4D Backup 6.7

ランゲージリファレンス
Windows® and Mac™ OS

4D

4D Insider 6.5 ランゲージリファレンス
Windows® and Mac™ OS

このマニュアルに記載されている事項は、将来予告なしに変更されることがあり、いかなる変更に関しても4D SAは一切の責任を負いかねます。このマニュアルで説明されるソフトウェアは、本製品に同梱のLicense Agreement（使用許諾契約書）のもとでのみ使用することができます。

ソフトウェアおよびマニュアルの一部または全部を、ライセンス保持者がこの契約条件を許諾した上での個人使用目的以外に、いかなる目的であれ、電子的、機械的、またどのような形であっても、無断で複製、配布することはできません。

4th Dimension、4D Server、4D、4Dロゴ、4Dロゴ、およびその他の4D製品の名称は、4D SAの商標または登録商標です。

MicrosoftとWindowsはMicrosoft Corporation社の登録商標です。

Apple, Macintosh, Mac, Power Macintosh, Laser Writer, Image Writer, ResEdit, QuickTimeはApple Computer Inc.の登録商標または商標です。

その他、記載されている会社名，製品名は、各社の登録商標または商標です。

注意

このソフトウェアの使用に際し、本製品に同梱のLicense Agreement（使用許諾契約書）に同意する必要があります。ソフトウェアを使用する前に、License Agreementを注意深くお読みください。

目次

4D Backup はじめに

このマニュアルについて

概要

このマニュアルは、4D Backupのランゲージの紹介と、それらのコマンドの使い方を説明しています。4Dインタフェースと4D Backupの使用は、『4D Backupユーザリファレンス』で説明しています。

4D Backupルーチンは、テーマごとに別れていて、マニュアルと4th Dimensionメソッドエディタウインドウの両方で確認できます。

4D Backupランゲージを理解できるように、ルーチンの説明では簡単なバックアップの使用例を示しています。

クロスプラットフォームマニュアルの取り扱い方法

このマニュアルは、MacintoshとWindows両方の環境における使用方法を説明します。2つのプラットフォーム上で4D Backupの考え方や機能はほとんど同じですが、必要がある場合は、その違いについても説明があります。こうした違いには、表示上のユーザインタフェースやキーボードコマンドも含まれます。

表記方法について

このマニュアルでは、戻り値のない4D Backupコマンドは、例えば、**BK END BACKUP**というように、すべて大文字で表記します。戻り値のある4D Backup関数は、例えば、**BK Start copy**のように、BKの後の最初の1文字だけ大文字で表されます。

4D Backupコマンドまたは関数には、BKという頭文字を付けて、4th Dimensionのコマンドや別のモジュールによって追加されたコマンドと区別しています。

各4D Backupコマンドまたは関数がメソッドまたはオブジェクトメソッド中にあらわれる場合は、4th Dimensionのコマンドまたは関数に組み込まれたものと区別するために、ボールドの斜体で表示されます。斜体ではないボールドの場合は、4th Dimensionのランゲージ用語を示します。

4D Backupルーチンの使用

4D Backupルーチンは、バックグラウンドで、別々に作業します。ほとんどの4D Backupルーチンはバックアッププロセスで実行されるので、ルーチンを使用するためにプロセスを明確にオープンする必要があります。作業が終了したら、プロセスをクローズしてください。

バックアッププロセスの開始

バックアッププロセスを起動するためには、フルバックアップを遂行するか、ミラーの更新を遂行するかによって、下記のどちらかのルーチンを使用できます。

■ **BK Begin full backup:**フルバックアップ

■ **BKBegin mirror update**:ミラーの更新

これらの機能は、バックアップまたはミラーの更新を実行しません。これらの操作を実行するために与えられる4D Backupプロセスをオープンするだけです。

フルバックアップおよびミラーの更新は、別々の4th Dimensionプロセス内で実行します。4th Dimensionのマルチプロセスな言語になじみがなければ、このマニュアルのランゲージの節を読む前に、『4th Dimensionランゲージリファレンス』マニュアルで、概念を再確認してください。

バックアップの実行

以前記録されたルーチンのひとつにより、一旦バックアッププロセスがオープンすると、フルバックアップおよびミラーの更新といったすべてのバックアップ操作は**BK Start copy**関数によって実行することができます。この関数は、第3章「実行」で説明しています。

コピーが起動した後、コピーが終わるまで待たなければなりません。しかし、**BK Get state**関数を用いて終わらせるかどうかを決定することができます。この関数は、コピーがきちんと実行できたら4を返します。

典型的な、バックアップとミラーの更新のメソッドは、下記の記述を含みます。

```
...
$vBackup := BK Start copy
Repeat
   ...
Until (BK Get state # 4)
...
```

バックアッププロセスのクローズ

バックアッププロセスは、データベースの一定の制御を実行します。例えば、バックアッププロセスがオープンしている間、ユーザはデータを加える、修正する、あるいは消去することができません。データベースに再びデータを書き込むためには、バックアッププロセスを終了させなければなりません。

そのためには、**BK END BACKUP**コマンドを使用して、バックアッププロセスを終了します。

バックアッププロセスのアボート

バックアッププロシージャの終わりに、**BK END BACKUP**コマンドを忘れると、データベースは追加、修正、削除といったすべての書き込み操作を妨げたまま残ります。この場合、バックアップをアポートする必要があります。

以下のキーの組み合わせを数秒押し続けることで、バックアッププロセスをアボートすることができます。

■ Macintosh上では、「control + option + shift」キーを押しながらクリックします。

■ Windows上では、「Alt + Shift」キーを押しながらマウスを右クリックします。

例外

4D Backupの4つのルーチンは、それ自体がオープンおよびクロースを制御するので、バックアッププロセスのオープンまたはクローズを要求しません。

BK FULL BACKUP WINDOWおよび**BK UPDATE MIRROR WINDOW**（第2章「標準ダイアログボックス参照）のコマンドは、Windowsメニューから適切なメニュー項目を選択することで、フルバックアップおよびミラーの更新に対するパラメータウインドウを表示します。

BK Update mirror（第5章「論理ミラー」参照）および**BK FULL backup**（第3章「実行」参照）の関数は、データベースと同じフォルダの中で見つけたデフォルトプロジェクトを使用して適切なオペレーションを速やかに実行します。

標準ダイアログボックス：はじめに

この節では、標準的な4D Backupウインドウである「フルバックアップ」ウインドウと「ミラーの更新」ウインドウを表示するコマンドについて説明しています。

これらのコマンドは「ユーザ」モードの「ウインドウ」メニューにある「フルバックアップ」メニューと「ミラー更新」メニューに相当します。

上記のコマンドを使用すると、下記のようなことが可能になります。

■「ランタイム」モードでのカスタムメニューアイテムからのバックアップの実行

■ レイアウト上のボタンスクリプトからのバックアップの実行

■ 任意の時間における、あるプロセスからのバックアップの実行

■ データベース終了時のバックアップの実行

これらのルーチンは、各自のバックアッププロセスのオープンやクローズを行います。バックアッププロセスを開始するために**BK Begin full backup**関数や**BK Begin mirror update**関数を呼び出したり、バックアッププロセスを終了するために**BK END BACKUP**コマンドを呼び出す必要はありません。

BK UPDATE MIRROR WINDOW

BK UPDATE MIRROR WINDOW

引数　　タイプ　　説明

このコマンドには、引数がありません。

説明

BK UPDATE MIRROR WINDOWコマンドは、「ミラーの更新」ウインドウを表示します。このコマンドは、「ユーザ」モードの「ウィンドウ」メニューから「ミラー更新」を選択するのと同じです。

例えば、「ランタイム」モードで動作するデータベースの場合に、このコマンドを使うことができます。ユーザは、このコマンドを呼び出すカスタムメニューアイテムを選択することによって、「ミラーの更新」ウインドウにアクセスします。詳細については、BK **FULL BACKUP WINDOW**コマンドの例を参照してください。

このコマンドは、本来はコンパイルされたデータベースの管理者がデフォルトのバックアップパラメータを変更するために「ミラーの更新」ウインドウにアクセスできるようにするために使用されるものです。コンパイルされたデータベースは、「デザイン」モードにアクセスすることができません。

定期的にミラー更新を行うなら、代わりに**BK Update mirror**関数を使用することをお勧めします。しかし**BK Update mirror**関数がエラーを返した場合は、**BK UPDATE MIRROR WINDOW**コマンドを呼び出し、ユーザがパラメータを変更した後、再度実行してください。

例題

次のプロシージャは、**BK Update mirror**関数がエラーを返した場合に「ミラーの更新」ウインドウを表示します。

```
    If (BK Update mirror  # 0)
       ALERT ("ミラー更新が失敗しました。")
⇒      BK UPDATE MIRROR WINDOW
    End if
```

参照

なし

BK FULL BACKUP WINDOW

BK FULL BACKUP WINDOW

引数　　タイプ　　説明

このコマンドには、引数がありません。

説明

BK FULL BACKUP WINDOW コマンドは、「フルバックアップ」ウインドウを表示します。このコマンドは、4th Dimension や 4D Client の「ユーザ」モードや 4D Server の「プラグイン」メニューから「フルバックアップ」を選択するのと同じです。

例題

「カスタムメニュー」モードで動作するカスタムデータベースがある場合、このコマンドを使うことによって、ユーザはメニューアイテムからデータベースのフルバックアップを実行することができます。

次に「ランタイム」モードのメニューバーに表示されるメニューアイテムにこのプロシージャを組み込みます。

これでユーザは「ランタイム」モードでデータベースのバックアップを作成することができます。これは、コンパイルされたデータベースやランタイムアプリケーションになっているデータベースでも当てはまります。

参照

なし

実行：はじめに

この節では、バックアップの実行に関するコマンドについて説明します。これらのコマンドで、ハードディスク上のデータのバックアップを起動、およびバックアップの進行状況を把握することができます。また、ミラーデータベースを更新するためにログファイルをデータベースに送付することもできます。

また、この節では次のプラグインエリアについても述べています。

■ **%BACKUP PROGRESS**

■ **%FILLING PROGRESS**

ログファイルについてもこの章に含まれるため、この章では複製（コピー）の実行を行うルーチンについても説明します。パラメータの設定および論理ミラーの管理に関するルーチンについては、第5章「論理ミラー」で説明されています。

BK Begin full backup

BK Begin full backup→整数

引数	タイプ	説明

このコマンドには、引数がありません。

説明

第1章の「バックアッププロセスの開始」で述べた4つのルーチンを除いて、すべての4D Backupのコマンドと関数は**BK Begin full backup**関数（あるいは、**BK Begin mirror update**関数）と**BK END BACKUP**コマンドの間に納まらなければなりません。

BK Begin full backup関数は、バックアッププロセスを開始します。この関数はデータベースのバックアップに必要な初期処理を行います。この関数はすべてのトランザクション処理が終わるまで待機し、どのデータも追加修正ができないようにデータベースをロックします。

BK Begin full backup関数は、実行が成功すると0を返します。そうでない場合はエラーコードが返されます（詳細は、「付録B」のエラーリストを参照）。リターン値に対して、きちっとした検査を行うことをお勧めします。もし、BK Begin full backup関数が0以外の値を返した場合、その後に続く命令は無視されます。

BK Begin full backup関数を呼び出した後、バックアップ処理を続けるためにはBK Get state関数を定期的に呼び出す必要があります。これに関する詳細は、後述のBK Get state関数を参照してください。

