトランザクションが原因です。サンプルのスケジューラは、次回の実行予定日、あるいは営業日の変わり目まで操作を延期します。この処理のコードについては後半で詳述します。

トランザクションを使用するデータベースにミラーリングシステムを導入する場合、トランザクションが手早く終るように工夫をしてください。入力画面で単純にトランザクションを使用するのではなく、サブフォームを配列で置換し、リストボックスの使用を検討するべきです。これにより、トランザクション最大の問題である入力待ちのトランザクションをなくすことができます。古いバージョンから変換されたデータベースに存在する「データ入力時に自動トランザクションを使用する」オプションも外しておくようにしてください。

Transfer the log file segment

ログファイルセグメントは、作成されて閉じられた後、ミラーサーバへ転送されます。転送の方法はデベロッパ次第です。手動あるいは自動で共有ネットワークボリュームを経由しても構いません。4D Internet Commands を使用すれば、FTP ファイル転送システムを構築するこ

とができ、転送後の統合は 4D Open で実行することができます。サンプルでは、4D SOAP による転送と統合を実装しています。比べてみたところ 4D SOAP がもっともシンプルであり、統合の成否を通知する際に必要なミラーサーバからメインサーバへの通信も容易だからです。

INTEGRATE LOG FILE

INTEGRATE LOG FILE は、指定されたログファイルの統合を実行するコマンドです。具体的には、ログファイルのエントリーを解析し、その操作をデータファイルに加えます。統合が完了すれば、メインサーバとミラーサーバの同期が合っていることになります。

コマンドの使用には注意が必要です。INTEGRATE LOG FILE は、4D Server でなければ実行できません。4D Server で実行されたのでなければ、エラコード 1412 が返され、データファイルには何も起きません。New log file と同様、データベースはログファイルを使用している必要があります。ログファイルを使用していなければ、エラー1403 が返されます。

上記の点はそれほど大きな問題ではありません。むしろ、問題となるのは同期が合わなくなってしまったとき(ログファイルの送信または統合に失敗、ミラーサーバのデータファイルを間違って更新した場合など)の対応です。バックアップファイルとログファイルは、いずれも内部的な情報、ここでは便宜上、時系列管理番号と呼ぶ情報が記録されています。統合が実行されるためには、バックアップファイルとログファイルセグメントの時系列管理番号が一致していなければなりません。この番号はファイル名だけではなく、ファイル自体に記録されているので、ファイル名を変更しても変えたことにはなりません。この番号の役目は、バックアップに続く正しいログファイルだけが統合されるようにシステムを保護することです。時系列を乱すような統合を試みると、エラーコード 1410 が返されます。このエラーは、「ログファイルが違います。ファイルは問題ありませんが、最後のバックアップに続くべきログではありません」ということを意味します。

ERROR HANDLING

止まることなく動作しつづけることは、ミラーリングシステムに不可欠な要素です。もちろん、問題が起きることはあります。問題によっては、ミラーリングを停止せずに次の機会まで延期すれば済みますが、深刻なエラーの場合、そのようなわけにはいきません。いずれにしても、正しいエラーハンドリングを実装するべきです。ON ERROR CALL ルーチンが使用されていない場合、ミラーリングコマンドがエラーを返した時点でそのプロセスはアボートし、タスクが途中で終了してしまいます。エラーの原因が知らされることもありません。そのプロセスが常駐プロセスとして設計されている場合、アボートされて存在しなくなります。ミラーリングを実行するのであれば、こうした事態を割けるため、是非とも ON ERROR CALL ルーチンを作成して使用するようにしてください。

Backups

メインサーバでバックアップを実行すると、ミラー関係は壊れてしまいます。バックアップは、最初にミラーリングシステムをセットアップするとき、あるいはメインとミラーの同期が失われたときに再セットアップのために実行するものです。サンプルでは、ミラー関係が壊れるようなバックアップを実行しようとすると、ALERT が表示されるようになっています。

