4D バージョン 6.8
ランゲージ追加/修正情報

目次

CREATE DATA FILE

CREATE DATA FILE (アクセスパス)

引数	タイプ		説明
アクセスパス	文字列	←	作成するデータベースの名前、 または完全なアクセスパス

説明

CREATE DATA FILEコマンドは、オンザフライで新しいデータファイルをディスク上に作成し、4Dアプリケーションで開かれているデータファイルと置き換えます。

このコマンドの全般的な動作は、**OPEN DATA FILE**コマンドと同じです。唯一の相違点は、ストラクチャファイルを再オープンした後に、引数＜アクセスパス＞で設定された新しいデータファイルを作成するところです。

処理を開始する前に、コマンドは指定されたアクセスパスが既存のファイルに該当していないかどうかを調べます。

4D Server：4D Clientや4D Serverではこのコマンドは使用できません。

OPEN DATA FILE

OPEN DATA FILE (アクセスパス)

引数	タイプ		説明
アクセスパス	文字列	←	開こうとするデータベースの名前、または完全なアクセスパス

説明

OPEN DATA FILEコマンドは、4Dアプリケーションによって開かれたデータファイルをオンザフライで変更します。

引数＜アクセスパス＞には、開こうとするデータファイルの名前、または完全なアクセスパスを渡します。ファイル名だけを渡す場合、データファイルはデータベースのストラクチャファイルと同じ階層に配置されていなければなりません。

アクセスパスが有効なデータファイルを指している場合、4Dは現在使用しているデータベースを終了した後、指定されたデータファイルを使って再度オープンします。データベースメソッドの「On Exit」と「On Startup」が続けて呼び出されます。

このコマンドは非同期的に実行されます。つまり、コマンド呼び出しの後、4Dはメソッドの残りの部分を続けて実行します。この後、アプリケーションは「ファイル」メニューの「終了」コマンドが選択された場合と同様の処理を行います。表示されているダイアログボックスはキャンセルされ、実行中のプロセスは10秒間の猶予の後に打ち切られます。

処理を開始する前に、コマンドは指定されたデータファイルが有効かどうかを検証します。つまり、ファイルの拡張子がWindowsでは“.4dd”、MacOSではファイルタイプが“dat5”でなくてはなりません。また、そのファイルが既にオープンされている場合、コマンドはファイルが現在使用中のストラクチャファイルに対応するかどうかを確認します。

＜アクセスパス＞に空の文字列を渡した場合、このコマンドはデータファイルを変更せずにデータベースを再度開きます。

4D Server：4D Clientや4D Serverではこのコマンドは使用できません。

CALL PROCESS

CALL PROCESS (プロセス)

引数	タイプ		説明
プロセス	数値	→	プロセス番号

説明

CALL PROCESS コマンドは、＜プロセス＞の最前面に表示されたフォームを呼び出します。

重要：**CALL PROCESS** コマンドは、同一マシン上で実行されたプロセス間でのみ有効です。

存在しないプロセスを呼び出した場合には、何も行いません。

＜プロセス＞（目的のプロセス）で現在フォームが表示されていない場合には何も行いません。目的のプロセスで表示されたフォームがOn Outside call イベントを受け取ります。「デザイン」モードの「フォームプロパティ」ウインドウにおいて、このフォームに対してOn Outside call イベントを必ず有効にし、フォームメソッドでこのイベントを管理する必要があります。このイベントが無効であったり、またはフォームメソッドでイベントの管理を行わない場合、何も行われません。

注：On Outside call イベントは、受け取り側である入力フォームの入力状況を変更します。特に、フィールドが編集中である場合には、On Data change イベントが生成されます。

呼び出し元プロセス（**CALL PROCESS** コマンドが実行されたプロセス）は“待機”しません。つまり、**CALL PROCESS** コマンドは即座に効力を持ちます。必要であれば、2つのプロセス間で読み書きを行えるように（**GET PROCESS VARIABLE** および **SET PROCESS VARIABLE** コマンドにより）インタープロセス変数やプロセス変数（この目的のために用意）を用いて、呼び出し元プロセスから応答のための待機用ループを作成する必要があります。

フォームを表示しないプロセスの間で通信を行うには、**GET PROCESS VARIABLE** および **SET PROCESS VARIABLE** コマンドを使用してください。

CALL PROCESS コマンドには **CALL PROCESS** (-1)というもう一つの構文があります。

メソッドの実行速度が遅くならないように、4th Dimensionはインタープロセス変数が変更されるたびに再描画することはしません。もし、プロセス参照番号の代わりに“-1”を**CALL PROCESS**コマンドの引数<プロセス>に渡すと、4th Dimensionはプロセスを1つも呼び出さず、その代わりに、同一マシン上で実行されているプロセス内のすべてのウインドウに表示されるインタプロセス変数をすべて更新します。4D Serverでは、クライアントから実行される**CALL PROCESS** (-1)は、そのクライアント上で実行されているプロセスのインタープロセス変数のみを更新します。

C_INTEGER

C_INTEGER ({メソッド ;}変数 {; 変数２ ; …; 変数 N})

引数	タイプ		説明
メソッド	メソッド	→	メソッドの名前（オプション）
変数	変数または${...}	→	定義する変数の名前

注：このコマンドは、以前のデータベースとの互換性を保つために 4th Dimension に残されています。実際には、4D および 4D Compiler アプリケーションは、内部的に整数を倍調整数へとタイプ変換します。

例えば

C_INTEGER($MyVar)
$TheType:=Type($MyVar) `$TheType = 9 (Is Longint)

説明

C_INTEGER コマンドは、指定されたそれぞれの変数を整数タイプの変数としてキャストします。

コマンドの第 1 の形式は、オプション引数＜メソッド＞が渡されない形式であり、プロセス変数、インタープロセス変数、ローカル変数の宣言とタイプ定義に使用されます。

注：この形式は、インタープリタを使用したデータベースで使用できます。

コマンドの第 2 の形式は、オプション引数＜メソッド＞が渡される形式であり、メソッドの結果またはパラメータ（$0、$1、$2 等）またはその両方を 4D Compiler 用に事前に定義するために使用されます。このコマンドの形式は、データベースのコンパイル中に、変数設定フェーズをスキップし、コンパイル時間を節約するために使用します。

警告：2 番目の形式はインタープリタモードでは実行できません。このため、このシンタックスは、インタープリタモードでは実行されないメソッドでだけ使用するようにしてください。このメソッドの名前は「COMPILER」で開始する必要があります。

上級ヒント：シンタックス「C_INTEGER (${...})」を使用すると、パラメータ群がメソッドの最後のパラメータ群である場合に、不定数のパラメータを同一タイプとして宣言できます。例えば、「C_INTEGER (${5})」宣言は、4D と 4D Compiler に対して、5 番目のパラメータから始め、そのメソッドがそのタイプの不定数のパラメータを受け付けられることを示しています。

Euro converter

Euro converter (値;元の通貨;変換通貨)　→　実数

引数	タイプ		説明
値	実数	→	値を変換
元の通貨	文字列	→	表示されている値の通貨コード
変換通貨	文字列	→	変換したい値の通貨コード
戻り値	実数	←	変換した値

説明

Euro converter 関数は"ユーロ"に所属するユーロ通貨の元と先の異なった通貨の値を変換します。

変換できるものは

- ユーロから国際通貨
- 国際通貨からユーロ
- 他の国際通貨から国際通貨。この場合、ヨーロッパ組織の特質として、変換はユーロの仲介によって計算されます。

例えば、ベルギーフランをドイツマルクに変換すると、4D は以下の計算を実行します。

ベルギーフラン－＞ユーロ－＞ドイツマルク

最初の引数を変換する値とします。

2番目の引数は表示されている値の通貨コードを示します。

3番目の引数は変換したい値の通貨コードを示します。

特に通貨コードを、4th Dimension は以下のあらかじめ定義された定数を提供します。

定数	タイプ	値
Austrian Schilling	文字列	ATS
Belgian Franc	文字列	BEF
Deutsche mark	文字列	DEM
Euro	文字列	EUR
Finnish Markka	文字列	FIM
French Franc	文字列	FRF
Greek drachma	文字列	GRD
Irish Pound	文字列	IEP
Italian Lire	文字列	ITL
Luxembourg Franc	文字列	LUF
Netherlands Guilder	文字列	NLG

Portuguese Escudo	文字列	PTE
Spanish Peseta	文字列	ESP

必要であれば、4th Dimensionは変換した結果を小数点2位をもって自動的に四捨五入します。例外としてイタリアリラ、ベルギーフラン、ルクセンブルグフラン、スペインペセタ等の変換に4Dは小数点0位を四捨五入します（結果は整数の数値です）。

修正されたユーロと12の参加メンバー国の通貨の変換レートです。

通貨	1ユーロの値
Austrian Schilling	13.7603
Belgian Franc	40.3399
Deutsche mark	1.95583
Finnish Markka	5.94573
French Franc	6.55957
Greek drachma	340.750
Irish Pound	0.787564
Italian Lire	1936.27
Luxembourg Franc	40.3399
Netherlands Guilder	2.20371
Portuguese Escudo	200.482
Spanish Peseta	166.386