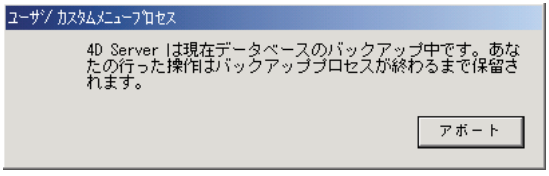
バックアップ中のデータベースへのアクセス

BK Begin full backup関数を呼び出すと、データベースへのアクセスは次のように制限されます。

■ **4th Dimension：**バックアッププロセス以外のすべてのプロセスは中止されます。その他のプロセスは、BK END BACKUPコマンドが呼び出されるまで再開されません。

バックアッププロセスを解除するには、Macintosh上では「control+option+shift」キーを押しながらマウスクリック、Windows上では「Alt+Shift」キーを押しながら右マウスボタンクリックします。

■ **4D Server：**4D Serverに接続したクライアントはレコードの検索を行うことはできですが、レコードの追加、更新、削除はできません。クライアントが追加、更新、あるいは削除の要求を送信すると、下図のダイアログボックスが現われ、クライアントはバックアップが終わるのを待たなければなりません。



バックアップが終了すると、このウインドウは消え、要求された処理は実行されます。

「アボート」ボタンを選択すると、ユーザは要求を取り消すことができます。

注：バックアップの実行前にプロシージャからの要求が実行された場合、処理の取り消しは行うべきではありません。その処理を取り消すと、4th Dimensionがプロシージャの残りの処理を取り消してしまい、部分的に実行されたプロシージャがデータベース内で論理的に矛盾を起こします。この状態で「アボート」ボタンが押されると、4th Dimensionはエラーコード-9976を返します。

トランザクションの扱い

BK Begin full backup関数は、すべてのトランザクションが終了するのを待ちます。同じプロセス内で**START TRANSACTION**コマンドを使ってトランザクションを開始した後、**VALIDATE TRANSACTION**コマンドや**CANCEL TRANSACTION**コマンドを実行せずに**BK Begin full backup**関数を使用したバックアッププロセスを開始してしまうと、**BK Begin full backup**関数はエラーコード1404を返します。

トランザクション処理中に「入力」ダイアログボックスを表示しないようにしてください。ユーザがダイアログボックスを開いたままにしておくと、データベースのバックアップを実行することができなくなります。その上、トランザクションが終わったのを確認するまで、すべてのユーザに対してデータベースは書き込みできないようにロックがかかります。このようなことから、トランザクションはダイアログボックスを表示した「後」にのみ起動するべきす。ユーザがダイアログボックスを有効にした後、トランザクションを開始し、その後でトランザクションを受け付けたり、取り消したりすることができます。

追加、更新、削除

バックアップに関する情報を保管するためにレコードを更新したい場合、**BK END BACKUP**コマンドを呼び出した後でその処理を行います。**BK Begin full backup**関数から**BK END BACKUP**コマンドまでが実行されている間は、レコードの追加、更新、削除はできません。

例題

1. 次の例は、フルバックアップを実行する最も簡単なプロシージャです。

```
C_INTEGER ($vError)
$vError:=BK Full backup
```

2. 次のプロシージャもバックアップを実行します。

```
⇒   If (BK Begin full backup=0)
        `バックアッププロセスをオープンする
      If (BK Start copy=0)
          `バックアップの実行
        Repeat
        Until (BK Get state #4)
          `コピーの最後
      End if
      BK END BACKUP
        `バックアッププロセスのクローズ
End if
```

3. 発生する可能性があるエラーを処理することによって、プロシージャを改善することができます。

```
    C_INTEGER ($vError ; $vState)
⇒   $vError:=BK Begin full backup
    If ($vError#0)
      ALERT ("バックアップを開始できません。エラー番号" +String($vError))
    Else
      $vError:=BK Start copy
      If ($vError#0)
        ALERT ("バックアップを開始できません。エラー番号" +String($vError))
      Else
        Repeat
          $vState:=BK Get state
        Until ($vState#4)
        If ($vState#5)
          ALERT ("コピー中に障害が起きました。エラー番号" +String($vState))
        End if
      End if
      BK END BACKUP
    End if
```

参照

BK END BACKUP、BK Begin mirror update、BK Get state

BK END BACKUP

BK END BACKUP

引数	タイプ	説明

このコマンドには、引数がありません。

説明

このコマンドは、**BK Begin full backup**関数や**BK Begin mirror update**関数で開始されたバックアッププロセスをクローズします。**BK Begin full backup**関数や**BK Begin mirror update**関数をそれぞれ呼び出すと、バックアップ終了時にBK END BACKUPコマンドが呼び出されるまで処理を続行します。

フルバックアップ中、4th Dimensionはデータベース上のデータへの変更を許可しません。**BK END BACKUP**コマンドを実行することによって、データベース上のレコードへの追加、修正、削除が自由に行えるようになります。

注：BK Begin full backup関数を呼び出す前にロックされたレコードは、データベースが開放された後もロックされたままになります。

このコマンドを呼び出すタイミングによって、次の3つの中の1つが発生します。

- データベースやログファイルが正しくコピーされた時。**BK Get state**関数は、リターン値5を返します。
- データベースやログファイルをコピーしている最中の時。この場合、**BK END BACKUP**コマンドの呼び出しは、「フルバックアップ」ウインドウから実行されるバックアップを中止するための「中止」ボタンをクリックするのに相当します。この場合、処理は取り消され、宛先ファイルは消去されます。ミラー更新に関しては、データベースを前の状態に戻すために送信するログファイルと新しいログファイルが結び付けられます。
- データベースあるいはログファイルがまだコピーされていない時。フルバックアップでは、この状況は特に問題はありません。ミラー更新では、いくつかの処理がまだ実行されています（カレントログファイルはクローズされ、新しいログファイルが作られます）。この場合、送信されるログファイルと新しいログファイルはデータベースを前の状態に戻すために結び付けれられます。

BK Get state関数がリターン値5を返すまで（すなわち、コピーが終了するまで）、**BK END BACKUP**コマンドを呼び出さないようにしてください。

注：4th Dimensionを使用中に、BK END BACKUPコマンドをバックアッププロシージャで省略すると、データーベースはロックしたままになります。この場合、Macintosh上では「control+option+shift」キーを押しながらマウスクリック、Windows上では「Alt+Shift」キーを押しながら右マウスボタンクリックして、バックアッププロセスを解除してください。

参照

BK Begin full backup、BK Get state、BK Begin mirror update

BK Start copy

BK Start copy → 整数

説明

フルバックアップを実行していると、**BK Start copy**関数はバックアップ先のディスクにデータをコピーします。

ミラー更新を実行していると、**BK Start copy**関数はログファイルを送信します。

処理が成功するしないに関わらず、リターン値がコードで返されます。コピーが成功すれば、**BK Start copy**関数は0を返します。そうでない場合は、エラーコードが返されます（エラーリストは、「付録B」を参照してください）。

バックアップは、独立したプロセスで行われます。コピーの進行状況をユーザに知らせるといった他の処理のプロセスを使うことができます。

BK END BACKUPコマンドを呼び出す前に、コピーが終わるまで待たなければなりません。コピーが終わる前にこのコマンドが呼び出されると、最後のコピーは取り消されます。

バックアップまたはミラー更新中は、処理を続けるために定期的に**BK Get state**関数を呼び出すようにしてください。これに関する詳細は、後述の**BK Get state**関数を参照してください。

参照

BK Begin full backup、BK Begin mirror update、BK Get state

BK Full backup

BK Full backup→整数

引数	タイプ		説明
この関数には、引数がありません。			
戻り値	整数	←	操作が正しく実行された場合は0、そうでない場合はエラーコード

説明

すでにフルバックアップを実行し、プロジェクトにバックアップパラメータを保存した場合、この関数は、副バックアップを順次実行することができます。

BK Full backup関数は、他のルーチンから独立しています。すなわち、**BK Begin full backup**と**BK END BACKUP**ルーチンの間に入れる必要はありません。

BK Full backup関数は、バックアッププロジェクトのすべてのパラメータ有効にし、選択されたボリュームが生きている限り、データベースのバックアップを実行します。

関数の戻り値の整数は、バックアップ実行中に起こった問題かどうかを決定します。コードは下記に示します。

コード	説明
0	実行は成功しました。
1301	バックアップはすでに始まっています（バックアッププロセスはすでに作成されています）。
1302	バックアップは始まっていません（バックアッププロセスは作成されていません。BK Begin full backupまたはBK Begin mirror updateをコールしてください）。
1303	バックアップ先が違います（バックアップ先のボリュームが見つからないか、修正されたパスにアクセスできません）。
1305	4D Backupは4D Serverにインストールされていません。

エラーコードの完全なリストは、付録A「4D Backupエラーコード」を参照してください。

参照

なし

BK GET PROGRESS

BK GET PROGRESS (進行率;ディスク使用率)

引数	タイプ	説明
進行率	整数型の変数	バックアップの進行率（パーセントで表示）
ディスク使用率	整数型の変数	ディスク使用率

説明

このコマンドは、バックアップの進行とディスク記憶装置（すなわち、ディスクの使用量）に相当する率を変数に返します。このコマンドの機能は、先に述べたプラグインエリアの機能と似ています。

このコマンドによって返される値は、0から100までの整数です。

引数＜進行率＞は、0から始まり100で終わりです。

引数＜ディスク使用率＞は、ディスクの使用率と同じです。リターン値が0の場合、ディスクは空の状態です。100の場合はディスクをすべて使用している状態です。75は、75%ディスクを使っている状態であることを示しています。

このコマンドは、バックアップ中にコピーの進行状況を把握するために使用します。例えば、4th DimensionのMESSAGEコマンドの助けを借りて使用します。

例題

下記はバックアップ進行状況を把握するための例です。

```
C_INTEGER ($Progress ; $Fill)
If (BK Begin full backup = 0)
   If (BK Start copy = 0)
      Repeat
⇒        BK GET PROGRESS( $Progress ; $Fill)
         MESSAGE ("バックアップ処理中です；" +String ($Progress) +"%")
      Until (BK Get state # 4)
   End if
   BK END BACKUP
End if
```

参照

なし

BK Get state

BK Get state→整数

引数	タイプ	説明
この関数には、引数はありません。		
戻り値	整数	バックアッププロセスの状態

説明

この関数は、バックアッププロセスの状態を返します。

この関数は、次の理由により定期的に呼び出す必要があります。

- この関数は、バックアッププロセスの状態を知らせます。これは、**BK END BACKUP**コマンドを使用してバックアッププロセスをクローズできるため、例えばバックアップが終了しているかどうかを調べることができます。
- この関数は、バックアッププロセスに処理時間を割り当てます。定期的に、**BK Get state**関数を呼び出さない場合、バックアップコピーは進行しません。

BK Get state関数は、次の環境下で呼び出すことができます。

- コピーの前
- コピー中
- コピーの後

この関数の説明の最後に、**BK Get state**関数を呼び出す方法の1つの例が紹介されています。

BK Get state関数と**BK Get error number**関数を混同しないようにしてください。**BK Get state**関数は、バックアップ状況の情報を返します。**BK Get error number**関数は、問題がある場合にエラー番号を返します。