4D 2004.4 では、ミラーリングシステムに改良が加えられました。ミラー関係を壊すことなく、ミラーサーバのバックアップを実行することができるのです。この新しい機能により、アーカイブ目的、またはデータ保護目的で最新のバックアップをとることができ、通常のサーバと同じようにスケジューラでミラーサーバをバックアップを実行することができます。あるいは、サンプルのデータベースがそうであるように、メインサーバ側でミラーサーバのバックアップを制御することもできます。

The example database

サンプルデータベースは、単純なものから複数のミラーを有する比較的複雑なものまで、さまざまなレベルのミラーリングシステムが作成できるように設計されています。使用には、4D 2004.4 以降の 4D Server が必要です。

Creating the example database mirror

ミラーリングを実行するためには、ログファイルを使用したバックアップがなければなりません。その後、データベース全体をミラーマシンにコピーします。ミラーをセットアップするには、次の手順を順番どおりに踏むことが求められます。

1) メインデータベースを起動します。(このデータベースは XML 形式の環境設定ファイルを使用しており、最初の起動で設定を選択するようになっています。メインデータベース用には Main DB を選択してください。)
2) バックアップを実行します。(注記：バックアップには Plugins フォルダも含めることが勧められています。)
3) バックアップの環境設定を開き、新しいログファイルを作成します。
4) メインデータベースを終了します。
5) ストラクチャ、データファイル、バックアップ、ログファイルを含む、すべてのファイルをミラーマシンにコピーします。
6) ミラーマシンの Preferences フォルダの中にある Mirror フォルダを削除します。
7) ミラーデータベースを起動します。ミラーデータベース用には Mirror の設定を選択してください。
8) メインデータベースを起動します。

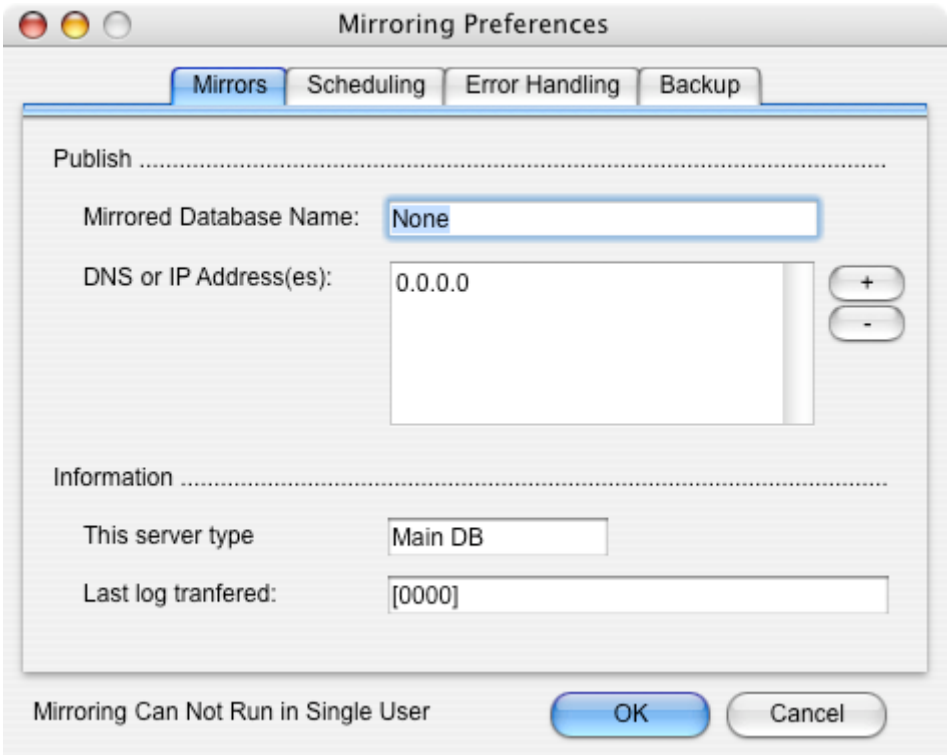
ミラー関係が壊れたときはいつでも、上記の手順を踏んでミラーリングシステムを再セットアップしなければなりません。ステップ 2、データベースのバックアップから開始してください。