例題

以下の例題はこの関数を使用して変換したものです。

```
$value:=10000 `フランスフランで値を表示
                `ユーロの値に変換
$InEuros:=Euro converter($value;French Franc; Euro)
`イタリアリラの値へ変換
$InLires:=Euro converter ($value;French Franc; Italian Lira)
```

Form event

Form event → 数値

引数	タイプ		説明
このコマンドには、引数はありません。			
戻り値	数値	←	フォームイベント番号

説明

Form event関数は、発生したばかりのフォームイベントのタイプを示す数値を返します。通常、ユーザはフォームやオブジェクトメソッド内から**Form event**関数を使用します。

4th Dimensionは、前もって定義された以下のような定数を持っています。

定数	値	説明
On Load	1	フォームは表示または印刷されようとしています。
On Unload	24	フォームは終了または解放されようとしています。
On Validate	3	レコードのデータ入力が有効となりました。
On Clicked	4	オブジェクトがクリックされました。
On Double Clicked	13	オブジェクトがダブルクリックされました。
On Before Keystroke	17	フォーカスを持つオブジェクト内に文字が入力されようとしています。イベント内で**GET edited text**関数を実行しようとするとオブジェクト内の入力前のテキストを返します。**【Get edited text returns the object's text wihtout this character】**
On After Keystroke	28	フォーカスを持つオブジェクト内に文字が入力され ています。イベント内で**GET edited text**関数を実行すると、最後に入力された文字をエリアの内容を返します。**【Get edited text returns the object's text including this character】**
On Getting Focus	15	フォームオブジェクトがフォーカスを獲得します。
On Losing Focus	14	フォームオブジェクトがフォーカスを失います。
On Activate	11	フォームのウインドウが最前列のウインドウになります。
On Deactivate	12	フォームのウインドウはもはや最前列のウインドウではありません。
On Outside Call	10	フォームが**CALL PROCESS**コマンド呼び出しを受け取りました。
On Drop	16	データがオブジェクト上にドロップされました。
On Drag Over	21	データがオブジェクト上にドロップされる可能性があります。
On Menu Selected	18	メニュー項目が選択されました。
On Data Change	20	オブジェクトデータが変更されました。

On Plug in Area	19	プラグインエリアが自身の実施すべきオブジェクトメソッドを要求しました。
On Header	5	フォームのヘッダエリアが印刷／表示されようとしています。
On Printing Detail	23	フォームのディテイルエリアが印刷されようとしています。
On Printing Break	6	フォームのブレークエリアの1つが印刷されようとしています。
On Printing Footer	7	フォームのフッタエリアが印刷されようとしています。
On Close Box	22	ウインドウのクローズボックスがクリックされました。
On Display Detail	8	レコードがリスト内に表示されようとしています。
On Open Detail	25	レコードがダブルクリックされ、ユーザは入力フォームに移ります。
On Close Detail	26	ユーザは入力フォームから出力フォームへ戻ります。
On Timer	27	**SET TIMER** コマンドで定義された Tick 数が経過しました。
On Resize	29	フォームウインドウのサイズが変更されました。

イベントとメソッド

フォームイベントが発生すると、4th Dimension は以下のようなアクション（動作）を実行します。

- まず、4th Dimension は、フォームの各オブジェクトをブラウズし、対応するオブジェクトイベントプロパティが選択されているオブジェクトのオブジェクトメソッドを呼び出します。
- 次に、対応するフォームイベントプロパティが選択されている場合、フォームメソッドを呼び出します。

オブジェクトメソッドは（それが存在する場合）一定の順序で呼び出されることはありません。一般的な法則としては、常にオブジェクトメソッドはフォームメソッドより先に呼び出されるというだけです。

オブジェクトがサブフォームである場合、サブフォームのリストフォームのオブジェクトメソッドが呼び出され、それからリストフォームのフォームメソッドが呼び出されます。この後、4D は親フォームのオブジェクトメソッドを呼び出します。つまり、オブジェクトがサブフォームである場合、4D はサブフォームオブジェクト内のオブジェクトとフォームメソッドに対して同じ一般法則を適用します。

On Load イベントと On Unload イベントを除き、発生したイベントに対応するイベントプロパティがフォームプロパティ上で選択されていない場合でも、同じイベントプロパティが選択されているオブジェクトのオブジェクトメソッドは呼び出されます。つまり、フォームレベルでイベントを有効あるいは無効にしても、オブジェクトのイベントプロパティに影響を与えないということです。

ひとつのオブジェクト内に含まれるイベント数は、イベントの性質によって異なります。

■ On Loadイベント - On Loadオブジェクトイベントプロパティが選択されているフォームの（すべてのページの）オブジェクトは、そのオブジェクトメソッドが呼び出されます。さらに、On Loadフォームイベントプロパティが選択されている場合には、そのフォームのフォームメソッドが呼び出されます。

■ On ActivateまたはOn Resizeイベント - オブジェクトメソッドは呼び出されません。その理由は、このイベントが特定のオブジェクトに適用されるのではなく、フォーム全体に適用されるからです。したがって、On Activateフォームイベントプロパティが選択されている場合、フォームメソッドだけが呼び出されます。

■ On Drag Overイベント - On Drag Overオブジェクトイベントプロパティが選択されている場合、イベント内に含まれるドロップ可能なオブジェクトのオブジェクトメソッドだけが呼び出されます。フォームメソッドは呼び出されません。

■ On Open DetailおよびOn Close Detailイベント－これらのイベントは、**DISPLAY SELECTION**、**MODIFY SELECTION**コマンドによってOUTPUTフォームが表示している状態から、特定のレコードの表示、修正を行うためにINPUTフォームとの間の移動を行った時にだけ発生します。これらのイベントの取得は出力フォームのフォームメソッドで行う必要があります。

■ On Timerイベント - このイベントは、フォームメソッドで前もって**SET TIMER**コマンド呼び出しが実行されている場合にのみ発生します。On Timerフォームイベントプロパティが選択されている場合、フォームメソッドだけがこのイベントを受け取り、オブジェクトメソッドは呼び出されません。

警告：On Drag Overイベント中は、他のイベントとは異なり、オブジェクトメソッドは、ドラッグ&ドロップされるオブジェクトのプロセスコンテキスト内で実行され、ドラッグ&ドロップ送信先オブジェクトのプロセスコンテキスト内では実行されません。

以下の表は、オブジェクトメソッドとフォームメソッドを各イベントタイプごとに呼び出す方法を要約したものです。

定数	オブジェクトメソッド	フォームメソッド	対象オブジェクト
On Load	はい	はい	全オブジェクト
On Unload	はい	はい	全オブジェクト
On Validate	はい	はい	全オブジェクト
On Clicked	はい (クリック可能な場合)	はい	関連するオブジェクトのみ
On Double Clicked	はい (クリック可能な場合)	はい	関連するオブジェクトのみ
On Before Keystroke	はい (キーボード入力可能な場合)	はい	関連するオブジェクトのみ
On After Keystroke	はい (キーボード入力可能な場合)	はい	関連するオブジェクトのみ
On Getting Focus	はい (タブ可能な場合)	はい	関連するオブジェクトのみ
On Losing Focus	はい (タブ可能な場合)	はい	関連するオブジェクトのみ
On Activate	なし	はい	なし
On Deactivate	なし	はい	なし
On Outside Call	なし	はい	なし
On Drop	はい (ドロップ可能な場合)	はい	関連するオブジェクトのみ
On Drag Over	はい (ドロップ可能な場合)	なし	関連するオブジェクトのみ
On Menu Selected	なし	はい	なし
On Data Change	はい (変更可能な場合)	はい	関連するオブジェクトのみ
On Plug in Area	はい	はい	関連するオブジェクトのみ
On Header	はい	はい	全オブジェクト
On Printing Details	はい	はい	全オブジェクト
On Printing Break	はい	はい	全オブジェクト
On Printing Footer	はい	はい	全オブジェクト
On Close Box	なし	はい	なし
On Display Details	はい	はい	全オブジェクト
On Open Details	なし	はい	なし
On Close Details	なし	はい	なし
On Resize	なし	はい	なし
On Timer	なし	はい	なし

重要：あらゆるイベントに対して、対応するイベントのプロパティがそのフォームやオブジェクト用に選択されている場合は、そのフォームやオブジェクトのメソッドが呼び出されることを覚えていてください。「デザイン」モードで（「フォームプロパティ」および「オブジェクトプロパティ」ウインドウを使用して）イベントを無効にすると、メソッドを呼び出す回数をかなり縮小でき、これによりフォームの実行速度を著しく最適化することができるというメリットがあります。

警告：オブジェクトとそれが属するフォームの両方に対してOn LoadとOn Unloadイベントイベントが有効である場合、オブジェクトに対してOn LoadとOn Unloadイベントが生成されます。これらのイベントがオブジェクトに対してのみ有効となる場合、これは起こりません。この2つのイベントはフォームレベルでも有効でなくてはなりません。