BK Get state関数は、次のようなリターン値を返します。

リターン値	説明
1	ディスクが選択されていません。この値はバックアッププロセスが開始された後に返されます。この場合、**BK SET VOLUME**コマンドを実行することによってディスクを選択する必要があります。ボリュームを選択せずにバックアップを実行した場合はエラーになり、バックアップを続行することができません。
2	選択したディスクは使用できません。プロジェクトか**BK SET VOLUME**コマンドで選択されたディスクはバックアップには使用できません。それは、ディスクにバックアップに十分な空きがないか、あるいはプロジェクトが存在していないためです。取り出し可能なボリュームであれば、そのディスクはなくなっているかもしれません。
3	コピーの準備ができています。この値はバックアップを開始する準備がすべて整っていることを示しています。（**BK Start copy**関数を使用して）コピーを開始する前にBK Get state関数を呼び出すことをお勧めします。
4	コピー処理中です。**BK GET PROGRESS**コマンドを使用するか、「%BACKUP PROGRESS」と「%FILLING PROGRESS」の2つのプラグインエリアを使用して進行状況を確認することができます。
5	コピーが正常に終了しました。ここで、バックアッププロセスをクローズするために**BK END BACKUP**コマンドを呼び出すことができます。
6	バックアッププロセスが開始されていません。**BK Begin full backup**関数または**BK Begin mirror update**関数が呼び出されていないか、関数が失敗しています。関数が失敗した場合、関数から返されたコードが失敗した原因を示します。
7	バックアップが失敗しました。BK Start copy関数は実行されましたが、問題が起きたためバックアップが中断されました。

例題

下記は標準的なバックアップの実行例を示しています。このプロシージャは発生し得るすべての状態について処理します。

```
C_INTEGER ($State ; $Progress ; $Fill)
If (BK Begin full backup # 0)
      `バックアップの開始が失敗した場合
   ALERT ("バックアップを開始できません。")
```

```
            `アラートを表示
        Else
            `そうでない場合
            MESSAGE ("バックアップ中...")
            `メッセージを表示
            BK OPEN PROJECT ("週間バックアップ")
            `バックアップ用プロジェクトのオープン
⇒           $State:=BK Get state
            `バックアップ実行前の状態の読み出し
            Case of
            \($State=1)
                `バックアップ状態=1の場合
                ALERT ("ディスクが選択されていません。")
                `アラートを表示
            \($State=2)
                `バックアップ状態＝2の場合
                ALERT ("選択したディスクが使用できません。")
                `アラートを表示
            \($State=3)
                `バックアップ状態＝3の場合
                If (BK Start copy # 0)
                    `コピーが正常に開始されるかされないかのテスト
                    ALERT ("バックアップが開始できませんでした。")
                    `失敗の場合、アラートを表示
                Else
                    `そうでない場合
                    Repeat
                        `繰り返し読み出しと表示を実行
                        BK GET PROGRESS ("$Progress ; $Fill)
                        MESSAGE ("バックアップ中：" + String ($Progress)+"%")
⇒                   Until(BK Get state # 4)
                    `バックアップの処理が終了するまで
⇒                   If (BK Get state=5)
                        `バックアップが正常に終了した場合
                        MESSAGE ("バックアップが正常に実行されました。")
                    Else
                        `そうでない場合
                        ALERT ("バックアップ中に障害が発生しました。")
                    End if
                End if
            End case
        BK END BACKUP
        End if
```

参照

なし

BK Start copy

BK Start copy→整数

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	整数	←	コピーが成功した場合は0、そうでない場合はエラーコード

説明

フルバックアップを実行していると、**BK Start copy**関数はバックアップ先のディスクにデータをコピーします。

ミラー更新を実行していると、**BK Start copy**関数はログファイルを送信します。

処理が成功するしないに関わらず、リターン値がコードで返されます。コピーが成功すれば、**BK Start copy**関数は0を返します。そうでない場合は、エラーコードが返されます（エラーリストは、「付録B」を参照してください）。

バックアップは、独立したプロセスで行われます。コピーの進行状況をユーザに知らせるといった他の処理のプロセスを使うことができます。

BK END BACKUPコマンドを呼び出す前に、コピーが終わるまで待たなければなりません。コピーが終わる前にこのコマンドが呼び出されると、最後のコピーは取り消されます。

バックアップまたはミラー更新中は、処理を続けるために定期的に**BK Get state**関数を呼び出すようにしてください。これに関する詳細は、後述の**BK Get state**関数を参照してください。

参照

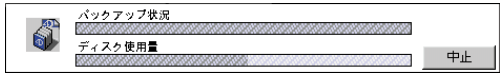
BK Begin full backup、BK Begin mirror update、BK Get state

プラグインエリア

%BACKUP PROGRESS
%FILLING PROGRESS

説明

4th Dimension内に独自のバックアップウインドウを作成した場合、これらのプラグインエリアによって、コピーの進捗状況やバックアップ先ディスクの使用割合を表示する独自のサーモメーターを表示することができます。例えば、4D Backupメインウインドウに表示される進歩サーモメーターをシュミレートすることができます。



これらのプラグインエリアをインストールするには、入力レイアウトにアクティブオブジェクトを作成します。このフォームをバックアップが起動された時に表示させることができます。アクティブエリアに変数名を入力し、エリアタイプのポップアップメニューから「プラグインエリア」を選択します。

次に、先のポップアップメニューの下にあるポップアップメニューから表示したい処理状態のタイプによって「%BACKUP PROGRESS」か「%FILLING PROGRESS」を選択します。

注：「_フルバックアップ」「_ミラー更新」「_ログの復元」が4th Dimensionのプラグインエリアのポップアップメニューに表示されますが、これらの項目は選択しないでください。これらを選択すると、4D Backupはレイアウト上にあるプラグインエリア内で機能できなくなります。

プラグインエリアがレイアウト内にインストールされると、標準のバックアップウインドウでサーモメータのような動作を行います。

「%FILLING PROGRESS」は、バックアップ先ディスクの使用率を示します。
「%BACKUP PROGRESS」はバックアップ前は空ですが、処理が終了するまでコピーの進行率を表示します。

注：サーモメータが機能するには、バックアップを実行したのと同じプロセスに置かなければなりません（他のプロセスでは機能しません）。

参照

BK GET PROGRESS

インフォメーション：はじめに

この節では、バックアップに関する情報を得るルーチンについて説明します。各バックアップ後、データファイルはバックアップに関する情報を保存します。

データベースで実行された最新のバックアップに関する情報：番号、日付、バックアップを行なった時間を取得することができます。また、カレントバックアップに含まれるファイルのサイズを調べるコマンドも使用することができます。

BK Get current set

BK Get current set→倍長整数

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	倍長整数	←	カレントバックアップセットの数

説明

BK Get current set関数は、カレントバックアップのセット番号を返します。各バックアップごとにカウンターが加算されます。

バックアップの番号は、バックアップファイル名の終わりに記述されます：

- Macintosh上では、バックアップ番号はカッコ（[ ]）の間に記述されます。例えば、“ MyBase[25] ”という書類は“ MyBase ”データベースの25番目のバックアップであるという意味です。
- Windows上では、バックアップ番号はファイル名の後に記述されます。ファイル名は8バイトまでしか許可されていないので、はみ出た文字は切り取る必要があり、母音は削除されます。例えば、“ Mbs025.4BK ”という書類は“ MyBase ”データベースの25番目のバックアップであるという意味です。

セット番号は読むことしかできず、変更することはできません。この番号はデータベースのバックアップと共に自動的に加算され、データベースのデータファイルに格納されます。

セット番号とは実行されるバックアップのセットを表します。例えば、一度もバックアップされていないデータファイルのセット番号は1です。最初のバックアップの後、セット番号は2といったように番号が振られます。

BK Get last backup date

BK Get last backup date→日付

説明

この関数は、フルバックアップかミラー更新かどうかに関わらず、最後にバックアップが実行された日付を返します。バックアップが一度も行なわれていない場合、4D Backup は「!00.00.00!」という日付を返します。

例題

次のプロシージャによってバックアップが定期的に行なわれているかどうかを確かめることができ、一番最近に行なったバックアップの日付を表示します。

```
    C_INTEGER ($Backup ; $Period)
    C_DATE ($DateBackup)
    $Backup:=BK Bigin full backup
    If ($Backup=0)
⇒       $DateBackup:=BK Get last backup date
        If ($DateBackup=!00.00.00!)
            ALERT ("このデータベースはバックアップを行なったことがありません。")
        Else
            $Period:=Current date -$DateBackup
            If ($Period>0)
                ALERT ("最後のバックアップは"+String ($Period)+"日前です。")
            End if
        End if
    End if
    BK END BACKUP
```

参照

BK Get last backup hour

BK Get last backup hour

BK Get last backup hour→倍長整数

説明

この関数によって最後にバックアップを行なった時間を調べることができます。

返される値は、夜の12時（?00:00:00?)からバックアップが終わるまでの時間の秒数です。データベースのバックアップを行なったことがない場合、BK Get last backup hour関数は0を返します。

例題

1. 返された値から時間を割り出すためには、次のフォーミュラを使用します。

```
    C_LONGINT ($vHour ; $vMinutes ; $Seconds)
    If (BK Begin full backup=0)
⇒       $vHour:=BK Get last backup hour / 3600
⇒       $vMinutes:=(BK Get last backup hour /%3600) / 60
⇒       $vSeconds:=BK Get last backup hour /%60
    End if
    BK END BACKUP
```

2. 次のプロシージャは、最後のバックアップから8時間以上経過している場合に警告メッセージを表示します。

```
    C_INTEGER ($vDuration)
    If (BK Begin full backup=0)
⇒       $vDuration:=Current date -BK Get last backup date
        If ($vDuration=0)
⇒           If ((Current time -BK Get last backup hour)>?08:00:00?)
                ALERT ("この8時間以内にデーターベースのバックアップが
                                                    行なわれていません。")
            End if
        Else
            ALERT (String ($vDuration)+"日間"+"データーベースのバックアップが
                                                    行なわれていません。")
        End if
    End if
    BK END BACKUP
```

注：Macintosh版の4th Dimensionをご使用の場合、時間記号のデフォルト文字は「?00:00:00?」ではなく「!!00:00:00!!」です。

参照

BK Get last backup date

BK GET SIZES

BK GET SIZES (データ;ストラクチャ;ログ;同封;合計)

引数	タイプ		説明
データ	倍長整数型の変数	←	データファイルのバイト単位のサイズ
ストラクチャ	倍長整数型の変数	←	ストラクチャファイルのバイト単位のサイズ
ログ	倍長整数型の変数	←	ログファイルのバイト単位のサイズ
同封	倍長整数型の変数	←	同封ファイルのバイト単位のサイズ
合計	倍長整数型の変数	←	バックアップのバイト単位の合計サイズ

説明

このコマンドは、バックアップされたファイルのバイト単位でのサイズを渡します。

引数<同封>変数は、同封ファイルのバイト単位のサイズの合計と同じ値を受け取ります。

引数<合計>変数は、アーカイブするバックアップファイルの合計サイズを受け取ります。バックアップファイルはそれについてのヘッダ部分が含まれているので、サイズが元々のサイズの合計よりも大きくなることに注意してください。データの照合や修正のためのコードもまたアーカイブのサイズを数パーセント単位で増やします。