The mirror preference settings

サンプルデータベースには、ミラーリングのスケジュール、トランザクション、エラーハンドリング、遠隔バックアップの設定をするために単一のインタフェースがあります。

ミラーとメインを区別することは肝要です。データは同一でなければならないため、データで区別することはできません。それではミラー関係が壊れてしまいます。サンプルでは、外部の環境設定ファイルで区別することを選びました。使用するのは簡単な XML 形式ファイルです。ミラーサーバの場合、ファイルはサーバがミラーであることを示すために使用されるだけです。ミラーサーバをセットアップするたびにファイルを削除し、起動後に MIrror 設定を選択すれば、そのサーバはミラーとして動作するようになります。

実際の環境設定は、すべてメインサーバ側で定義するようになっています。



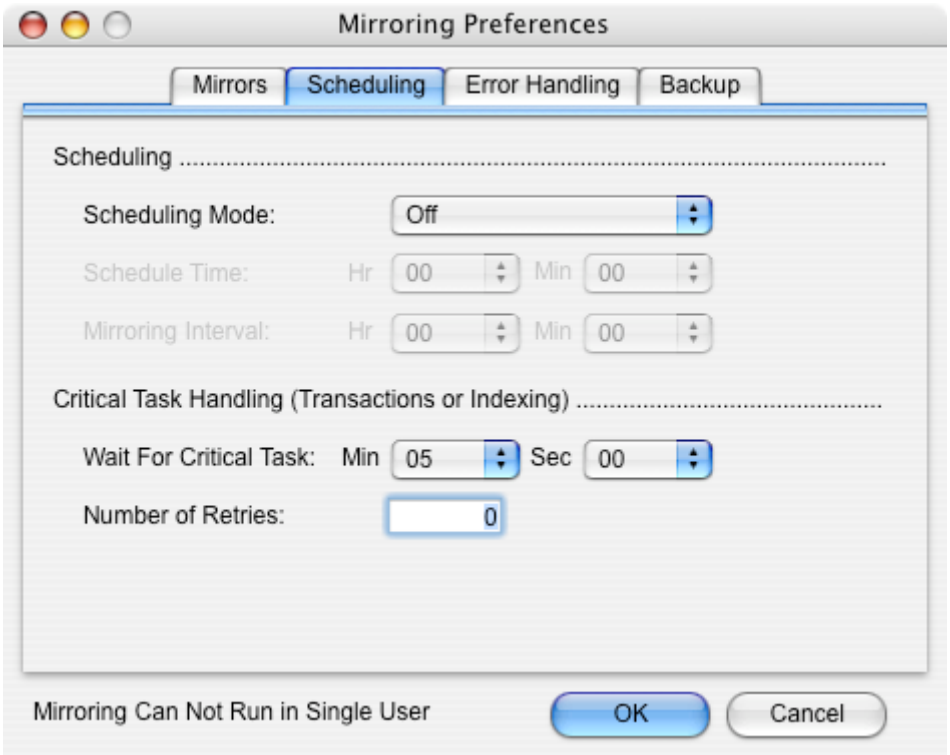
セキュリティに配慮し、サンプルでは，ミラーデータベース名の入力を求めています。この内部的な情報は、New log file を実行する前にミラーが実行中であり、ログを正しいデータベースに送信しようとしていることを確認するために使用されます。行き先のないログファイルを作成するわけにはいかないからです。

ミラーサーバの IP アドレスまたは DNS 名も入力する必要があります。サンプルのコードは、複数のミラーサーバをサポートしています。

サーバのタイプ、最後に転送されたログは表示のみの情報です。変更はできません。番号は、転送と統合に成功した最後のログファイルを示しています。情報をみることにより、ミラーの再セットアップが必要であるかが分かります。

ダイアログの下部には、サーバミラーリングプロセスのステータス表示が配置されています。ミラーリングを実行していない場合、ミラーリングの間隔が入力されていれば、チェックボックスをクリックすることにより、ミラーリングプロセスを起動することができます。プロセスがすでに起動していれば、設定に加えられた変更が反映されます。

次のタブは、スケジュールの設定画面です。選択肢は 4 個、1)Off、2)Time Only これは 24 時間に一度、特定の時間にミラーリングを実行するための設定です。3)Time&Interval これは特定の時間に実行した後、一定間隔でミラーリングを実行するための設定です。4)Interval Only これは時間と分で設定された間隔を置いて定期的にミラーリングを実行するための設定です。