Get 4D folder

Get 4D folder→文字列

引数	タイプ		説明
このコマンドには、引数はありません。			
戻り値	文字列	←	4Dフォルダのパス名

説明

Get 4D folder関数は、アクティブな4Dフォルダへのパス名を返します。4D環境はこの4Dフォルダを使用して以下の情報を保存します。

- ユーザレジストレーションファイル
- 4D環境アプリケーション、ツール、ユーティリティプログラムで使用される初期設定（Preference）ファイル
- TCP／IPネットワークプロトコルのオプションファイル
- 4D Serverからダウンロードされたリソースを格納するために4D Clientによって作成された「.rex」ファイルと「.res」ファイル
- 4D Serverからダウンロードされた4D Extensionsを格納するために4D Clientによって作成されたローカルデータベース

また、独自のオンラインヘルプファイルや、設定ファイル等を保存するために4Dフォルダを使用することもできます。4Dフォルダへの実際のパス名を取得するために**Get 4D folder**関数を使用することにより、あるローカライズされたシステムで動作している任意のプラットフォーム上での機能を保証します。

警告：4Dフォルダの中には、任意のファイルまたはドキュメントを自由に格納できますが、4D環境自身によって作成されたファイルの移動および変更は避けた方が賢明です。

4D 6.8以降、4Dフォルダは以下の場所に作成されます

- On Windows NT 4::
  {Disk}:¥{System folder}¥Profiles¥All Users¥Application Data¥4D
- On Windows 98 and Windows Millenium::
  {Disk}:¥{System folder}¥All users¥Application Data¥4D
- On Windows 2000 and Windows XP::
  {Disk}:¥Documents and Settings¥All Users¥Application Data¥4D
- On MacOS 9::
  {Disk}:System folder:Application Support:4D
- On MacOS X::
  {Disk}:Library:Application Support:4D

互換性に関する注意：4D 6.8では、4Dフォルダの保存場所が変更されました。前バージョンの4Dにおいてこのフォルダは、Windowsでは Windowsシステムファイルフォルダ内に、MacOSでは「システムフォルダ:初期設定」フォルダ内に配置されていました。同一コンピュータ上で前バージョンの4Dが使用されていた場合、4D 6.8アプリケーションは以下の場所に4Dフォルダを探します。

1. 新しい場所（前述）：存在する場合にはそのフォルダが使用されます。見つからない場合、4Dはステップ2へ進みます。

2. 以前の場所（システム）。存在する場合にはそのフォルダが使用されます。見つからない場合、4Dはステップ3へ進みます。

3. 新しい場所に4Dフォルダを作成します。

▼ 以下の例は、シングルユーザデータベースの起動中に、4Dフォルダの中に配置されたファイル内にユーザ自身が設定した内容をロードしたいとします。これを実行するには、「On Startup」データベースメソッド内に以下のようなコードを記述します。

```
MAP FILE TYPES ("PREF" ; "PRF" ; "初期設定ファイル")
`「PREF」Macintoshファイルタイプを「PRF」Windowsファイル拡張子にマップする
˝ $vsPrefDocName:=Get 4D folder+"MyPrefs" `初期設定ファイルにパス名を作成する
` そのファイルが存在するかチェックする
If (Test path name ($vsPrefDocName+(".PRF"*Num(On Windows)))#Is a document)
   $vtPrefDocRef:=Create document($vsPrefDocName ; "PREF") ` ファイルを作成する
Else
   $vtPrefDocRef:=Open document($vsPrefDocName ; "PREF") ` ファイルを開く
End if
If (OK=1)
   `ドキュメント内容を処理する
   CLOSE DOCUMENT($vtPrefDocRef)
Else
   `エラーの取り扱い
End if
```

LIST TO BLOB

LIST TO BLOB (変数 ; blob {; *})

引数	タイプ		説明
変数	変数	→	BLOBの中に格納する階層リスト
blob	BLOB	→	階層リストを受信するBLOB
*	*	→	値を追加する場合は*

説明

LIST TO BLOBコマンドは、BLOB内に階層リストを格納します。

オプション引数＜*＞を指定した場合には、階層リストはBLOBに追加され、これに合わせてBLOBのサイズも拡張されます。オプション引数＜*＞を使用すれば、BLOBがメモリ容量内であれば変数やリストをいくつでも順番にBLOBの中に格納することができます。

オプション引数＜*＞を指定しない場合には、階層リストはBLOBの最初に格納され、それ以前にそこにあった内容を上書きします。これに合わせてBLOBのサイズも調整されます。

既存のBLOB内容に追加するか、上書きするかによらず、BLOBは必要なサイズに調整されます。実際に値をセットしたバイト以外は0（ZERO）で埋められます。

警告：BLOBを使用して階層リストを格納すると、階層リストは4D内部形式を使用してBLOBに格納されるため、格納されたBLOBの内容を読み出すには BLOB to list関数を使用しなければなりません。

呼び出し後、階層リストが正常に格納された場合には、システム変数OKは1に設定されます。階層リストを格納するために必要なメモリがない、等の理由で処理が実行できなかった場合には、システム変数OKは0に設定されます。

プラットフォームからの独立性に関する注意点

LIST TO BLOBコマンドおよび **BLOB to list**関数は、4D内部形式を使用してBLOB内に格納された変数を処理します。このメリットとして、これら2つのコマンドを使用していればプラットフォーム間でのバイトのスワップに配慮する必要がありません。つまり、これらのコマンドのうちいずれかを使用してWindows上で作成されたBLOBはMacintosh上でも使用でき、その逆も可能です。

LOAD COMPRESS PICTURE FROM FILE

LOAD COMPRESS PICTURE FROM FILE (ドキュメント参照番号 ; 方法 ; 品質 ; ピクチャ)

引数	タイプ		説明
ドキュメント参照番号	Docref	→	ドキュメントファイル参照番号
方法	文字列	→	圧縮方法（4バイト）
品質	整数	→	圧縮の品質（1～1000）
ピクチャ	ピクチャ	←	圧縮するピクチャ

説明

このコマンドは、ディスク上のドキュメントからロードされたピクチャを圧縮します。

Open document 関数を使って、PICT ドキュメントを開くことができます。次に、この関数から返されるドキュメントファイル参照番号を使い、ドキュメント中の PICT データをロードして圧縮します。このコマンドはピクチャをメモリにロードし、指定した方法と品質に従って圧縮し、それを引数＜ピクチャ＞に返します。

ピクチャは圧縮される前にメモリへロードされます。ピクチャをロードするためのメモリが足りない場合は、**LOAD COMPRESS PICTURE FROM FILE** コマンドを呼び出す前に **COMPRESS PICTURE FILE** コマンドを使ってください。

引数＜方法＞は、圧縮タイプを 4 バイトの文字列で指定します。引数＜品質＞は、圧縮されたピクチャの品質を 1 から 1000 までの整数で指定します。一般に、品質を下げると、ピクチャの圧縮率が増加します。

警告：指定された品質に対して可能な圧縮率は、圧縮するピクチャのサイズと種類によって異なります。小さなピクチャを圧縮すると、サイズが減少しないこともあります。

例題

以下の例は、「ファイルを開く」ダイアログボックスを表示し、ユーザはそこで PICT ファイルを選択します。PICT ファイルのピクチャがメモリにロードされて圧縮され、ピクチャ変数に格納されます。この後、ファイルが閉じられます。

```
vRef:=Open document ("";"PICT")
If (OK=1)
   LOAD COMPRESS PICTURE FROM FILE (vRef;"jpeg";500;Picture)
   CLOSE DOCUMENT (vRef)
End if
```

Open document

Open document (ドキュメント {; ファイルタイプ}) → ドキュメントファイル参照番号

引数	タイプ		説明
ドキュメント	文字列	→	ドキュメントファイル名、または ドキュメントへの完全なパス名、または 空の文字列の場合、標準のファイル ダイアログボックス表示
ファイルタイプ	文字列	→	Macintoshファイルタイプ（4桁の文字）、 またはWindowsファイル拡張子（1～3 桁の文字）、または省略した場合、テキ ストドキュメント（.TXT）
モード	整数	→	ドキュメントを開くモード
戻り値	DocRef	←	ドキュメントファイル参照番号

説明

Open document関数は、＜ドキュメント＞に指定したドキュメント名またはパス名を持つドキュメントファイルを開きます。

＜ドキュメント＞に空の文字列（ヌル""）を指定した場合には、Windows版では「開く」（Macintosh版では、「ファイルを開く」）ダイアログボックスが表示されます。このダイアログをキャンセルすると、ドキュメントファイルは開かれず、**Open document**関数はドキュメントファイル参照番号にヌル値を返し、システム変数OKに0を代入します。

- ドキュメントが正しく開かれると、**Open document**関数はドキュメントファイル参照番号を返し、システム変数OKに1を代入します。
- ドキュメントが既に開かれており、引数＜モード＞が省略されている場合には、**Open document**関数はドキュメントをReadモードで開き、システム変数OKに1を代入します。
- ドキュメントが既に開かれており、Writeモードでそのドキュメントを開こうとした場合、エラーが発生します。
- ドキュメントが存在しなかったり、既に開かれている場合には、エラーが発生します。