例題

次のプロシージャは、バックアップを格納するのに十分なスペースを持つボリュームのみ表示します。

```
   C_INTEGER ($Backup ; $vErr;$i)
   C_LONGINT ($Capacity ; $Use ; $Avail ; $Data ; $Structure ; $Log ; $Encl ; $Tot)
   $vErr:=BK Begin full backup
   If ($vErr=0)
      ARRAY STRING (32 ; TabVolume)
      BK GET VOLUME SIZE (TabVolume)
⇒     BK GET SIZES ($Data ; $Structure ; $Log ; $Encl ; $Tot)
      For ($i ; 2 ; Array size (TabVolume))
            `フロッピーディスクドライブをスキップするため2からループを始める
         BK GET VOLUME SIZE ($i ; $Capacity ; $Use ; $Avail)
         If ($Avail > ($Tot+100000))
               `追加可能な100 Kの空がある
            CONFIRM ("そのボリューム上でバックアップを作成しますか
                                                    :"+TabVolume{$i})
            If (OK=1)
               BK SET VOLUME ($1)
               $vErr:=BK Start copy
```

```
                Repeat
                Until (BK Get state # 4)
                $i:=Size of array (TabVolume)
                   `ループから強制的に抜ける
             End if
          End if
       End for
    End if
    BK END BACKUP
```

参照

なし

BK Begin mirror update

BK Begin mirror update→整数

説明

すべての4D Backupのコマンドと関数は、**BK Begin mirror update**関数（または**BK Begin full backup**関数）と**BK END BACKUP**コマンドに囲まれていなければなりません。この規則の対象外となる独立したルーチンは4つしかありません（第1章「このマニュアルについて」を参照してください）。

BK Begin mirror update関数は、ミラー更新のプロセスを開始します。この関数を呼び出すと、カレントログファイルの最後にMacintosh上では「.2」、Windows上では拡張子「.4L2」を付けて返信し、そのログファイルをクローズします。すると、新しいログファイルが作成されます。4D Serverを使用している場合、新しいログファイルはデータベースで行なわれる処理の保存を開始します。ミラーへのログファイルの送付は、データベースの機能を中断することなく実行されます。

新しいログファイル（“.2”または“.4L2”）が作成されると、古いログファイルは機能しなくなります。このファイルは、**BK Launch copy**関数を使用してミラーを実行すると送付されます。

ログファイルが統合されると、ミラーデータベースのデータファイルはミラーマシン上でバックアップされます。

BK Begin mirror update関数は、正しく実行された場合、0を返します。そうでない場合はエラーコードを返します（エラーコード表は「付録A」を参照してください）。システム的にリターン値の検査を行なうことをお勧めします。**BK Begin mirror update**関数が0以外の値を返した場合、それ以降のステートメントは無視され、ミラー更新は行なわれません。

ミラー更新が実行されなかったり、あるいは実行中に**BK END BACKUP**コマンドが呼び出されると、2つのログファイルはログファイルを前の状態に戻すために結合されます。

更新中のデータベースへのアクセス

4D Serverを使用している場合、更新プロセスはデータベースにロックをかけません。つまり、接続しているすべてのクライアントは作業（データの追加、更新、削除）を続けることができます。

4th Dimensionを使用している場合、フルバックアップと同じように、更新を始めたプロセス以外のすべてのプロセスは中止されます。

トランザクションの扱い

BK Begin full backup関数を使用する場合と同じように**BK Begin mirror update**関数は、カレントトランザクションがすべてクローズするまで実行されません。フルバックアップと同じように、このコマンドを呼び出す前にミラー更新と同じプロセスでトランザクションが開始されていないかチェックする必要があります。

説明

1. 下記はミラー更新を実行するための最も単純なプロシージャーです。

```
    C_INTEGER ($vError)
⇒   $vError:=BK Update mirror
```

2. 次のプロシージャは、更新プロセスを管理します。

```
⇒   If (BK Begin mirror update = 0)
      If (BK Start copy = 0)
        Repeat
        Until (BK Get state # 4)
      End if
      BK END BACKUP
    End if
```

3. 発生する可能性があるエラーを処理することによって、前出のプロシージャを改善することができます。

```
    C_INTEGER ($vError ; $vState)
⇒   $vError:=BK Begin mirror update
    If ($vError=0)
      ALERT ("ミラー更新を開始できません：エラー番号:" +String ($vError))
    Else
      $vError:=BK Start copy
      If ($vError#0)
        ALERT ("ログファイルを送信できません。エラー番号" +String ($vError))
      Else
        Repeat
          $vState:=BK Get state
        Until ($vState#4)
        If ($vState#5)
```

```
            ALERT ("コピー中に障害が起きました：エラー番号" +String ($vState))
         End if
      End if
      BK END BACKUP
   End if
```

参照

BK END BACKUP、BK Begin full backup

BK Update mirror

BK Update mirror→整数

説明

既に少なくとも1つのミラー更新を行い、プロジェクトのパラメータを保存している場合、この関数を使って、次回からのミラー更新を自動化することができます。

BK Update mirror関数は、独立しています。**BK Begin mirror**関数と**BK END BACKUP**コマンドで囲まれている必要はありません。バックアッププロジェクトが正しいパラメータを持ち、しかもミラーデータベースがネットワーク上に存在する限り、この関数を呼び出してミラー更新を実行することができます。

通常、この関数はミラー更新を自動的に行なうプロセスから呼び出すことができます。

この関数によって返された整数コードによって、下記のコードに基づいて、更新中に障害が発生したかどうかを検査することができます。

コード	説明
0	障害はありませんでした。
1401	ミラーが間違っています（このミラーは存在しますが、データベースのオープンと一致していません）。
1402	ミラーが見つかりません（このミラーは存在しないか、あるいは使用できません）。
1403	ログファイルがありません（データベースがログファイルなしで動作しています）。

エラーコード表の詳細は、付録Aを参照してください。

参照

なし

旧コマンド

BK GET ZONE LIST (ゾーン配列)

BK SET ZONE (ゾーン名)

BK GET MIRROR LIST (ミラー配列)

BK Set mirror (データベース名 ; オーナー名 ; マシン名)→整数

バージョン1.5の4D Backupから通信用に標準の4Dネットワークコンポーネントを使用します（ネットワークコンポーネントに関する詳細は、製品のインストールガイドを参照してください)。上記ルーチンは、バージョン1.5以前の4D Backupとの互換性を保証するためにバージョン1.5では単に変換されるだけです。これらは次のバージョンの4D Backupから削除される予定です。

プロジェクト：はじめに

4D Backupで実行されるすべてのバックアップは、いくつかのパラメータによって定義されます。これらのパラメータには、次のものが含まれます。

- バックアップファイル（ストラクチャファイル、データファイル、ログファイル、同封ファイル）
- バックアップ先（ボリューム、パス名）
- バックアップオプション（検査、消去など）

4D Backupでは、下記のようなアイコンで表される「バックアッププロジェクト」という名前の書類に、これらのパラメータを保存することができます。

あるバックアップから他のバックアップへバックアップパラメータを保存するために、このファイルを保存する、または再使用することができます。この章では、下記のコマンドを使用してプロジェクトのオープンや保存、およびプロジェクトのパラメータ定義を行うことができます。

BK ADD ENCLOSURE

BK ADD ENCLOSURE (**ファイル名**)

引数	**タイプ**		**説明**
ファイル名	文字列	→	追加する新しいファイル名

説明

このコマンドによって、同封ファイルのリストにファイルを追加することができます。ファイル名のみ（データベースと同じフォルダ内にそのファイルがある場合）あるいは、フルアクセスパスが前に付いている名前（ファイルが他の場所にある場合）を渡します。

指定された書類が有効でない場合、そのコマンドは何も行いません。

状況によって、BK Get error関数は、下記の値の1つを返します。

コード	**説明**
1201	このファイルは追加できません。これはデータファイルかストラクチャファイル、あるいはログファイルです。
1202	このファイルはすでに同封ファイルのリストに存在します。
1203	このファイルは追加できません。同封ファイルが最大数に達しています。
< 0	システムエラーが起きました。例えば、エラーコード-43はファイルが見つからないことを示します。

エラーコード表の詳細は、付録Aを参照してください。

4D Serverユーザの方へ：同封ファイルはサーバマシンからアクセスしなければなりません。そのファイルはサーバマシン上、あるいはサーバマシンからアクセス可能な共有ボリューム上に置くことができます。

例題

次の例は、標準の「ファイルオープン」ダイアログボックスを使って、ユーザが選んだ書類を同封ファイルのリストに追加します。4th Dimension（シングルユーザ）でのみ使用できます。

```
C_REAL ($Ref)
C_TEXT ($FileName)
If (BK Begin full backup = 0)
   $Ref:=Open document ("")
      `「ファイルオープン」ダイアログボックスを表示する
   If(OK=1)
         `ユーザがそのダイアログボックスが使える場合
      $FileName:=Document
         `「Document」システム変数から読み込む
         `「Document」は選択されたファイル名
      CLOSE DOCUMENT ($Ref)
         `書類をクローズする
      BK ADD ENCLOSURE ($FileName)
         `リストに追加する
      BK SAVE PROJECT
         `プロジェクトを保存する
   End if
   BK END BACKUP
End if
```

参照

BK GET ENCLOSURES、BK REMOVE ENCLOSURE

BK GET ENCLOSURES

BK GET ENCLOSURES(ファイル配列)

引数	タイプ		説明
ファイル配列	文字列配列	←	同封ファイルのリスト

説明

このコマンドは、同封ファイルのリストを文字列配列に格納します。また、レイアウト内に同封ファイルのリストを表示することができます。アクセスパスではなく、それぞれのファイル名のみが与えられます。

引数＜ファイル配列＞に格納するには、その配列要素の長さを定義しなければなりません。その長さは、使用しているシステムで許可されているファイル名の最大サイズに対応しています。配列要素の長さを定義しないと、その配列には格納されません。したがって、データベースが複数のプラットフォーム上で使用されている場合は、配列に255バイトを設定するのが無難です。

■ **MacOS：**31バイト

■ **Windows 95、Windows NT：**255バイト

4D Serverユーザの方へ：同封ファイルのリストは、クライアントからではなく、サーバマシンから読み出されます。この同封ファイル名は、サーバマシン上またはサーバにアクセス可能な共有ボリューム上のファイルと一致します。

例題

次の例は、同封ファイルがない場合にアラートを表示します。

```
   If (BK Begin full backup = 0)
      ARRAY STRING (255 ; ArrAssocFiles ; 0)
⇒     BK GET ENCLOSURES (ArrAssocFiles)
      If (Size of array (ArrAssocFiles) = 0)
         ALERT ("同封ファイルがありません。")
      End if
      BK END BACKUP
   End if
```

参照

BK ADD ENCLOSURE、BK REMOVE ENCLOSURE

BK GET NAMES

BK GET NAMES (データ;ストラクチャ;ログ)

引数	タイプ		説明
データ	文字列型の変数	←	データファイル名
ストラクチャ	文字列型の変数	←	ストラクチャファイル名
ログ	文字列型の変数	←	ログフィアル名

説明

このコマンドは、データーベースを構成する異なるファイル名を、引数として渡される文字列型の変数内に返します。例えば、この情報は、カスタムバックアップダイアログボックスを作成するのに使用されます。