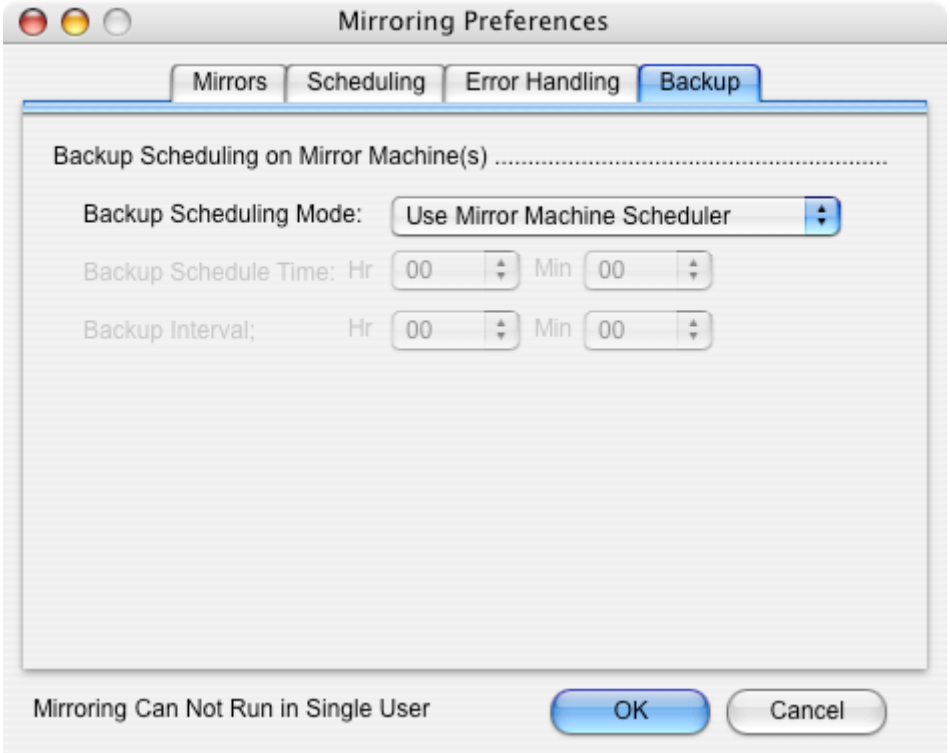
タブの下半分は、クリティカルオペレーションの対処に関する設定項目です。データベースの環境設定でタイムアウトが 1 秒に変更しておいたことを思い起こしてください。実際には、1 秒でトランザクションが終了することはありません。このタブは、1 秒間の待機をどれだけ繰り返すのかを設定するためのものです。

待機時間は 0.25 秒ごとに再測定されます。ループが実行されている間、クライアントは新しいトランザクションを開始することはできませんが、ループが実行されていることをクライアント側で調べることはでき、バックアップを終了するまでトランザクションが開始できないというメッセージを表示させることができます。最初の項目に入力する値は、バックアップ終了までクライアントを待機させる時間の設定です。この方式は、インデックス作成の完了を待つバックアップを迅速に実行させるための解決策と共存できるところがポイントです。

同じタブには、新しいログファイルの作成を断念し、次回のスケジュールまで延期する前に処理を試みる回数の設定項目があります。再試行は 1 分間隔で実行されます。この設定は、新しいログファイルの作成をするプロセスのために待機していたプロセスが先へ進めるようにし、なおかつミラーリングプロセスをアボートしなくてもよいようにするためのものです。ミラーリングに要する大体の最大時間は、Wait for Critical Task time 時間を 1.25 倍し、Number of Retries 分を加算すれば分かります。これが新しいログファイルの作成を試みるために要する時間です。この時間は、ミラーリングを実行する間隔より長くてはいけません。一日に一度のミラーリングであれば、かなりの時間を再試行に充てられますが、10 分程度の短い間隔の場合、再試行をスキップしたほうが賢明かもしれません。