Macintoshの場合、「ファイルオープン」ダイアログボックスにはデフォルトとしてすべてのドキュメントファイルが表示されます。別のタイプのドキュメントファイルを表示するには、オプション引数＜ファイルタイプ＞にドキュメントファイルのタイプを指定します。

Windowsの場合、「開く」ダイアログボックスにはデフォルトとしてすべてのドキュメントファイルタイプ（*.*）が表示されます。別のタイプのドキュメントファイルを表示するには、オプション引数＜ファイルタイプ＞に1から3文字のWindowsのファイル拡張子、または**MAP FILE TYPES**コマンドを使ってマップされるMacintoshのファイルタイプを指定します。

Windowsの場合、「開く」ダイアログボックスを使用しない場合でも、オプション引数＜ファイルタイプ＞に開こうとするドキュメントファイルのファイル拡張子を指定することがあります。デフォルトでは、**Open document**関数は.TXTという拡張子の付いたテキストドキュメントを開こうとします。＜ファイルタイプ＞を指定すると、**Open document**関数は“ドキュメント.ファイルタイプ”という名前のドキュメントファイルを開こうとします。

例えば、

vhDocRef:=**Open document** ("C:\Letter";"WRI")
という命令を実行すると、ディスク上の“C:¥Letter.WRI”というドキュメントファイルが開かれます。＜ファイルタイプ＞に3桁以上の文字を指定すると、**Open document**関数は最初の3桁だけを参照します。ドキュメントファイルのタイプを指定しないと、**Open document**関数はファイル拡張子のないドキュメントファイルを開こうとし、ファイルが見つからない場合、.TXTという拡張子を持つファイルを開こうとします。このファイルが見つからない場合、「ファイルが見つかりません。」というエラーを返します。

ドキュメントファイルが開かれると、**Open document**関数ははじめにファイルの位置をドキュメントの最初に設定しますが、**Append document**関数はドキュメントの終わりに設定します。

ドキュメントファイルを開いたら、**RECEIVE PACKET**や**SEND PACKET**コマンドを使用してドキュメントへの読み込みや書き込みを行うことができます。また、これらのコマンドと**Get document position**や**SET DOCUMENT POSITION**コマンドを組み合わせて、ドキュメントの一部に直接アクセスすることもできます。

オプションの引数モードは、ドキュメントがどのように開かれるかを指定することができるようにするものです。4つのオープンモードが指定可能です。4th Dimensionには下記の定数が定義されています。

定数	タイプ	値
Read and Write (デフォルト値)	整数	0
Write Mode	整数	1
Read Mode	整数	2
Get Pathname	整数	3

最後に、開かれたドキュメントファイルに対して**CLOSE DOCUMENT**を呼び出すことを忘れないようにしてください。

▼以下の例は、ドキュメントファイル“ノート”を開き、それに“さようなら”という文字列を書き込み、ドキュメントファイルを閉じます。これで、前の内容の“こんにちは”は上書きされるため残りません。

```
C_TIME (vDoc)
vドキュメント:=Open document ("ノート")      `新しいドキュメントファイル"ノート"を開く
If (OK=1)
   SEND PACKET (vドキュメント ; "さようなら")   ` ドキュメントファイルに書き込む
   CLOSE DOCUMENT (vドキュメント)               ` ドキュメントファイルを閉じる
End if
```

▼**Write Mode**で既に開かれているファイルをリードすることができます。

```
vDoc:=Open document ("PassFile";"TEXT") `ファイルのオープンファイルが閉じられる
                                        `前に、読み出し専用モードで検査すること
                                        `ができます。
vRef:=Open document ("PassFile";"TEXT";Read Mode)
```

システム変数とセット

Documentが正しく開かれると、システム変数OKは1になります。それ以外は0となります。呼ばれた後、システム変数Documentはdocument名が入れられます。

モードに3を渡すと!!00:00:00!!（ドキュメントの参照無し）を返します。ファイルは開かれませんが、Documentとシステム変数OKは更新されます。

■ システム変数OKは1になります。

■ Documentには、ドキュメントに渡された値に応じて、名前またはドキュメントのフルパスのいずれかがセットされます（ファイル名を渡すとDocumentはこの名前になります。フルパスを渡すとDocumentはこのフルパスになります）。

注：**docName**に設定されたファイルが見付からない場合またはドキュメントに空白を渡した場合は、オープンファイルダイアログボックスが表示されます。これが受け入れられると、Documentシステム変数とシステム変数OKは上記のように更新されます。これが受け入れられないと、システム変数OKは0になります。

QUIT 4D

QUIT 4D{(時間)}

引数	タイプ		説明
時間	数値	→	サーバ終了までの時間

説明

QUIT 4D コマンドは、4th Dimension もしくは 4D Client、4D Server を終了してデスクトップに戻ります。

コマンドが実行されたアプリケーションに応じ、このコマンドは異なる処理を行います。

4th Dimension および 4D Client で実行された場合

QUIT 4D コマンドを呼び出すと、カレントプロセスはそれ自身の実行を中止して、4th Dimension は次の動作を行います。

- 「On Exit」データベースメソッドがある場合には、4D は新しく作成されたローカルプロセス内でこのメソッドの実行を開始します。したがって、このデータベースメソッドを使用して、プロセス間通信を通じて、（データ入力を）終了、または処理の実行を中止しなければならないことを、他のプロセスに通知することができます。4D は、いずれ終了するということに注意が必要です。「On Exit」データベースメソッドは、必要なクリーンアップや終了の処理を実行することができますが、中断処理を拒否することができず、ある時点で終了することになります。
- 「On Exit」データベースメソッドがない場合には、4D は実行プロセスそれぞれを区別せずに 1 つずつアボートします。ユーザがデータ入力を実行している場合には、レコードはキャンセルされ、保存されません。

現在開いているウインドウ上で行われたデータ入力の変更内容をユーザに保存させたい場合は、プロセス間通信を使って、データベースが終了されようとしていることを他のすべてのユーザプロセスに合図することができます。これを実行するには、以下の 2 つの方法があります。

- **QUIT 4D** コマンドを呼び出す前に、カレントプロセスの中から上記の操作を実行する
- 「On Exit」データベースメソッドの中から上記の操作を取り扱う。

3 番目の方法もあります。**QUIT 4D** コマンドを呼び出す前に、ウインドウが妥当性検査を必要とするかどうかをチェックします。もし、このケースの場合は、ユーザにこのウインドウを有効にするか、または取り消すかを尋ね、それから再度、「終了」を選択するか尋ねます。しかし、ユーザインタフェースの観点から見ると、最初の 2 つの方法を使用するのをお勧めします。

注：4th Dimension や 4D Client では、引数＜時間＞は使用することができません。

4D Server（ストアドプロシージャ）で実行された場合

QUIT 4Dコマンドは、サーバマシン上のストアドプロシージャで実行できるようになりました。その場合、オプションのパラメータ＜時間＞を受け付けます。

引数＜時間＞により、アプリケーションが実際に終了するまでの4D Serverのタイムアウトを設定し、クライアントマシンが接続を切るための時間を与えることができます。＜時間＞には分単位の値を渡します。この引数は、サーバマシン上で実行された際にのみ考慮されます。4D Clientや4th Dimensionでは、この引数は無視されます。

引数＜時間＞を渡さない場合、終了する前に4D Serverはすべてのクライアントマシンが接続を切るまで待機します。

4D Clientや4th Dimensionとは異なり、4D Serverによる**QUIT 4D**の処理は非同期的に行われます。つまり、このコマンドの実行後、コマンドの呼び出しを行ったメソッドが中断されることはありません。

「On Server Shutdown」データベースメソッドが存在する場合、指定した引数に応じて、引数＜時間＞で設定した遅延時間後、またはすべてのクライアント接続が切断された後にメソッドが実行されます。

ストアドプロシージャ内で使用された場合の**QUIT 4D**コマンドの働きは、4D Serverの「ファイル」メニューから「終了」コマンドを選択した場合と同じです。すなわち、各クライアントマシンにダイアログボックスを表示して、サーバが終了することを通知します。

▼ 以下の例は、「ファイル」メニューに「終了」メニューを持ったプロジェクトメソッドを示しています。

```
`「M_終了」プロジェクトメソッド
CONFIRM ("終了してもよろしいですか？")
If (OK=1)
   QUIT 4D
End if
```

RECEIVE PACKET

RECEIVE PACKET ({ドキュメントファイル参照番号 ;} 受信変数 ; 文字数 ¦ 終了文字)

引数	タイプ		説明
ドキュメントファイル	時間 参照番号	→	ドキュメントファイルの参照番号 またはカレントチャンネル（シリアルポートまたはドキュメントファイル）
受信変数	変数	→	受信データを格納する変数
文字数または 終了文字	数値 文字列	→	受信する文字数（バイト） 受信を終了する文字または記号