アクセスパスはなく、単にファイル名だけを受け取ります。

データファイルが分割されている場合、最初のセグメント名が引数＜データ＞変数に返されます。

ログファイル名が空の文字列の場合、データベースはログファイルなしで動作します。

例題

次のプロシージャは、ユーザがログファイルなしで動作させている場合にアラートを表示します。

```
    If (BK Begin full backup  = 0)
       C_STRING (80 ; $Data ; $Structure ; $Log)
⇒      BK GET NAMES ($Data ; $Structure ; $Log)
       If ($Log="")
          ALERT ("注意！ログファイルなしで動作しています。")
       End if
       BK END BACKUP
    End if
```

参照

なし

BK GET OPTIONS

BK GET OPTIONS (データ;ストラクチャ;ログ;検査;消去;セット数;削除;加算)

引数	タイプ		説明
データ	整数型の変数	←	データファイルのバックアップ
ストラクチャ	整数型の変数	←	ストラクチャファイルのバックアップ
ログ	整数型の変数	←	ログファイルのバックアップ
検査	整数型の変数	←	書き込み時間におけるデータの検査
消去	整数型の変数	←	バックアップ前のディスクの消去
セット数	整数型の変数	←	バックアップのセット数
削除	整数型の変数	←	一番古いアーカイブ（バックアップ）の削除
加算	整数型の変数	←	バックアップセット番号の加算

説明

このコマンドは、カレントバックアップのオプションを返します。

引数＜データ＞、＜ストラクチャ＞、＜ログ＞の各変数は、下記のコードを基にしてバックアップの対象として選択されたファイルと一致しているかどうかを示します。

■ 変数が1の場合、そのファイルはバックアップされます。

■ 変数が0の場合、そのファイルはバックアップされません。

引数「検査」、「消去」の各変数は、下記のコードを元にして、選択されたオプションと一致するかどうかを示します。

■ 変数が1の場合、そのオプションは選択されています。

■ 変数が0の場合、そのオプションは選択されていません。

これらのオプションは、「フルバックアップ」ウインドウ内にあるチェックボックスと同じものです。引数＜消去＞は、フロッピーディスク以外の取り外し可能なディスクでのバックアップにのみ影響することに注意してください。4D Backupはフロッピーディスクを消去し、システム的に名前を変更します。一方、4D Backupは取り外し不可のハードディスクの消去は行ないません。

「セット数」変数は、指定したセット数を返します。デフォルトのセット数は3です。この数は、プロシージャや「フルバックアップ」ウインドウ上で変更することができます。

引数＜セット数＞は読み出され、データベースのデータファイルに格納されることに注意してください。複数のプロジェクトを使用することによって、不注意でアーカイブを消去してしまわないようにします。「セット数」変数によって返される値は、データベースのすべてのプロジェクトに対して同じです。

引数<削除>が1の場合、一番古いアーカイブは新規バックアップを開始する前に削除されます。<削除>が0の場合、一番古いアーカイブはが新規バックアップの終了後に削除されます。

引数<加算>が1の場合、バックアップ番号はストラクチャだけがバックアップされる場合にのみ加算されます。<加算>が0の場合、バックアップ番号は加算されません。

参照
BK SET OPTIONS

BK OPEN PROJECT

BK OPEN PROJECT (プロジェクト名)

引数	タイプ		説明
プロジェクト名	文字列	→	オープンするプロジェクト名

説明

このコマンドは、引数＜プロジェクト名＞で指定された名前の書類をオープンします。この書類は、4D Backupのインタフェースや**BK SAVE PROJECT**コマンドから直接生成されたかどうかに関係なく、4D Backupによって生成されるプロジェクトでなければなりません。プロジェクトがデータベースと同じフォルダにある場合、書類名（例えば「Daily backup」)）だけを指定することができます。データベースと同じフォルダにプロジェクトがない場合、フルパス（例えば、Macintosh上では「Hard Disk:Folder:Daily Backup」、Windows上では、「C:\Folder\Backupdl.4BP」）を指定しなければなりません。

BK Get error number関数は、オープンの際にプロジェクトと密接に関連しているエラーを管理することができます。**BK OPEN PROJECT**コマンドの後に**BK Get error number**関数を呼び出すと、下記の値のひとつを受け取ります。

コード	説明
0	書類のオープンにエラーはありません。
1101	プロジェクトが間違っています（そのプロジェクトはオープンするデータベースと一致しません)。
1102	プロジェクトではありません（指定されたファイルはプロジェクトではありません)。

エラーコードの詳細は、付録Aを参照してください。

もうひとつの4D Backup関数である**BK Get state**によってプロジェクトファイルに指定されたボリュームと密接に関連しているエラーを管理することができます。**BK OPEN PROJECT**コマンドを呼び出しすと、**BK Get state**関数によって下記のステータスの1つが返されます。

コード	メッセージ	説明
1	「ディスクが選択されていません」	プロジェクトで選択されたボリュームが存在しない。
2	「このディスクは使用できません」	例えば、コピーするだけの十分なスペースをそのボリュームが持っていない。
3	「コピーの準備ができています」	バックアップを実行する準備ができている。

最初の2つのケースでは、使用可能で十分なサイズを持ったボリュームを探さなければなりません。例えば、ユーザが別のディスクを選択できるように「フルバックアップ」ウインドウを表示することができます。

4D Serverユーザの方へ：このプロジェクトは、サーバマシン（クライアントではありません）で読み出します。名前とアクセスパスは、サーバマシンあるいはサーバマシンからアクセスできる共有ボリュームに置かれたプロジェクトに使用されます。

例題

週の7日間において、それぞれ7つの異なったバックアッププロジェクトを持っているとします。このシステムは、各曜日に異なったディスク上にバックアップを作成し、週末を利用して多くの同封ファイルのバックアップを作成することができます。このプロジェクトは、Macintosh上では「Day 1」、「Day 2」、Windows上では「Day 1.4BP」、「Day 2.4BP」などと名付け、データベースと同じフォルダに置かれます。

4th Dimensionでは日曜日＝1、月曜日＝2...となることを忘れないでください。

```
   C_INTEGER ($Error)
   If (BK Begin full backup # 0)
      ALERT ("バックアップを開始できません")
   Else
⇒     BK OPEN PROJECT ("Daily backup "+String (Day of (Current date)))
      $Error:=BK Get error number
      If ($Error # 0)
            `エラーが起きた場合
            ALERT ("次のプロジェクトをオープンする際にエラーが起きました：
                                                "+BK Get error text($Error))
      Else
         If (BK Get state # 3)
               `「コピーの準備ができて」いない状態の場合
               ALERT ("このプロジェクトは間違っています。データベースは
                                                バックアップできません。")
         Else
            If (BK Start copy = 0)
                  `コピーが実行される場合
               Repeat
```

```
            Until (BK Get state # 4)
               `コピーが終了するまで待機
         End if
      End if
   End if
   BK END BACKUP
End if
```

参照

BK SAVE PROJECT、BK Get error number、BK Get error text、BK Get state

BK REMOVE ENCLOSURE

BK REMOVE ENCLOSURE (ファイル名)

引数	タイプ		説明
ファイル名	文字列	→	リストから削除したいファイル名

説明

このコマンドは、指定されたファイルを同封ファイルのリストから削除します。このファイルはディスクから消去されるのではなく、単にリストから削除されるだけです。

そのファイルがリストにない場合は、このコマンドは何も行いません。また、**BK Get error number**関数がエラーコード1204を返します。

注：フルパスではなく、書類名だけを指定しなければなりません。

参照

BK GET ENCLOSURES、**BK ADD ENCLOSURE**

BK SAVE PROJECT

BK SAVE PROJECT ({プロジェクト名})

引数	タイプ		説明
プロジェクト名	文字列	→	保存するプロジェクト名

説明

このコマンドは、引数<プロジェクト名>で指定された名前のプロジェクトを保存します。この名前を持つプロジェクトが存在しない場合は、プロジェクトが作成されます。既に存在する場合は、 その内容が新しいパラメータに置き換えられます。保存されていたパラメータはデフォルトのパラメータや開いたプロジェクトのパラメータ、あるいはランゲージコマンドによって変更されたパラメータかに関わらず、メモリ上で新しいパラメータになります。

書類名（例えば、Macintosh上で「Daily backup」、Windows上で「Backupdl.4BP」）のみ渡すことができたり、あるいはフルパス（例えば、Macintosh上で「HardDisk:Folder:daily backup」、Windows上で「C:\Folder\Backipdl.4BP」）を持った名前を優先することができます。前者の場合、プロジェクトはデータベースと同じフォルダに保存されます。

引数<プロジェクト名>を省略した場合、カレントプロジェクトはデフォルトのプロジェクト（「バックアッププロジェクト」）、またはBK OPEN PROJECTコマンドによってオープンされたプロジェクトかに関わらず、保存されます。先にオープンされたプロジェクトがない場合、デフォルトプロジェクト（Macintosh上では「バックアッププロジェクト」、Windows上では「Backup.4BP」）が保存されます。

BK Get error number関数がBK SAVE PROJECTコマンドを使用した後に呼ばれると、下記の値のひとつを返します。

コード	説明
0	書類が正しく保存されました。
< 0	システムエラーが起きました。例えば、エラーコード-34はディスクに空きがないことを示しています。

4D Serverユーザの方へ：プロジェクトはクライアントではなくサーバマシンで保存されます。フルパス名で指定する場合は、サーバマシンのディスクか、サーバマシンに対して使用可能な共有ボリュームと同じ名前にする必要があります。

参照

BK OPEN PROJECT、BK Get error number

BK SET OPTIONS

BK SET OPTIONS (データ;ストラクチャ;ログ;検査;消去;セット数;削除;加算)

引数	タイプ		説明
データ	整数型の変数	→	データファイルのバックアップ
ストラクチャ	整数型の変数	→	ストラクチャファイルのバックアップ
ログ	整数型の変数	→	ログファイルのバックアップ
検査	整数型の変数	→	書き込みの時間におけるデータの検査
消去	整数型の変数	→	バックアップ前のディスクの消去
セット数	整数型の変数	→	バックアップのセット数
削除	整数型の変数	→	一番古いアーカイブ（バックアップ）の削除
加算	整数型の変数	→	バックアップセット番号の加算

説明

このコマンドは、カレントバックアップのオプションを設定します。

引数＜データ＞、＜ストラクチャ＞、＜ログ＞の各変数は、下記のコードをもとにして、バックアップの対象として選択されたファイルであるかどうかを示します。

■ 変数が1の場合、そのファイルはバックアップの対象になります。

■ 変数が0の場合、そのファイルはバックアップの対象になりません。

■ 変数が-1の場合、前の設定をそのまま残します。

同様に、引数＜検査＞、＜消去＞は次のように設定されます。

■ 変数が1の場合、そのオプションは選択されます。

■ 変数が0の場合、そのオプションは選択されません。

■ 変数が-1の場合、前の設定をそのまま残します。

これらのオプションは、「フルバックアップ」ウインドウ内にあるチェックボックスと同じものです。引数＜消去＞は、フロッピーディスク以外の取り外し可能なディスクでのバックアップにのみ影響することに注意してください。デフォルトでは、4D Backupはフロッピーディスクを消去し、名前を変更します。一方、4D Backupは取り外し不可のディスクを消去しません。特定のボリュームが取り外し可能かどうかを判断するには、**BK GET VOLUME INFO**コマンドを使用してください。

引数＜セット数＞は、保存するためのバックアップのセット数を設定します。その値はどの番号も1から100の間です。マイナスの値を指定した場合は、前の設定が使用されます。