次のタブは、ミラーリングエラーメッセージの設定画面です。SMTP サーバ、電子メールアドレスが入力されていれば、エラーメッセージが電子メール送信されますが、デフォルトの空白設定のままにした場合、エラー通知メールは送信されません。

このタブの残る設定項目は、メインサーバのハードディスク空き容量に関するものです。ミラーリングが継続している間、ディスクには新しいログファイルセグメントが作成されます。これらのセグメントファイルは、それぞれ異なる期間のログを記録したファイルであり、削除されるべきではありません。仮に最後のバックアップからの復元を実行しなければならない場合、小さなログファイルセグメントすべてが必要です。長時間、連続運転するサーバの場合、ログファイルセグメントが占有するディスクスペースが問題になるかもしれません。設定項目に値を入力してあり、SMTP 設定がされていれば、残り容量が一定値を下回ったときに警告のメールが送信されるようになります。単位は MB です。ミラーマシンの空き容量はチェックしていません。ミラーサーバは通常のバックアップができるため、統合されて不要になったログファイルは自動的に削除するようになっているからです。



最後のタブは、ミラーマシンのバックアップに関するものです。バックアップ方法のオプションは、ミラーリングと同じです。1) Use Mirror Machine Scheduler これは任意の選択肢です。ミラーサーバの環境設定を使用し、バックアップのスケジュールを作成します。この場合は注意が必要です。ミラーマシンのバックアップ中にメインサーバからログファイルが送られたときの対応を考えなくてはなりません。さもなければ、処理のコンフリクトが生じます。そのような理由により、サンプルでは、メインマシンを主体にしてミラーマシンのバックアップを遠隔制御するというアプローチがとられています。2) via Main Server Time これは特定の時間にバックアップを実行するという設定です。ミラーサーバのバックアップは、ログファイルを送信した後に実行できるものなので、バックアップの時間とミラーリングの時間を合わせる必要があります。そうしないと、バックアップの予定時間を過ぎ、次にログファイルが送信されるタイミングでバックアップが実行されてしまうからです。3) via Main Server Time & Interval、4) via Main Server Interval は前述のミラーリング設定と同じオプションです。

Mirror preference settings – Server side

ミラーの環境設定は、サーバマシンに保管されています。4D は、バックアップや他の設定ファイルを Preferences フォルダに保存するスタイルをすでに実装しているので、同じようにミラーの設定も XML 形式でそのフォルダに保存することにしました。

ファイルの場所は、単一のメソッドで定義しています。変更したい場合、このメソッドに含まれる文字列を変更してください。値が Windows パス形式で構いません。必要に応じ、自動的に

Windows から Macintosh へ形式が変換されるようになっています。

```
  ` パラメータを宣言
 C_TEXT($0)
 C_TEXT($1;$tFile)

 $tFile:=$1
Case of
  ¥ ($tFile="Backup")
    $0:="Preferences¥¥Backup¥¥Backup.xml"
  ¥ ($tFile="Multiples")
    $0:="Preferences¥¥Mirror¥¥"
  Else
    $0:="Preferences¥¥Mirror¥¥Settings.xml"
End case
```

旧テクニカルノートのサンプルと比較すると、上記メソッドの内容が違うことに気づくかもしれません。このメソッドを始め、多くのコードは 100%書き換えられています。

環境設定ファイルの読み込みと作成：4D Server が起動すると、ミラーリングプロセスは設定ファイルを読み込むために Mirror_HandleMirrorPreferences メソッドをコールします。環境設定ファイルのデータ操作は、すべて再帰的な形式の Mirror_HandleMirrorPreferences に集約されているため、多少、コードが読みづらいかもしれません。最初は、ミラー用の環境設定フォルダが存在しない状況を想定しながらコードを追ってゆくと良いでしょう。Case 文の条件($tAction="Load")からその部分が始まっています。