説明

RECEIVE PACKET コマンドは、シリアルポートまたはドキュメントファイルからデータを読み込みます。

＜ドキュメントファイル参照番号＞を指定した場合には、**Open document**、**Create document**、**Append document** で開かれたドキュメントファイルからデータを読み込みます。＜ドキュメントファイル参照番号＞を指定しない場合は、**SET CHANNEL** コマンドで開かれたシリアルポートかドキュメントファイルからデータを読み込みます。

ソースに関わらず、テキストか文字列である読み込まれた文字は、受信変数として返されます。特定の文字数まで読み込むためには、＜文字数＞にその数を渡します。特定の文字列（1桁以上の文字で構成される）が現われるまで文字を読み込むには、＜終了文字＞にその文字列を渡します（この文字列は＜受信変数＞に返されません）。

ドキュメントを読み込む時に、＜文字数＞または＜終了文字＞が指定されていない場合、**RECEIVE PACKET** コマンドはドキュメントの最後で読み込みを終了します。しかし、文字列変数は固定長ですが、テキスト変数は32000バイトまでのデータを格納できる点に注意してください。シリアルポートから読み込む際、**RECEIVE PACKET** コマンドはタイムアウトになるまで（指定されている場合）、あるいはユーザによって受信が中断されるまで待ち続けます。

RECEIVE PACKET の実行中に、「Ctrl ＋ Alt ＋ Shift」キー（Windows）または「command ＋ option ＋ shift」キー（Macintosh）を押して、受信を中断することができます。中断することにより、エラー-9994が発生します。**ON ERR CALL** を使用してインストールしたエラー処理メソッドにより、このエラーを検出することができます。通常、シリアルポート経由での通信の場合にのみ、受信の中断処理を実行する必要があります。

ドキュメントファイルを読み込む場合、**RECEIVE PACKET** コマンドは、ドキュメントファイルの先頭から読み込みを開始します。その後のデータ読み込みは、最後に読み込まれた文字の次から開始します。

バージョン6における注意：このコマンドは、**SET CHANNEL**を用いて開かれたドキュメントファイルに対して有効です。しかし一方で、**Open document**、**Create document**、**Append document**で開かれたドキュメントファイルに関しては、ドキュメントファイル内での以下の書き込み位置（**SEND PACKET**）や読み込み位置（**RECEIVE PACKET**）を取得、または変更するためには、新しいコマンドである**Get document position**および**SET DOCUMENT POSITION**を使用することができます。

ファイルの最後を越えて読み込もうとした場合は、**RECEIVE PACKET**コマンドは、そのポイントまでに読み込んだデータを返し、システム変数OKに1を代入します。以下の**RECEIVE PACKET**コマンドは空の文字列を返し、システム変数OKに0を代入します。

注：Windows上で、**RECEIVE PACKET**コマンドを使いドキュメントファイルの文字を読込む際に、Windows用の文字をMacintosh用の文字へ変換するためにASCII入力テーブルを使用しない場合には、**Win to Mac**関数を使用することができます。

▼以下の例は、20バイトのデータをシリアルポートから読み込み、変数“読込20”に格納します。

RECEIVE PACKET (読込20 ; 20)

▼以下の例は、変数“ドキュメント”に格納されたドキュメントファイル参照番号の示すドキュメントファイルからデータを読み込み、変数“vデータ”に格納します。ここでは改行（**Char** (13)）を発見するまで読み込みます。

RECEIVE PACKET (ドキュメント ; vデータ ; **Char** (13))

▼以下の例は、変数“myDoc”により参照されたドキュメントからデータを読み込み、変数“vData”に格納します。HTMLタグ“</TD>”（テーブルセルの終わり）が現われるまでデータを読み込みます。

RECEIVE PACKET (myDoc;vData;"</TD>")

▼以下の例は、ドキュメントファイルから読み込んだデータをフィールドに格納します。データは、固定長形式で格納されています。このメソッドは、サブルーチンを呼び出してデータの後ろに付随する不要なスペースを取り除きます。

```
ドキュメント:=Open document ("" ; "TEXT")          ` “テキスト”ファイルを開く
If (OK = 1)                                        ` ユーザがドキュメントファイルを開いたら？
 Repeat                                            ` データがなくなるまで繰り返す
  RECEIVE PACKET (ドキュメント ; $Var1 ; 15)       ` 名字のデータを15バイト読み込む
  RECEIVE PACKET (ドキュメント ; $Var2 ; 15)       ` 名前のデータを15バイト読み込む
  If (OK=1)                                        ` まだドキュメントファイルを最後まで読み込んでない場合
   CREATE RECORD ([従業員])                        ` 新しいレコードを作成
   [従業員]名字:=Strip ($Var1)                     ` 名字をフィールドに格納
   [従業員]名前:=Strip ($Var2)                     ` 名前をフィールドに格納
   SAVE RECORD ([従業員])                          ` レコードを保存
  End if
 Until (OK=0)
 CLOSE DOCUMENT (ドキュメント)                     ` ドキュメントファイルを閉じる
```

```
End if
```

データに付随する不要なスペースを取り除くためのサブルーチン“Strip”を次に示します。

```
For ($i ; Length ($1) ; 1 ; -1)                ` 文字列の最後からループを開始
  If ($1[[$i]]#" ")                            ` スペース以外の場合...
    $i:=-$i                                    ` ループを強制的に終了
  End if
End for
$0:=Delete string ($1 ; -$i ; Length ($1))     ` スペースを削除
```

システム変数とセット

RECEIVE PACKET コマンドへの呼び出しの後に、パケットをエラーなく読み込むと、システム変数OKに1が代入されます。それ以外の場合、システム変数OKに0が代入されます。

SET DATABASE PARAMETER

SET DATABASE PARAMETER ({テーブル;} セレクタ;値)

引数	タイプ		説明
テーブル	テーブル	→	属性を設定するテーブル 引数が省略されている場合は デフォルトテーブル
セレクタ	倍長整数	→	変更するデータベース属性コード
値	倍長整数	→	属性の値

説明

このコマンドは、カレントプロセス用に4Dデータベース内部の様々な属性を変更することができます。

セレクタは、変更するデータベースの属性コードを指定します。4th Dimensionは「データベース属性」のカテゴリー内に、前もって定義されている下記のような定数があります。

定数	タイプ	値
Seq Order Ratio	倍長整数	1
Seq Access Optimization	倍長整数	2
Seq Distinct Values Ratio	倍長整数	3
Index Compacting	倍長整数	4
Seq Query Select Ratio	倍長整数	5
Minimum Web Process	倍長整数	6
Maximum Web Process	倍長整数	7
Web Conversion	倍長整数	8
Database Cache Size	倍長整数	9
4th Dimension Scheduler	倍長整数	10
4D Server Scheduler	倍長整数	11
4D Client Scheduler	倍長整数	12
4D Server Timeout	倍長整数	13
4D Client Timeout	倍長整数	14
Port ID	倍長整数	15
IP Addres to listen	倍長整数	16
Character set	倍長整数	17
Max Concurrent Web Processes	倍長整数	18

値は、属性の値を指定します。値の内容は変更しようとする属性によって違います。セレクタで指定する可能性のある値を示します。

セレクタ＝1 (Seq Order Ratio)(シーケンシャルソートの実行)
値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下ではソートがシーケンシャルモードで実行されます。この率は、100,000 分の 1 単位で表わされます。デフォルト値は 9,000（=9%）です。

セレクタ＝2 (Seq Access Optimization)(シーケンシャルソートの最適化)
値：0 または 1（0：最適化されず、1：最適化する）

内容：シーケンシャルアクセス（配列のソート、検索、選択）用の最適化モード。最適化モードでは、4D はディスクからの多くのレコードを一度に読もうとしますが、これらをキャッシュ内には置きません。このモードは、キャッシュのサイズが低いためです。デフォルトでは、値は 1 になります（最適化モード）。

セレクタ＝3 (Seq Distinct Values Ratio)(シーケンシャル Distinct Values の実行)
値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下では **DISTINCT VALUES** コマンドがシーケンシャルモードで実行されます。この率は、100,000 分の 1 単位で表わされます。デフォルト値は 0 です。

セレクタ＝4 (Index Compacting)(インデックスの圧縮)
値：0 または 1（0：no、1：yes）

内容：インデックスページ圧縮の可能または不可能。デフォルトでは値は 1（インデックスは必要であればコンパクト化される）です。インデックスページは多くのインデックスやレコードを含むデータベースでは、4D のメモリキャッシュを多量に使います。キャッシュがいっぱいで 4D が空きの追加を必要としていると、キャッシュ内のデータは正しくアンロードしません。データをアンロードする前に空きを増やさないと、プログラムはインデックスページの圧縮に空きスペースができるかチェックします。別の方法は後でデータを再ロードを避けることが可能です。

セレクタ＝5 (Seq Query Select Ratio)(シーケンシャル検索の実行)
値：0→100,000

内容：レコードの（セレクトされたレコードとレコードの合計数の間の）選択率。その率以下では **QUERY SELECTION** コマンドがシーケンシャルモードで実行されます。この率は、100,000 分の 1 単位で表わされます。デフォルト値は 0 です。