引数＜セット数＞はプロジェクトで定義されますが、データベースのデータファイル内に格納されることに注意してください。複数のプロジェクトを使用することにより、不注意でアーカイブを消去してしまわないようにします。＜セット数＞へ設定される値は、データベースのすべてのプロジェクトに対して適用されます。

引数＜削除＞が1の場合、一番古いアーカイブは新規バックアップを開始する前に削除されます。＜削除＞が0の場合、一番古いアーカイブは新規バックアップの終了後に削除されます。＜削除＞が-1の場合、前回の値はそのままです。

引数＜加算＞が1の場合、バックアップ番号はストラクチャだけがバックアップされる場合にのみ加算されます。＜加算＞が0の場合、バックアップ番号は加算されません。＜加算＞が-1の場合、前回の値はそのままです。

参照

BK GET OPTIONS

ユーティリティ：はじめに

この章では、エラーを管理するルーチンについて説明します。

BK Get error number

BK Get error number→整数

説明

この関数は、4D Backupによって出力された最新のエラーコードを返します。一番最近に実行したコマンドがエラーを出さなかった場合、**BK Get error number**関数は0を返します。

この関数は、各コマンドの後で呼ぶべきでしょう。

返されたエラーコードが正の数の場合、それは4D Backupのエラーです。4D Backupのエラーコード表が「付録A」にあります。

エラーコードが負の数の場合、これはシステムエラーです（例えば、エラーコード-34はディスクに空きがないことを示します）。頻繁に発生するシステムエラーの大部分は、「付録A」にリストが挙げられています。

注：プラットフォームに依存しないデータベースを保証するには、4D Backupによって返されるエラーコードはMacintoshとWindows上で同じでなければなりません（エラーコードはMacintosh用）。場合によっては、4D Backupは何らかの変換処理を行います。詳細は、付録Aを参照してください。

参照

なし

BK Get error text

BK Get error text (エラーコード）→テキスト

引数	タイプ		説明
エラーコード	整数	→	エラーコード
戻り値	テキスト	←	エラーのテキスト

説明

この関数は、発生したエラーの内容を説明するテキストを返します。このテキストは障害の原因をユーザに知らせるために使用します。このコマンドは、4D Backupのエラーだけに機能します（エラーコードの正の数を持っているものだけ）。

エラーテキストは、エラーコードと一緒に「付録A」に掲載されています。

例題

次の例は、プロジェクトファイルのオープンにおける一般的なエラー操作です。

```
C_INTEGER ($Error)
If (BK Begin full backup # 0)
   ALERT ("バックアップを開始できません。")
Else
   BK OPEN PROJECT ("Daily backup")
   $Error:=BK Get error number
   Case of
      \($Error>0)
⇒        ALERT(BK Get error text ($Error))
      \($Error=-43)
         ALERT ("プロジェクトが見つかりません。")
      \($Error=-49)
         ALERT (プロジェクトはすでに他のアプリケーションでオープン
                                                されています。")
      \($Error<0)
         ALERT ("エラー番号"+String ($Error+Char (13)+"プロジェクトを
                                                オープンできませんでした。")
      \($Error=0)
         ...     `バックアップの実行
   End case
   BK END BACKUP
End if
```

参照

BK Get error number

ボリューム：はじめに

これらのルーチンは、バックアップボリュームの情報を提供します。選択したボリュームの数を調べる、接続したボリュームのリストを得る、ボリュームを変更する、あるいは特定ボリュームの情報を入手することができます。

多くのルーチンはボリューム番号を引数として持っています。このボリューム番号が**BK GET VOLUME LIST**コマンドによって返されたテーブル内のボリュームのインデックスに相当します。

BK EJECT DISK

BK EJECT DISK (ボリューム番号)

引数	タイプ		説明
ボリューム番号	整数型の変数	→	**BK GET VOLUME LIST**コマンドで返されるリスト内のボリューム番号

説明

このコマンドは、＜ボリューム番号＞によって指定されたボリュームを取り出します。

BK EJECT DISKコマンドは、フロッピーディスクドライブのような取り出し可能なボリュームに対してのみ使用します。指定したボリュームが取り出し可能かどうかを調べるためには、**BK GET VOLUME INFO**コマンドを使用します。

注：複数の取り出し可能なボリュームに分割されたバックアップを作成している場合、4D Backupは自動的にすべてのボリュームを取り出し、挿入されたボリュームを使用します。これを定期的に制御する必要はありません。

参照

BK GET VOLUME INFO、BK GET VOLUME LIST

BK Get filename

BK Get filename→文字列

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	テキスト	←	バックアップファイルの現在の名前

説明

この関数は、現在のバックアップファイルの名前を返します。Macintosh上では、角カッコで囲まれたバックアップ番号の後ろに付いたバックアップファイルの現在の名前を返します。例えば、“ Base.data[21] ”。Windows上では、バックアップ番号は拡張子 “ .4BK ” の前に付け加えられます。例えば、“ Base021.4BK ”。

この関数は、バックアップファイルに対してフルパス名を返しません。

参照

BK SET FILENAME

BK Get volume

BK Get volume→整数

引数	タイプ		説明
この関数には、引数はありません。			
戻り値	整数	←	現在のバックアップのために選択されたディスクの数

説明

この関数は、カレントバックアップに対して選択されたディスクの番号を返します。

参照

BK SET VOLUME、BK GET VOLUME LIST

BK Get volume icon

BK Get volume icon (ボリューム番号)→ピクチャ

引数	タイプ	説明
ボリューム番号	整数型の変数 →	**BK GET VOLUME LIST** コマンドによって返されるリスト内のボリューム番号

説明

BK Get volume icon コマンドは、特定ボリュームのアイコンを含む4th Dimensionのピクチャを返します。この関数によってユーザが使用可能なボリュームを確認できるカスタムダイアログボックスを表示することができます。これの特長は、4D Backupのメインウィンドウ内に表示されるものと同様のインタフェースを表示することができる点です。

例題

次の例は、使用可能なボリュームのリストを持つピクチャ配列を作成します。カスタマイズされた「バックアップ」ダイアログボックス内のボタンスクリプトに配置すると、ボリュームリストのアイコンを表示します。配列行がピクチャイメージを表示するのに十分なスペースを持つように、このダイアログボックスに対して32ポイントのフォントサイズを選択します。

```
C_INTEGER ($i)
ARRAY STRING (32 ; ArrayVol ; 0)
If (BK Begin full backup # 0)
   ALERT ("バックアップができません。")
Else
   BK GET VOLUME LIST (ArrayVol)
   ARRAY PICTURE (ArrayIcon ; Size of array (ArrayVol))
   For ($i ; 1 ; Size of array (ArrayVol))
⇒     ArrayIcon{$i}:=BK Get volume icon ($i)
   End for
End if
BK END BACKUP
```

参照

なし

BK GET VOLUME INFO

BK GET VOLUME INFO (ボリューム番号 ; ボリューム名 ; ロック ; 取り出し可 ; 存在 ; ボリュームタイプ)

引数	タイプ		説明
ボリューム番号	整数型の変数	→	BK GET VOLUME LISTコマンドによって返されるリスト内のボリューム番号
ボリューム名	文字列型の変数	←	ボリューム名
ロック	整数型の変数	←	ボリュームがロックされているかどうか
取り出し可	整数型の変数	←	ボリュームは取り出しできるかどうか
存在	整数型の変数	←	イジェクト可のボリュームはあるかどうか
ボリュームタイプ	整数型の変数	←	ボリュームタイプ

説明

このコマンドによって引数＜ボリューム番号＞で指定されたボリュームに関する情報を得ることができます。

引数＜ボリューム名＞は、ボリューム名を受け取ります。Macintosh上では、Finder上に現れるボリューム名のことで、そのボリューム名の長さは最大27バイトです。Windows上では、ボリューム名は後ろにコロン（“ C: ”）が続くボリュームの文字です。

引数＜ロック＞は、ボリュームがロックされているかどうかを示します。＜ロック＞が1の場合、ボリュームはロックされており、修正をかけることができません。そのため、このボリュームをバックアップとして選択することはできません。この場合、**BK Get state**関数が2（選択されたディスクが使用できない）を返します。＜ロック＞が0の場合、そのディスクはロックされておらず、バックアップに使用することができます。

引数＜取り出し可＞は、ボリュームが取り出しできるかどうかを示します。＜取り出し可＞が1の場合、指定したボリュームは取り出しすることができます。この場合、バックアップの前にディスクを消去するために**BK SET OPTIONS**コマンドを使うことができます。また、**BK EJECT DISK**コマンドを使ってもディスクを消去することができます。＜取り出し可＞が0の場合、ボリュームは取り出すことができません。

引数＜存在＞は、取り出し可能なボリュームにディスクが存在するかどうかを示します。＜存在＞が0の場合、ディスクは存在します。取り出し不可のディスクは、1を返します。

引数＜ボリュームタイプ＞は、以下のコード表に基づいて、そのボリュームタイプの情報を返します

コード	ボリュームタイプ
1	ハードディスク
3	800K フロッピーディスクドライブ
4	1.4MB フロッピーディスクドライブ

参照

BK GET VOLUME LIST

BK GET VOLUME LIST

BK GET VOLUME LIST (ボリューム名配列)

引数	タイプ		説明
ボリューム名配列	文字列	←	ボリューム名の配列

説明

このコマンドは、引数<ボリューム名配列>に接続したボリューム名を格納します。この配列は、それぞれ32バイトまでのボリューム名を持つことができます。その配列がすでに要素を持っている場合、これらの要素は消去され、ボリューム名リストによって置き換えられます。

配列に格納されるボリュームの順番は、「フルバックアップ」ウインドウに表示される順番と同じです。

- Macintosh上では、最初にフロッピーディスクドライブをマウントし、次に起動ディスク、そして他のボリュームをマウントします。しかし、あるバックアップから他へバックアップした場合に保存されるボリューム番号は保証されないことに注意してください。例えば、外部ディスクの電源が切られたり、ディスクのSCSI番号が変更されたりすると、順番は変わります。
- Windows上では、ボリュームの順番は（A:、B:、C:...）のように定義されたボリューム文字の順番と一致します。

バックアップボリュームの順番はとても重要です。4D Backupのコマンドを使ってボリュームを操作する場合、特定ボリュームを参照するのに配列要素の番号を使用します。例えば、ボリューム名「MyBackupVol」が配列要素2の場合、ボリューム番号として2を渡すことによって、他のコマンド内でそのボリューム名を参照することができます。

例題

次の例は、ボリュームの場所やタイプに関係なく、ボリューム名「Backup」上でバックアップを行います。この方法でバックアップを行うことによって、接続しているボリューム名を変更するだけで、簡単にバックアップボリュームを変更することができます。

```
C_INTEGER ($BackupVol)
ARRAY STRING (32 ; ArrayVolNums ; 0)
If (BK Begin full backup # 0)
   ALERT ("バックアップできません。")
Else
⇒     BK GET VOLUME LIST (ArrayVolNums)
   $BackupVol:=Find in array (ArrayVolNums ; "Backup")
   If ($BackupVol=-1)
         ALERT ("ボリューム「Backup」は使用できません。バックアップは
                                                   できません。")
```

```
    Else
        If (BK Get volume # $BackupVol)
            `ボリューム番号を変える場合
            BK SET VOLUME ($BackupVol)
            `新しいボリューム番号をセットし
            BK SAVE PROJECT
            `新しいプロジェクトを保存する
        End if
        If (BK Start copy # 0)
            ALERT ("バックアップを起動できません。")
        Else
            Repeat
            Until (BK Get state # 4)
            `バックアップが終わるまで待機
            If (BK Get state # 5)
                ALERT ("バックアップ中に問題が発生しました。")
            End if
        End if
    End if
    BK END BACKUP
End if
```