"Load"条件のコードは、やがて"create"パラメータで自身を再コールします。メソッドの初めに戻り、今度は($tAction="Create")の部分からコードを追ってみてください。

```
  ` パラメータを宣言
 C_TEXT($1;$tAction)

  ` ローカル変数を宣言
 C_BOOLEAN($fAbort)
 C_BOOLEAN($fIntervalValid)
 C_BOOLEAN($fTimeValid)
 C_DATE($dDate)
 C_DATE($dCurrentDate)
 C_STRING(16;$sXML_Reference)
 C_STRING(16;$sXML_ElementReference)
 C_LONGINT($LProcessID)
 C_TEXT($tMirror_Time;$tMirror_TimeInterval;$tMirror_BackupTime;$tMirror_BackupTimeInterval;$tMirror_T
 C_TEXT($tCurrentTime)                                                    ransactionDelay;$tMirror_TransactionRetries)
 C_TEXT($tMinimumDiskFreeSpace)
 C_TEXT($tPreferencesPath)
 C_TEXT($tSettingsFolderPath)
 C_TEXT($tXML_ElementValue)
 C_TIME($hMirroringInterval)
 C_TIME($hMirroringTime)
 C_TIME($hTime)
```

```
  ` ローカル変数に代入
$tAction:=$1

  ` デフォルト値を定義
$fAbort:=False
$tPreferencesPath:=Mirror_tFormatPathname (Mirror_tMirrorPath ("Mirror"))
```

ファイルパスを Mirror_tMirrorPath から取得した後、Mirror_tFormatPathname メソッドが実行されます。これは単純にパスの形式をプラットフォームに合わせるためのメソッドです。

```
Case of
  ¥ ($tAction="Create")
    CONFIRM("Choose the type for this server.";"Main DB";"Mirror")
    If (OK=1)
      <>Mirror_tServerType:="Main DB"
    Else
      <>Mirror_tServerType:="Mirror"
    End if
    <>Mirror_tDatabaseName:="None"
    <>Mirror_tServerIPAddress:="0.0.0.0"
    <>Mirror_tScheduleMode:="Off"
    $tMirror_Time:="00:00:00"
    $tMirror_TimeInterval:="00:00:00"
    <>Mirror_tBackupScheduleMode:="Use Mirror Machine Scheduler"
    $tMirror_BackupTime:="00:00:00"
    $tMirror_BackupTimeInterval:="00:00:00"
    $tMirror_TransactionDelay:="00:05:00"
    <>Mirror_tNextTimeIntervalMode:="00000000000000"
    <>Mirror_tNextBackupIntervalMode:="00000000000000"
    <>Mirror_LTransactionRetries:=0
    <>Mirror_tLastLogNumber:="[0000]"
    <>Mirror_tSMTPServer:=<>Mirror_tDatabaseName
    <>Mirror_tErrorEMailAccount:="someone@domain.com"
    <>Mirror_tAuthenticationRequired:="No"
    <>Mirror_tAuthenticationUserName:=<>Mirror_tDatabaseName
    <>Mirror_tAuthenticationPassword:=<>Mirror_tDatabaseName
    <>Mirror_LCurrentDiskFreeSpace:=0
    <>Mirror_LMinimumDiskFreeSpace:=0

  ¥ (Count parameters#1)
    $fAbort:=True  ` Not enough parameters
  Else
    $tMirror_BackupTime:=String(<>Mirror_hBackupTime;HH MM SS )
    $tMirror_BackupTimeInterval:=String(<>Mirror_hBackupTimeInterval;HH MM SS )
    $tMirror_Time:=String(<>Mirror_hTime;HH MM SS )
    $tMirror_TimeInterval:=String(<>Mirror_hTimeInterval;HH MM SS )
    $tMirror_TransactionDelay:=String(<>Mirror_hTransactionDelay;HH MM SS )
End case
```

続く部分は、ミラーリングの実行予定に関するものです。特筆するようなことはありません。

```
Case of
  ¥ ($fAbort)
  ¥ ($tAction="SetNextTimeInterval")
    Case of
      ¥ (<>Mirror_tScheduleMode="Time Only")
        If (Current time(*)><>Mirror_hTime)
          <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Add to date(Current date(*);0;0;1);<>Mirror_
                                                                                                    hTime)
        Else
          <>Mirror_tNextTimeIntervalMode:=Mirror_tDateTimeStamp (Current date(*);<>Mirror_hTime)
        End if
```

〈以降メソッドが続く。原文またはサンプルストラクチャを参照してください。〉