セレクタ=6 (Minimum Web Process)(Web プロセスの最小数)
値：0→32,767

内容：非コンテキストモード内に保持する Web プロセスの最少数。デフォルトでは、値は 0 になります（下記参照）。

セレクタ＝7 (Maximum Web Process)(Web プロセスの最大数)
値：0→32,767

内容：非コンテキストモード内に保持する Web プロセスの最大数。デフォルトでは、値は 10 になります。

Webサーバが、非コンテキストモードでプロセスの再利用をするために、4DはWebプロセスを5秒間延滞し、次に起こりうるHTTPリクエストの実行のために待機させます。能力の面で言えば、各問い合わせに新しいプロセスを作成するよりも、この原理はずっと利点の多いものです。Webプロセスが再利用されると、もう一度5秒間延滞させられます。5秒以内に何のリクエストも発生しない場合、Webプロセス数が指定した最小数でなければプロセスはアボートされ、最小数に達した場合は再度延滞させられます。

これらの引数は、リクエストの数やメモリ等に応じて、Webサーバの機能を調整できるようにするものです。

セレクタ＝8 (Web conversion mode)(Web変換モード)
値：0、1，2または3
0＝（デフォルト）ブラウザが対応している場合はHTML 4.0フォーマットに変換し、対応していない場合はHTML 3.2と配列を使います。
1＝6.0.x 変換モード
2＝6.5 変換モード
3＝HTML 4.0フォーマット＋CSS-Pに変換（バージョン6.5.2から）

説明：デフォルトでは、4D Web Server 6.8はCSS1(cacading style sheets)を使って4th Dimensionで表示される4Dフォームと同様のHTMLページを生成します。この特性により、フォームが既存のバージョンの4Dにより作られたデータベースに正しく変換されないことがあります。したがって、フォームを変換モードにしておく必要がある場合があります。

内容：いくつかの場合、4Dの6.0xバージョンで作成した、とくにHTMLページに{mypage.htm}のような参照を含むフォームのWeb変換は4Dの6.5では正確でないかもしれません。この場合は、フームの両立を確実にするため、4Dの6.0xの変換モードを実行できます。このモードは**SET DATABASE PARAMETER**が呼ばれたプロセス（Webコンテキスト）でのみセットできます。6.0xのデータベースのすべてのフォームの両立を確実するか、または、1つのフォームが表示される前に**On Web Connection Database**メソッドを呼ぶことが出来ます。このコマンドはコンテストモードかWebプロセスの外側からも呼び出されますが、効果はないでしょう。

注：セレクタの追加は**Get database　parameter**関数のデーターベースキャッシュサイズ（9）でできます。このセレクタは**SET DATABASE PARAMETER**関数では出来ません。

セレクタ ＝ 10 (4th Dimension Scheduler)
セレクタ ＝ 11 (4D Server Scheduler)
セレクタ ＝ 12 (4D Client Scheduler)
値：これら3つのセレクタに対し、引数＜値＞は16進数、0x00aabbccの形式で表わされます。詳細は次の通りです。

aa = システムへのコール毎の最小tick数（0 ～ 100）

bb = システムへのコール毎の最大tick数（0 ～ 100）

cc = システムへのコール間のtick数（0 ～ 20）

これらの値のうち1つが範囲外であれば、4Dによって最大数に設定されます。引数＜値＞には、次の定義済標準値のうちいずれかを渡すことができます。

値 = -1：4Dに割り当てられた最高優先度

値 = -2：4Dに割り当てられた平均優先度

値 = -3：4Dに割り当てられた最低優先度

説明：この引数を使用して、4Dシステム内部コールをダイナミックに設定することができます。セレクタの値に応じて、スケジューラの値は次のアプリケーションのために設定されます。

- このコマンドが4th Dimensionから呼び出された場合、4th Dimension（シングルユーザ）および4D Tools（セレクタ = 10）。
- このコマンドが4D Serverから呼び出された場合、4D Server（セレクタ = 11）。
- このコマンドが4D Clientから呼び出された場合、4D Client（セレクタ = 12）。

（例題1を参照）

セレクタ = 13（4D Server Timeout）

説明：この引数を使用して、4D Serverのタイムアウトの値を変更することができます。4D Serverのタイムアウトのデフォルト値は、サーバ側の「データベースプロパティ」ダイアログボックスの「接続設定」ページで定義します。

セレクタ「4D Server Timeout」により、対応する引数＜値＞に新しいタイムアウト（分単位で指定）を設定できます。この機能は、クライアント側でCPUを占有する時間がかかる処理を実行する前に、タイムアウト設定を長くしたい場合は特に便利です。例えば、膨大なページの印刷などは、予期しないタイムアウトになる可能性があります。

また、2種類のオプションがあります。

■ 引数<値>に正の値を渡すと、グローバルかつ永続的なタイムアウトが設定されます。この新しい値はすべてのプロセスに対して適用され、4Dアプリケーションの初期設定に保存されます（「データベースプロパティ」ダイアログボックスで変更した場合と同じ）。

■ 引数<値>に負の値を渡すと、ローカルで一時的なタイムアウトが設定されます。この新しい値は呼び出し元のプロセスに対してのみ適用され（他のプロセスではデフォルトの値を維持）、例えば処理の終了時のように、クライアントが動作していることを示す信号をサーバが受信すると即座に、デフォルト値へリセットされます。このオプションは、4Dプラグインにより開始された時間のかかる処理を管理する際に便利です。

「タイムアウトしない」オプションを設定するには、<値>に0を渡します。

（例題2を参照）

セレクタ = 14（4D Client Timeout）

説明：この引数を使用して、4D Clientのタイムアウトの値を変更することができます。4D Clientのタイムアウトのデフォルト値は、クライアント側の「データベースプロパティ」ダイアログボックスの「接続設定」ページで定義します。

このセレクタに関する詳細は、「4D Server Timeout」の説明（13）を参照してください。

4D Clientのタイムアウトは、非常に特殊な状況において変更されます。

セレクタ = 15（Port ID）

説明：この引数を使用して、4D Webサーバ機能が監視するTCPポートのIDをオンザフライで変更することができます。デフォルト値は80で、この値は「データベースプロパティ」ダイアログボックスの「Webサーバ I」ページで設定することができます。

セレクタ「Port ID」は、コンパイルしてエンジンを組み込んだ4D Webサーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。

セレクタ = 16（IP Address to listen）

説明：この引数を使用して、ユーザは4D Webサーバが HTTP リクエストを受信するIPアドレスをオンザフライで変更することができます。デフォルトでは、特定のアドレスは定義されていません（<値> = 0）。この引数は「データベースプロパティ」ダイアログボックスの「Webサーバ I」ページで設定することができます。

セレクタ「IP Address to listen」は、コンパイルしてエンジンを組み込んだ4D Webサーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。

引数<値>には、16進数のIPアドレスを渡します。つまり、“a.b.c.d”のようなIPアドレスを指定するには、以下のようなコードを作成します。

C_LONGINT($addr)
$addr:=($a<<24)|($b<<16)|($c<<8)|$d
SET DATABASE PARAMETER(IP Address to listen;$addr)

セレクタ = 17（Character set）

値：
0：Western European（西ヨーロッパ）
1：Japanese（日本語）
2：Chinese（中国語）
3：Korean（韓国語）
4：User-defined（ユーザ定義）
5：Reserved（予備）
6：Central European（中央ヨーロッパ）
7：Cyrillic（キリル文字）
8：Arabic（アラビア語）
9：Greek（ギリシャ語）
10：Hebrew（ヘブライ語）
11：Turkish（トルコ語）
12：Baltic（バルト語）

説明：この引数を使用して、ユーザはデータベースに接続しているブラウザとの通信に、4D Webサーバが使用する文字セットをオンザフライで変更することができます。実際のところ、デフォルト値はOSの言語に依存します。

この引数は「データベースプロパティ」ダイアログボックスの「Web サーバII 」ページで設定することができます。セレクタ「Character set」は、コンパイルしてエンジンを組み込んだ4D Webサーバで役立ちます（この場合、「デザイン」モードへのアクセス手段がありません）。

セレクタ = 18（Max Concurrent Web Processes）
値：デフォルト値は32,000ですが、10から32,000までの任意の値を渡すことができます。

説明：この引数を使用して、4D Webサーバでサポートされる任意のタイプ（コンテキスト、非コンテキスト、または“プロセス再利用”に属するプロセス－セレクタ7、「Webプロセスの最大数」を参照）の同時Webプロセス上限数を定義することができます。この上限数（マイナス1）に達した場合、4Dはそれ以上プロセスを作成しなくなり、HTTPステータス503（「Service Unavailable to all new requests」すべての新しいリクエストへのサービス不可）を返します。

この引数により、同時に行われる非常に膨大な数のリクエストやコンテキスト作成に関する過大な要求の結果として、サーバが飽和状態になることを防げます。また、この引数は「データベースプロパティ」ダイアログボックスでも設定することができます。