参照

BK GET VOLUME INFO、BK GET VOLUME SIZE

BK GET VOLUME SIZE

BK GET VOLUME SIZE (ボリューム番号;最大サイズ;使用サイズ;使用可能サイズ)

引数	タイプ		説明
ボリューム番号	整数型の変数	→	GET VOLUME LISTコマンドによって返されるリスト内のボリューム番号
最大サイズ	倍長整数型の変数	←	ディスクの最大サイズ
使用サイズ	倍長整数型の変数	←	ディスクの使用サイズ
使用可能サイズ	倍長整数型の変数	←	ディスクの使用可能サイズ

説明

この関数は、引数＜ボリューム番号＞で指定されたボリュームのサイズ情報を示します。

引数＜最大サイズ＞は、指定されたボリュームの最大サイズを受け取ります。

引数＜使用サイズ＞は、そのボリューム上で現在、データが使用しているサイズを受け取ります。

引数＜使用可能サイズ＞は、指定されたボリュームの空きサイズを受け取ります。これはバックアップを行うのに可能なサイズです。

返される数値は、バイトで表されます。ディスクスペースは通常、メガバイト（MB）で計算されます。キロバイト（K）は1024バイト、メガバイト（MB）は1024K、ギガバイト（GB）は1024MBであることを覚えておいてください。漢字Talk 7.5以上から最大4GBまで管理することができます。また、Windows NTにおいて、ディスクスペースはとても重要になります。

参照

BK GET VOLUME LIST

BK SET FILENAME

BK SET FILENAME (ファイル名)

引数	タイプ		説明
ファイル名	文字列	→	バックアップファイルの名前

説明

このコマンドによってバックアップファイルの名前、および任意でファイルの格納場所を設定することができます。

フルパス名（例えば、「HardDisk:MyFolder:Backup」)を使ってファイル名を指定すると、バックアップを行うファイルがそのボリュームのどこへ格納されるかも指定します。

バックアップファイルの名前（例えば「Backup」)のみを指定すると、ファイルはそのボリュームの最上位階層に格納されます。

4D Backupは、バックアップ番号を自動的にファイルの最後に付け加えます。

- Macintosh上では、バックアップ番号は自動的にファイルの最後に角カッコ（[ ]）で囲んだ形で付け加えられます。例えば、35回目のバックアップのファイル名を「HardDisk:MyFolder:Backup」にした場合、4D Backupはディスク「HardDisk」の「Myfolder」というフォルダに「Backup[35]」というファイルを作成します。
- Windows上では、バックアップ番号は拡張子“.4BK”の前にあるファイル名に付け加えられます。拡張子を除くファイル名の長さは、母音削除機能により自動的に8バイト以下に切り取られます。例えば、35回目のバックアップのファイル名を「C:\Folder\Backup」にした場合、4D Backupはディスク「C」の「Folder」というフォルダに「Bckp035」というファイルを作成します。

参照

BK Get filename

BK SET VOLUME

BK SET VOLUME (ボリューム番号)

引数	タイプ	説明
ボリューム番号	整数型の変数 →	**BK GET VOLUME LIST**コマンドによって返されるリスト内のボリューム番号

説明

このコマンドによって、バックアップボリュームを変更することができます。4D Backupは、バックアップファイルの格納されている場所を調べるために使用されるパス名を再度初期化します。BK SET FILENAMEコマンドを使用して新しいパス名を指定しない場合、次回のバックアップは指定したディスクのルートレベルで行われます。

参照

BK Get volume、BK GET VOLUME LIST、BK SET FILENAME

付録 A ： 4D Backup のエラーコード

4D Backup によって返されるエラーコード

ルーチンを正常に実行すると、4D Backup は 0 を返します。そうでない場合は、エラーコードを返します。**BK Read error** 関数を使用することによって、返された最終エラーコードを受け取ることができます。いくつかの 4D Backup の関数は自動的にエラーコードを返します。**BK Get error text** 関数を使用することによって、エラーの内容を知ることができます。

次の表は、4D Backup によって返されるエラーコードの一覧です。

エラーコード	エラー内容
1000	エラーが発生しました（バックアップ中にエラーが起きました）。
1101	プロジェクトが間違っています（そのプロジェクトはオープンするデータベースと一致しません）。
1102	プロジェクトがありません（指定された書類は 4D Backup のプロジェクトではありません）。
1103	バックアップ先が間違っています（バックアップを行うファイルがありません）。
1201	同封ファイルが間違っています（このファイルはデータファイルかストラクチャファイル、あるいはログファイルなので追加できません）。
1202	同封ファイルが重複しています（このファイルはすでに同封ファイルのリスト内に存在しています）。
1203	同封ファイルが多すぎます（このファイルは同封ファイルの最大数に達しているため追加できません）。
1204	これは同封ファイルではありません（リストにないためこのファイルは同封ファイルから消去することはできません）。

エラーコード	エラー内容
1301	バックアップは既に開始しています（バックアッププロセスはすでにプロセスを作成されています）。
1302	バックアップはまだ開始していません（バックアッププロセスが作成されていません。**BK Begin full backup**または**BK Begin mirror update**を呼び出してください）。
1303	バックアップ先が正しくありません（バックアップ先ボリュームが見つからないか、アクセスパスが変更されています）。
1304	バックアップを開始できません（4D Backupが正しくインストールされているかを確認してください）。
1305	マニュアルバックアップは、新しいバックアップのセグメントを作成する必要があります（スケジューラまたは4D Backupコマンドでのバックアップをする場合に、新しいセグメントが必要であることをバックアップを表します）。
1306	4D Backupは4D Serverにインストールされていません（4D Backupをストラクチャファイルにと4D Serverのアプリケーションにインストールしなければなりません）。
1401	ミラーが間違っています（このミラーは存在していますが、オープンするデータベースと一致していません）。
1402	ミラーが見つかりません（このミラーは存在していないか、使用可能ではありません）。
1403	ログファイルがありません（このデータベースはログファイルなしで動作しています）。
1404	このプロセスでトランザクションは開かれています（4D Backupを呼び出す前にトランザクションが妥当なものかを確認するか、そのトランザクションをキャンセルしてください）。
1405	ログファイルを変えることはできません。
1406	古いログファイルが既に存在しています（ファイル「従業員.log.2」または「従業員.4L2」がすでに存在しています）。
1407	新しいログファイルが作成できません。
1408	新しいログが開けません。

システムからのエラーコード

BK Get error number関数が負のエラーコードを返した場合、エラーが使用しているシステムによって発見されたことを示します。

プラットフォームに依存しないデータベースを保証するには、MacintoshやWindowsシステムによって返される4D Backupのエラーコード同じものでなければなりません。

このリストは4D Backupの使用中に起こり得る共通のシステムエラーについて書かれています。

エラーコード	エラー内容
-34	ディスクに空きがありません。
-36	I/Oエラーです。「Disk Doctor」でディスクのチェックを行なってください。
-43	ファイルが見つかりません。
-44	ボリュームがハードウェアの設定によりロックされています。
-48	ファイル名が重複しています。
-49	ファイルはすでに書き込み許可でオープンされています。
-108	ヒープゾーンに十分なメモリ空間がありません。

付録 B ：復旧方法

ここでは、アクシデントが発生した際のデータベースの復旧方法について説明します。アクシデントの種類と使用しているバックアップの種類によって復旧処理は異なります。

データベースの実行が停止した場合

停電などの電源異常やシステムエラーなど、データベースはさまざまな原因で停止する可能性があります。

このようなアクシデントが発生した場合、復旧方法は4th Dimension のデータキャッシュの状態に応じて決まります。データキャッシュとはメモリ上のバッファのことで、データに関する処理を一時的に記録しておくためのものです。キャッシュが一杯になると、そのデータ処理はディスクに書き出されます。

アクシデントが発生した時の状態によって、次のように分けられます。

- キャッシュが空の場合（前回ディスクにキャッシュを書き出してから更新処理が行われていない）
- まだディスクに書き出していない更新処理がキャッシュに存在している場合

いずれの場合も4th Dimensionでデータベースを起動し、アクシデント発生時点のキャッシュの状態を見極める必要があります。各々の場合の見分け方と復旧方法について、次に説明します。

注：データキャッシュに関する詳細は、4th Dimensionのドキュメントを参照してください。

復旧手順

復旧の手順は、データキャッシュの状態に依存します：

キャッシュの状態	現象	データベースの復旧手順
キャッシュが空の場合	なし：データベースはいつものように開くことができます。	データベースに対して行ったすべての変更内容は保存されます。このケースでは、復旧処理を行う必要はありません。
キャッシュに更新処理が存在している場合	データベースに対して行われた最後のデータ処理が消えてしまいます。データベースがログファイルを含んで機能している場合、データベース起動時に4th Dimensionから次のような警告が表示されます：“ このログファイルはデータベースに保存されているデータよりも多く記録されています。” および “ データを戻したい場合は、最新バックアップからログファイルを復元する必要があります。”	**1.** 失われた処理を復元するためにカレントのログファイルを統合する。ログファイルを取っていない場合は、ディスクキャッシュへの最終書き出し後のデータ処理が失われます。 **2.** 4th Dimensionでデータベースを起動する。
キャッシュに書き出し中の場合	4th Dimensionでデータベースを開くことができなくなります。データファイルに問題があるという警告のダイアログが表示されます。	この場合、データベースの最新のバックアップを使う必要があります。 **1.** 最新のフルバックアップを復元する。あるいは、ミラーバックアップの場合、ミラーマシンからオリジナルデータベースのマシンへミラーデータベースをコピーする。 **2.** ログファイルを取っている場合は、このデータベースにカレントのログファイルを統合し、データ処理をすべて復旧する。

データベースファイルが失われた場合

ディスクセクタの欠損やウイルスによる被害、誤ったデータ消去など、データベースファイルはさまざまな理由から失われる可能性があります。

技術的な事柄に問題がある場合は、その問題点を見つけて解決することから始めます。例えば、ディスクユーティリティを利用して故障ディスクをつきとめたり、ウイルスを発見することです。

使用しているバックアップの種類によって復旧処理は異なります。

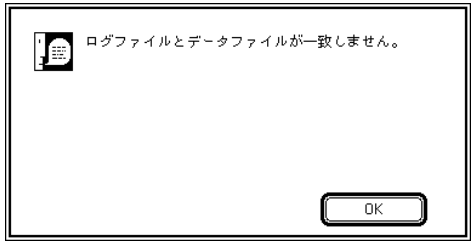
フルバックアップからの復旧

ここでは、フルバックアップからデータベースを復旧する手順について説明します。

1. データベースの最新のバックアップを使い、失われたファイルを復元する。

分割されたデータファイルに問題が発生した場合は、たとえ問題が起きたセグメントが1つの場合でも、セグメントのすべてを復元する必要があります。

失われたデータファイルを復元すると、データベースを開いた時に4th Dimensionから次のようなメッセージが表示されます。



2. この場合、復元したデータベースのカレントのログファイルを統合します。

注：最新のフルバックアップ以降、データファイルのバックアップを取らずにログファイルのバックアップだけしか取っていない場合は、ログファイルを作られた順に（“.a”、“.b”などの接尾語で示される）統合しなければなりません。