理論上、Webプロセスの最大数は次の計算式の結果になります：使用可能メモリ／Webプロセスのスタックサイズ。別の解決策は、ランタイムエクスプローラに表示されるWebプロセス情報を示す方法です。つまり現在のWebプロセス数およびWebサーバの開始以降に達した最大数が示されている情報です。

注：“プロセスの再利用”の上限数より小さい値を渡した場合、この上限数はセレクタ18の値に合わせるために減らされます。必要であれば、再利用の下限数（セレクタ6、Webプロセスの最小数）も変更できます。

セレクタの有効範囲

以下の表に各セレクタの有効範囲を示します。

セレクタ	値	有効範囲
Seq Order Ratio	1	カレントテーブルとプロセス
Seq Access Optimization	2	カレントテーブルとプロセス
Seq Distinct Values Ratio	3	カレントテーブルとプロセス
Index Compacting	4	4Dアプリケーション(*)
Seq Query Select Ratio	5	カレントテーブルとプロセス
Minimum Web Process	6	4Dアプリケーション(*)
Maximum Web Process	7	4Dアプリケーション(*)
Web conversion mode	8	カレントプロセス
Database cache size	9	4Dアプリケーション(*) (**)
4th Dimension Scheduler	10	4Dアプリケーション(*)
4D Server Scheduler	11	4Dアプリケーション(*)
4D Client Scheduler	12	4Dアプリケーション(*)
4D Server Timeout	13	4Dアプリケーション（正の数の場合）(***)
4D Client Timeout	14	4Dアプリケーション（正の数の場合）(***)
Port ID	15	4Dアプリケーション(*)
IP Address to listen	16	4Dアプリケーション(*)
Character set	17	4Dアプリケーション(*)
Max Concurrent Web Processes	18	4Dアプリケーション(*)

(*) この場合、引数<テーブル>は無視されます。

(**) このセレクタは読み込みのみ可能です。

(***) 引数<値>が負の値である場合、この設定はカレントプロセスにのみ影響し、次回のリクエストの際にはリセットされます。

例題

(1)シングルユーザ版の4Dを実行している場合、次のメソッドを使用して、スケジューラの値を定義することができます。

```
C_LONGINT($ticksbtwcalls;$maxticks;$minticks;$lparams)
If(Application type=4th Dimension) ` シングルユーザの4Dを使用
  $ticksbtwcalls:=12
  $maxticks:=20
  $minticks:=7
  $lparams:=($minticks<<16)|($maxticks<<8)|$ticksbtwcalls
  SET DATABASE PARAMETER (4th Dimension scheduler;$lparams)
End if
```

(2)以下のコードでは、予期しないタイムアウトを回避しています。

```
`カレントプロセスに対してタイムアウトを3時間まで延長する
SET DATABASE PARAMETER(4D Server Timeout;-60*3)
`4Dの制御を受けずに、時間のかかる処理を実行する
...
WR PRINT MERGE (Area;3;0)
...
```

(3)IPアドレス192.193.194.195は、次のコードを使用して設定します。

```
SET DATABASE PARAMETER(IP Address to listen;0xC0C1C2C3)
```

SET HTTP HEADER

SET HTTP HEADER (ヘッダー ¦ 名称配列{; 値配列})

引数	タイプ		説明
ヘッダー	文字列	→	HTTP要求ヘッダ全体を保持するフィールドか変数。もしくはHTTPヘッダのフィールド名を保持する配列
名称配列			
値配列	文字列	→	HTTPヘッダのフィールド値を保持する配列

説明

このコマンドは、4Dによって Web ブラウザへ送り返される HTTP ヘッダ内のフィールドを設定できるようにするものです。これは、非コンテキストモードでの Web プロセスにのみ影響します。

このコマンドは、"cookies"の管理も可能にします。
このコマンドは2通りの書き方ができます。

構文 1 ： SET HTTP HEADER(header)

TEXT 型のフィールドもしくは変数に格納した HTTP ヘッダ全体を引数ヘッダへ渡します。

この構文により”HTTP/1.0 200 OK”+Char(13)+”Set-Cookie:C=HELLO”のようなヘッダータイプを記述することができます。

Windows や MacOS でヘッダーフィールドは CR か CR+LF（Carriage return + Line feed）で分割されます。

▼ カスタム“cookie”の例を以下に示します。

```
C_TEXT($vTcookie)
$vTcookie:="SET-COOKIE: USER="+String( Abs( Random))+"; PATH=/"
SET HTTP HEADER($vTcookie)
```

注：このコマンドは、引数＜ヘッダ＞としてリテラルテキストタイプの定数を受け付けません。必ず 4D 変数またはフィールドを指定しなければなりません。

構文についてのより詳しい情報は、インターネットアドレス http://www.w3c.org で得られる R.F.Cs（Request For Comments）を参照してください。

注：フィールドは常に cr/lf シーケンス（キャリッジリターン／ラインフィード）で分離されていなければなりません。構文についてのより詳しい情報は、インターネットアドレス www.w3c.org で得られる R.F.Cs（Request For Comments）を参照してください。

構文 2 ： SET HTTP HEADER(fieldArray; valueArray)

HTTP ヘッダーはふたつのテキスト配列、フィールド配列と変数配列によって定義されます。ヘッダーは次のように記述されます。

```
fieldArray{1} := "X-VERSION"
fieldArray{2} := "X-STATUS"
fieldArray{3} := "Set-Cookie"
valueArray{1} := "HTTP/1.0"*
valueArray{2} := "200 OK"*
valueArray{3} := "C=HELLO"
```

はじめのふたつの項目は返答の一行目です。それらが入力されたとき、配列の 1 番目、2 番目の項目でなくてはなりません。しかし、4D は以下のように省略したヘッダー書式も受け付けてくれます。

```
fieldArray{1} := "Set-Cookie"
valueArray{1} := "C=HELLO"
```

上記の例では HTTP ヘッダの 1 行目であるべきステータスラインの情報が省略されていますが、4D が自動的に“HTTP/1.0 200 OK”を内部で補います。

同一プロセス内で複数回に渡って SET HTTP HEADER を実行した場合は、最後の回の内容だけが有効になります。

Server、Date や Conten-length フィールドは常に 4D が設定します。

START WEB SERVER

START WEB SERVER

説明

START WEB SERVER コマンドは、内蔵の 4th Dimension Web サーバ機能を使用して、イントラネットネットワークまたはインターネット上でデータベースのサービスを開始します。

Web サーバが正常に起動された場合には、システム変数 OK に 1 が設定され、そうでなければシステム変数 OK は 0（ゼロ）が設定されます。例えば、TCP／IP ネットワークプロトコルが正しく設定されていない場合には、システム変数 OK に 0 が代入されます。

システム変数とシステムセット

Web サーバが正常に開始された場合はシステム変数 OK に 1、そうでない場合はシステム変数 OK に 0 が設定されます。

System folder

System folder{(タイプ)}→文字列

引数	タイプ		説明
タイプ	倍長整数	→	フォルダのタイプ
戻り値	文字列	←	指定したフォルダへのパス

説明

System folder 関数は、アクティブなWindowsまたはMacintoshシステムフォルダ内にあるシステムによって使用されるフォルダへのパス名、あるいはアクティブなWindowsまたはMacintoshシステムフォルダ自体へのパス名を返します。

オプションの引数＜タイプ＞には、フォルダのタイプを示す値を指定します。以下の定数が、エクスプローラの『定数』タブページのテーマ「System folder」から利用可能です。

定数	タイプ	値
System	倍長整数	0
Fonts	倍長整数	1
Preferences or Profiles (All Users)	倍長整数	2
Preferences or Profiles (Current User)	倍長整数	3
Startup Items (All Users)	倍長整数	4
Startup Items (Current User)	倍長整数	5
Mac Shutdown Items (All Users)	倍長整数	6
Mac Shutdown Items (Current User)	倍長整数	7
Apple or Start Menu (All Users)	倍長整数	8
Apple or Start Menu (Current User)	倍長整数	9
Mac Extensions	倍長整数	10
Mac Control Panels	倍長整数	11
System Win	倍長整数	12
System32 Win	倍長整数	13
Favorites Win	倍長整数	14
Desktop Win	倍長整数	15
Program Files Win	倍長整数	16

フォルダのなかには、カレントユーザに依存しているものがあります。定数の2から9を使用すると、取得しようとするパス名が、すべてのユーザで共有のフォルダのものか、カレントユーザ用にカスタマイズされたフォルダへのものかを選択できます。

注：定数「Mac Shutdown Items」、「Mac Extensions」、「Mac Control Panels」は、Mac OSでのみ使用できます。これらの定数をWindowsで使用すると、**System folder**関数は空の文字列を返します。

また逆に、定数「System Win」、「System32 Win」、「Favorites Win」、「Desktop Win」、「Program Files Win」はWindowsでのみ使用できます。これらの定数をMacOSで使用すると、**System folder**関数は空の文字列を返します。