論理ミラーによる復旧

ミラーを使ったデータベースファイルの復旧は、次の手順で行います。

1. ミラーマシンからオリジナルデータベースのマシンへファイルをコピーする。

大規模なデータベースをマシン間でコピーする方法については、「付録A」に説明があります。

2. カレントログファイルをオリジナルデータベースに統合する。

ミラーの更新中にアクシデントが発生した場合

バックアップに論理ミラーを使っている場合は、ミラーの更新中にアクシデントが発生する可能性もあります。アクシデントが発生する原因は、停電やネットワークエラー、ディスクセクタの欠損などさまざまです。

この種のアクシデントはまれにしか起こりませんが、特別な復旧処理が必要です。次の点に対応して処理が異なります。

■ 2つのデータベースの状態（オリジナルとミラー）

■ アクシデントが発生した時点で、更新の行われていたフェーズ

データベースの状態

復旧の方針を決定する前に、オリジナルデータベースとミラーデータベースが使用可能であるかどうか調べる必要があります。

アクシデントが発生したのがデータキャッシュをディスクに書き出している最中なら、オリジナルデータベースはダメージを受けています。ミラーにログを統合している最中にアクシデントが発生したのであれば、ミラーデータベースがダメージを受けています。

2つのデータベースの状態を調べるには、4th Dimensionでデータベースを起動します。データベースが使用できない状態にある場合は、4th Dimensionから警告が出されます。

注：“ ログ統合 ”（Macintosh版）または “ Restore.4DL ”（Windows版）という名前のファイルがミラーデータベースのフォルダに存在する場合は、たとえ4th Dimensionで正常にデータベースを開くことができたとしても、データベースがダメージを受けていることを考慮する必要があります。

データベースの状態を調べる際、次の4種類の状況が考えられます。

■ 中間のログファイルはダメージを受けていないが、2つのデータベースはともに元のままの状態である。

■ オルジナルデータベースだけがダメージを受けている。

■ ミラーデータベースだけがダメージを受けている。

■ データベースは両方ともダメージを受けている。

ログファイルの統合状態

統合中のログファイルは、ミラーが更新されるたびに名前が変更されます。ログファイルの名前を調べれば、統合がどの時点で停止したかを知ることができます。

アクシデント発生後、“ MyBase.log ”（Macintosh版）または “ MyBase.4DL ”（Windows版）という名前のログファイル名は、次の名前のいずれかになっているはずです。

■ オルジナルデータベースを含んでいるマシン上（ログファイルの入っているフォルダ内で）：

Macintosh	Windows
MyBase.log	MyBase.4DL
MyBase.log.2	MyBase.4L2
ログの送付	Sending.4DL

■ ミラーマシン上（Macintosh上の“MyBase\”フォルダ内およびWindows内の“MyBase.MIR”ディレクトリ内）：

Macintosh	Windows
ログ受信中	Receive.4DL
ログ分析中	Analyze.4L2
ログ統合中	Restore.4DL
ミラーログ	Mirror.4DL

場面１：データベースが両方とも元のままの場合

このケースは、データキャッシュの書き出し時やログファイルの統合時にアクシデントが発生したのではありません。次の表は、各マシン上で使用されるファイル名とそれに対応する操作の状況を表したものです。

しかし、ミラー更新は失敗し、中間ログファイルの存在がデータベースの再起動を妨げるかもしれません。ログファイル統合の状況や各マシン上にあるファイルによっていくつかのケースに分けられます。

ケース1：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log	MyBase.4DL	なし	なし

状況と復旧方法：アクシデントが発生したにもかかわらず、更新は正しく行われます。オリジナルデータベースを再起動する。

ケース2：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log + ログの送付 または MyBase.log.2	MyBase.4DL + Sending.4DL または MyBase.4L2	なし または ログ受信中	なし または Receive.4DL

状況と復旧方法：ログはミラーマシンに送られて（または、送り終えて）いません。

1. **ミラーマシンに"ログ受信中"（Macintosh版）または“ Receive.4DL ”（Windows）が存在している場合は消去する。**
2. **ミラーを起動する。**
3. **オリジナルデータベースを起動する。**
4. **ミラー更新を行う。**

注：データベースマシンに"MyBase.log.2"（Macintosh版）または“ MyBase.4L2（Windows版）ファイルが存在する場合は、ファイル名を"ログの送付"（Macintosh版）または“ Receive.4DL ”（Windows版）に変更します。

ケース3：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log	MyBase.4DL	ログ分析中	Analyze.4DL

状況と復旧方法：ログファイルはミラーに送られたが、統合されていない状態です。

1. **ミラーマシン上で、"ログ分析中"（Macintosh版）または“ Analyze.4DL ”（Windows版）ファイルをミラーデータベースに統合する。**
2. **"ログ分析中"（Macintosh版）または“ Analyze.4DL ”（Windows版）ファイルを消去する。**
3. **ミラーを起動する。**

ケース4：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log	MyBase.4DL	ミラーログ	Mirror.4DL

状況と復旧方法：ミラーデータベースは更新されたが、データベースのフルバックアップが中断された状態です。

1. **ミラーマシン上で、4D Backupを使ってミラーデータベースのバックアップを作成する。**
   "MyBase\"フォルダ（Macintosh版）または“ MyBase.MIR ”ディレクトリ（Windows版）にアーカイブを置きます。
2. **"ミラーログ"（Macintosh版）または“ Mirror.4DL ”（Windows版）ファイルを消去する。**
3. **ミラーマシンを起動する。**

場面 2 ：オリジナルデータベースだけがダメージを受けている場合

このケースでは、ミラーデータベースは元のままのです。データベースマシン上のミラーデータベースの内容が再設定されます。ログファイル統合の状況や各マシン上にあるファイルによっていくつかのケースに分けられます。

ケース1：			
データベースマシン上のファイル		**ミラーマシン上のファイル**	
Macintosh	**Windows**	**Macintosh**	**Windows**
MyBase.log + ログの送付 または MyBase.log.2	MyBase.4DL + Sending.4DL または MyBase.4L2	なし または ログ受信中	なし または Receive.4DL

状況と復旧方法：ログファイルはミラーマシンに送られて（または、送り終えて）いません。

1. **ミラーマシンに"ログ受信中"（Macintosh版）または“ Receive.4DL ”（Windows版）が存在している場合は消去する。**
2. **ミラーデータベースからオリジナルのデータベースマシンにデータをコピーする。**
3. **引き続き、データベースマシンに、Macintosh版では“ ログの送付 ”と“ MyBase.log ”、Windows版では“ Sending.4DL ”と“ MyBase.4L2 ”（Windows版）ファイルを統合する。**
4. **4th Dimensionまたは4D Serverを使ってそのデータベースを開き、カレントログファイルとして“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）を選択する。**
5. **4ミラーマシンを起動する。**

注：データベースマシンに"MyBase.log.2"（Macintosh版）または“ MyBase.4L2（Windows版）ファイルが存在する場合は、ファイル名を"ログの送付"（Macintosh版）または“ Sending.4DL ”（Windows版）に変更します。

ケース2：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log	MyBase.4DL	ログ分析中	Analyze.4DL

状況と復旧方法：ログファイルはミラーに送られたが、統合されていない状態です。

1. **ミラーマシン上で、"ログ分析中"（Macintosh版）または“ Analyze.4DL ”（Windows版）ファイルをミラーデータベースに統合する。**
2. **"ログ分析中"（Macintosh版）または“ Analyze.4DL ”（Windows版）ファイルを消去する。**
3. **ミラーデータベースからオリジナルデータベースにデータをコピーする。**
4. **コピーしたデータベースの中に"ログ分析中"（Macintosh版）または“ Analyze.4DL ”（Windows版）ファイルを統合する。**
5. **4th Dimensionを使ってそのデータベースを開き、カレントログファイルとして“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）を選択する。**
6. **ミラーマシンを起動する。**

ケース3：

データベースマシン上のファイル		ミラーマシン上のファイル	
Macintosh	Windows	Macintosh	Windows
MyBase.log	MyBase.4DL	ミラーログ	Mirror.4DL

状況と復旧方法：ミラーデータベースは更新されたが、データベースのフルバックアップが中断された状態です。

1. **ミラーマシン上で、4D Backupを使ってミラーデータベースのバックアップを作成する**

"MyBase\"フォルダ（Macintosh版）または“ MyBase.MIR ”ディレクトリ（Windows版）にアーカイブを置きます。

2. **"ミラーログ"または（Macintosh版）“ Mirror.4DL ”（Windows版）ファイルを消去する。**
3. **ミラーデータベースをオリジナルデータベースにコピーする。**
4. **オリジナルデータベースの中に"ミラーログ"（Macintosh版）または“ Mirror.4DL ”（Windows版）ファイルを統合する。**

5. 4th Dimensionまたは4D Serverを使ってそのデータベースを開き、カレントログファイルとして“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）を選択する。

6. ミラーマシンを起動する。

場面 3 ：ミラーデータベースだけがダメージを受けている場合

ミラーデータベースだけがダメージを受けている場合はミラーを再インストールします。

1. 使用できなくなったミラーデータベースを削除する。

2. オリジナルデータベースマシンから“ MyBase.log.2 ”（Macintosh版）または“ MyBase.4DL ”（Windows版）ファイルを削除し、“ ログの送付 ”（Macintosh版）または“ Sending.4DL ”（Windows版）ファイルが存在していれば同様に削除する。

3. オルジナルデータベースをミラーデータベースにコピーする。

4. ミラーマシンを起動する。

場面 4 ：データベースが両方ともダメージを受けている場合

こうした状況はまれにしか起こりません。例えば、キャッシュからディスクへの書き出しおよびミラー上のログファイルへの統合と同時に停電が起きたような場合です。

次のようなログファイルが現れます。

- オリジナルデータベースマシン上の“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）ファイル
- ミラーマシン上の“ ログ統合中 ”（Macintosh版）または“ Restore.4DL ”（Windows版）ファイル

データベースの復旧は、ミラーバックアップからのデータベースの復元、およびログファイルの統合によって行います。

1. ミラー上のダメージを受けたデータファイルを削除する。

2. ミラーのデータファイルの最新バックアップからミラーデータベースを復元し、ミラーデータベースと同じフォルダに入れる。

   ミラーバックアップはミラーディスクの最上位階層に配置されている“ MyBase\ ”フォルダ（Macintosh版）または“ MyBase.MIRディレクトリ（Windows版）内に入っています。

3. 復元したデータベースに“ ログ統合中 ”（Macintosh版）または“ Restore.4DL ”（Windows版）ファイルを統合する。

4. “ ログ統合中 ”（Macintosh版）または“ Restore.4DL ”（Windows版）ファイルを削除する。

5. ミラーデータベースからオルジナルデータベースへデータファイルをコピーする。
6. オルジナルデータベースに“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）ファイルを統合する。
7. ミラーデータベースとオルジナルデータベースを起動する。
8. オルジナルデータベースから、カレントログファイルとして“ MyBase.log ”（Macintosh版）または“ MyBase.4DL ”（Windows版）ファイルを選択する。

O

R

S

U