引数<タイプ>を省略すると、この関数はアクティブなシステムフォルダへのパス名を返します（= System定数）。

Web サービス：システム設定

MacOS X における Web サーバの設定

MacOS X において、Web パブリッシング用に予約されている TCP／IP を使用するには、特定のアクセス権が必要となります。つまり、そのマシンの“ルート”ユーザだけが、これらのポートを使用してアプリケーションを起動することができます。

これらのポート番号は 0 から 1023 までです。デフォルトとして 4D データベースの公開には、標準モードでは TCP ポート 80、SSL モードではポート 443 が使用されます。

“ルート”ユーザとして接続せずに、デフォルトの TCP ポートを使用して 4D データベースを公開すると、警告ダイアログボックスが表示されます。

標準の HTTP 発行用のデフォルトポート番号は変更することができます。しかし、SSL での公開を行うには、ポート 443 を使用しなければなりません。

データベースの公開には 2 種類のオプションがあります。

■ 4D Web サーバで使用する TCP ポート番号を変更する。

1023 より大きいポート番号を使用しなければなりません（例えば、ポート 8080）。ポート番号の変更を行うには、「データベースプロパティ」ダイアログボックス、または **SET DATABASE PARAMETER** コマンドを使用します。

しかし、ポート番号を変更できるのは、標準の HTTP プロトコルで公開された 4D Web サーバ、言い換えれば、SSL プロトコルを使用しないサーバだけである点に留意してください。SSL を利用して 4D Server を公開するには、ポート 443 を使用する必要があります。また、暗号化モードで 4D Web サーバを公開するためには、“ルート”ユーザとして接続しなければなりません。

■“ルート”ユーザとして接続する。

デフォルトとして、MacOS X が動作するマシンでは“ルート”ユーザが有効ではありません。まず“ルート”ユーザを有効にしたあと、そのユーザ名を使用してログインしなければなりません。“ルート”ユーザを有効にするには、Apple 社より提供され、「Applications:Utilities」フォルダにインストールされている NetInfo Manager ユーティリ

ティを使用します。

ユーティリティの起動後、「ドメイン」メニューから「セキュリティ」コマンドを選択し、さらに「ルートユーザを有効」オプションを選択します。まず最初に同じメニューにある「認証...」コマンドを使い、マシン管理者を指定しなければなりません（短い名前と管理者のパスワードを入力する）。

この操作に関する詳細は、MacOS Xのドキュメントを参照してください。

“ルート”ユーザを作成したら、このセッションをクローズし（Appleメニュー）、“ルート”ユーザ名を使用してログインします。これで、ポート番号80でWebサーバを起動したり、あるいは暗号化接続を使用して4D Webサーバを起動することができます。

SSL プロトコルの使用

4D における SSL のインストールとアクティブ化

4D で SSL プロトコルを使用したい場合には、以下のコンポーネントをインストールしてください。

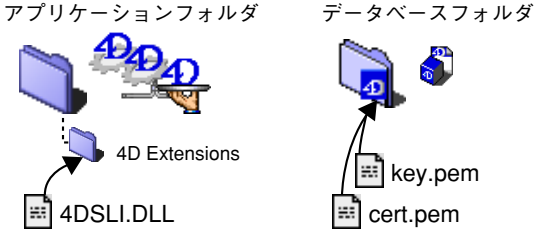
■ 4DSLI.DLL：SSL 管理専用の Secured Layer Interface

このファイルは、データベースを発行する 4D アプリケーションの[4D Extensions]フォルダ内に配置してください。

■ key.pem（Web サーバのみ）：Web サーバ発行用の暗号化秘密鍵を含むドキュメント。このファイルはデータベースフォルダ内に配置してください。

■ cert.pem（Web サーバのみ）：Web サーバ発行用の“証明書”を納めたドキュメント。このファイルは、データベースフォルダ内に配置してください。

4D Web サーバで SSL を実装するために必要なファイル



注：暗号化コマンドである **ENCRYPT BLOB** および **DECRYPT BLOB** を使用する際にも、4DSLI.DLL が必要となります。

これらのファイルがインストールされると、Web サーバおよびクライアント／サーバ（存在する場合）も、暗号化モードで接続できます。

デフォルトでは、SSL 接続は Web サーバに対してはアクティブとなり、クライアント／サーバ接続に対しては非アクティブになります。SSL 設定は、「データベースプロパティ」ダイアログボックスの「接続設定」ページにあります。

SSLデータのやり取り専用のTCPポートは443です。したがって、SSLを使用するWebサーバは、1台のマシンに1つしかインストールできません。「データベースプロパティ」ダイアログボックスの「Webサーバ I」ページで定義したTCPポートは、標準モードのWebサーバ接続に対して使用されます。

一般的に、接続モードが何であれ、4D Webサーバの管理のために設定した各データベースプロパティ（パスワード、タイムアウト、キャッシュサイズ等）は適用されます。

既存のWebサーバにおけるSSLの使用

4D WebサーバでSSLを使用する場合に、特別なシステム構成は必要ありません。しかし、SSL Webサーバは非暗号化モードでも動作できるという点に注意してください。また、接続モードは、ブラウザ側の要求があれば（例えば、ブラウザのURLエリアでユーザが“HTTPS”を“HTTP”で置き換えた場合）、もう一方のモードへ切り替えることができます。開発者は、非暗号化モードで行われたリクエストを禁止したり、リダイレクトすることが可能です。**Secured Web connection**関数を使用すると、現在の接続モードを取得できます。

同様に、バージョン6.8のSSLを使用した4D WebサーバでSSLを実行する際、ページに配置されたURL、および同一サイトから他のページを参照しているURLはすべて、“HTTPS”で始まっていることを確認してください。URLが“HTTPS”で始まらない場合、接続は非暗号化モードに切り替わります。

Webサービス：Webサーバセッテイング

4Dでは、必要に応じてWebセッションのカスタマイズすることができます。設定には、以下のオプションが有効です。

- デフォルトのホームページを定義する
- データ入力制御のためにJavascriptを使用する
- スタティックなページでの、4DVARのコメントを使用する
- HTMLテキストの変換と、Webフィルタの定義
- ページキャッシュの利用
- サーバがHTTPクエリを受信するIPアドレスを定義し、セカンダリIPアドレスをインストールする
- Webプロセスの同時処理数の上限を定義する
- 「4D WebSTARを許可する」オプションを設定する

セカンダリ IP アドレスのインストール

マルチホーミングシステムの導入には、OS に応じた特別な設定が必要になります。

注：4D／4D Server アプリケーションは 1 つの IP アドレスのみ使用できます（4D／4D Server の 1 インスタンスへのアクセス用に複数の IP アドレスを使用することはできません）。

Macintosh 上の設定

▼ Macintosh 上でマルチホーミングの設定をするには

1. この機能を使用するには、バージョン 1.3 以降の Open Transport を使用する。
2. 「コントロールパネル」の「TCP／IP」を開く。
3. 「設定方法：」のポップアップメニューから「手入力」を選択する。
4. テキストファイルを作成し、"IP Secondary Addresses"という名前を付け、システムフォルダ内の「初期設定」フォルダに入れる。

   IP Secondary Addresses の各行にシステムで使用するセカンダリ IP アドレス用の IP アドレス、サブネットマスクおよびルータアドレスを記述します。

WindowsNT、Windows2000 上の設定

▼ WindowsNT または Windows2000 上でマルチホーミングの設定をするには

1. スタートメニュー＞設定＞コントロールパネル＞ネットワーク＞プロトコルタブ＞TCP／IP プロトコル＞プロパティボタン＞詳細ボタン

   上記の作業により「詳細な IP アドレス指定」ダイアログボックスが表示されます。

2. IP アドレス内の「追加」ボタンをクリックし、追加する IP アドレスを入力する。

   この作業にはネットワーク管理者のサポートが必要になることがあります。より詳しい情報は、Windows ドキュメントをご覧ください。

「4D WebSTARを許可する」オプション

「データベースプロパティ」ダイアログボックスの「Web サーバI」ページにおいて、「4D WebSTARを許可する」オプションを使用することができます。このオプションは、4D「4D Connect」プラグインによる4D Webサーバへの接続を許可（チェック）、または禁止（チェックなし）する目的のために設定されています。

「4D Connect」は4D WebSTARウェブサーバ用のプラグインで、4D Webサーバとの通信を可能にします。

セキュリティ上の理由から、デフォルトでは「4D WebSTARを許可する」オプションがチェックされていません。お使いのWebの環境設定に応じ、4D社では以下のような設定をお勧めします。

■ お使いの4D Webサーバが、「4D Connect」プラグインによって4D WebSTARから接続されない場合は、このオプションをOFFにしておいて下さい。

■ お使いの4Dウェブサーバが、「4D Connect」プラグインによって、4D WebSTARから接続される場合は、このオプションをONにして下さい。

この設定の場合、4D Webサーバをファイアーウォールの下で稼働させ、そのファイアーウォールを利用して4Dへの直接リクエストをフィルタリングすることをお勧めします。