4D v16への変換

"4D v16 への変換" マニュアルへようこそ。ここでは、4D v15 のデータベースを 4D v16用に変換する前、変換中、そして変換後にチェックすべきさまざまな点についての説明をします。

4D v15 データベースの 4D v16 への変換はスムーズに行われるはずですが、すべてが上手く行くためのいくつかの推奨点が "**変換の原則**" の章にまとめられています。しかしながら、変換が終わった後にもいくつかチェックすべき事柄があります。それは **"新しい互換性オプション"** と "**振る舞いの変更**" であり、どちらもアプリケーションレベルと 4Dコマンドレベルで関わってくるため、4D v16 の新機能を使いこなすにはどちらも理解しておく必要があります。

最後に、このマニュアルでは **4D v16 で廃止予定の機能**をまとめてあります。デベロッパーの方はここを見ることで、機能の変更や代替の必要性を検討するための情報が手早く得られます。

注: このマニュアルに記載されている新機能・変更の一部は、4D v15の "R-リリース" プログラムで既に紹介・導入されているものです。

以前の、あるいはさらにもっと古いデータベースを変換する際には、通常それらの間のバージョンも介して変換することが必要になります。それぞれの変換の際にチェックすべきさまざまな項目については、以前のバージョンでのドキュメントを参照してください:

- **4D v15**: "4D v15への変換 (PDF)" と "廃止予定の機能と削除された機能 (4D v15 以降・PDF)"
- **4D v14**: "4D v14へのアップグレード (PDF)" と "廃止予定の機能と削除された機能 (4D v14 以降・PDF)"
- **4D v13**: "4D v13へのアップグレード (PDF)"
- **4D v12**: "4D v12へのアップグレード (PDF)"
- **4D v11**: "4D v11 SQLへのアップグレード (PDF)"

  注: 各アップグレードマニュアルの最初の方で、前バージョンのデータベースの変換方法と、廃止予定・削除済み機能について説明しています。

変換の原則

変換の前にしておくべきこと

- 変換を行うためには、データベースの **"インタープリター" 版** (ストラクチャーのxxxx.4DB ファイル) と、デザイナーパスワードが必要になります。
- 変換の前に、**必ずデータベースのコピー (バックアップ) を作成してください。**
- シンタックスチェックを実行します。データベースをコンパイルしない場合も、このチェックによって起こり得るエラーを検知することができます。
- **Maintenance and Security Center (MSC)** を使用してストラクチャーとデータの検査と修復を行ってください。
- **GET PICTURE FORMATS** コマンド (または 4D Pack **_o_AP Is Picture Deprecated** コマンド) を使用して、データベース内にPICTファイルがあるかどうかをチェックし、あった場合には **CONVERT PICTURE** コマンドを使用して変換します (4D v14では、**SET DATABASE PARAMETER** コマンドのセレクターによって、32-bit版で QuickTime がまだ使用されている可能性があります)。
- (任意)データのジャーナリングが必要な場合、プライマリーキー (v14以降実装) を実装することができます(デザインリファレンス マニュアルの **主キーを設定、削除する** を参照ください)。
- v13.5以降のデータベースにおいて、**重複不可**属性を持つフィールドは**インデックスが必須**となりました。今後はインデックスがついてない重複不可属性のフィールド内では一切レコードを作成/編集することはできなくなります。レコードを保存しようとすると、エラーが生成されます (-9998 重複不可のレコードが存在します・1088 インデックスが無効または未設定です)。存在しないインデックスの作成方法、またはインデックスがなされていないフィールドをすべてまとめたディスクファイルの生成方法については、"4D v15への変換" ドキュメントの **付録: 変換に有用なメソッド** を参照ください。

変換の方法

v15 の 4D または 4D Server (v11、v12、v13、v14 も同様) を使用して作成されたデータベースは 4D v16 において互換性があります (ストラクチャーとデータファイル)。どんなインタープリター版のストラクチャーファイルも変換することができます。変換するためには、4D v16 を起動し、そのストラクチャーファイル (xxx.4DB ファイル) をインタープリターモードで開くだけです。

ストラクチャーファイルが変換されることを警告するダイアログが表示されます:

ストラクチャーファイルが 4D v16 に変換されたあとは、その前のバージョンで開くことはできなくなります。

4D v15 および 4D v15 Rx のデータベースの変換を行う場合、データファイルは変換されません。
しかし、4D v14 およびそれ以前のデータベースを変換する場合はデータファイルも変換されるため、二つ目のダイアログが表示されます:

4D v16 用に変換されたデータファイルは、v14.4 以降 または 4D v15 (4D v14 R5) を使用すれば引き続き開くことができます。

変換の後で

Maintenance and Security Center(MSC) を再度使用してストラクチャーとデータの検査と修復を行ってください。

ストラクチャーに関する留意点:

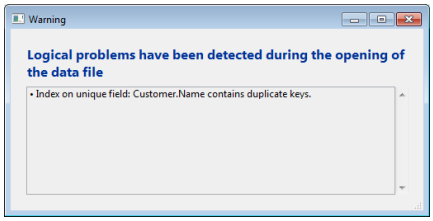
- オーファンメソッド (__Orphan__xxxxx) は MSC のログファイルに警告として記録されます。オーファンメソッドのコードが不要であることを確認したあと、これらはエクスプローラーを使って削除することができます。
- フォーム上のオブジェクト名は重複してはいけません。これらは MSC のログファイルに警告として記録されます。データベースの修復を行うとこれらの名称を変更することができます (コードにオブジェクト名を使っている場合は確認が必要です)

ストラクチャーに関する新機能:

- ストラクチャーで PICT フォーマットを使用しているピクチャーを検知します。MSC のドキュメントの **アプリケーションの検証** を参照ください。

データに関する新機能: 重複不可フィールド内の重複を検知します。次の情報が追加で得られます:

- MSC や **VERIFY DATA FILE** などのコマンドを使用した場合、生成されるログファイルにはテーブル名、フィールド名、そして重複している値が記録されます。
  注: データの入力中に重複する値を検知すると表示されるエラーダイアログボックスにも、テーブル名・フィールド名・重複値が表示されます。また、**GET LAST ERROR STACK** コマンドでも重複に関する詳細情報が得られます。
  4D がデータファイルを開く際にインデックスを構築 (あるいは再構築) する必要がある場合、重複不可に設定されている関連フィールドでの重複が自動で検知されます。この場合、データベースを開く前に特別な警告ダイアログボックスが表示され、重複した値を特定・削除するために必要な情報をユーザーに提供します:



インデックスの再構築

Unicode ライブラリ (ICU - International Components for Unicode) のアップデートへの対応のため、4D v16 へのアップグレードの際にすべてのテキストおよびキーワードインデックスの再構築が必要です。変換されたデータベースを初めて起動させたときにこの処理は自動で行われます (注意: この処理にはかなりの時間を要する場合があります)。

同様に、v16 データベースを 4D v15 R5以前のバージョンで開こうとすると、すべてのテキストおよびキーワードインデックスの再構築がトリガーされます。

注: 4D v16 ではデータベース全体の再インデックスのアルゴリズムが大幅に最適化されました。プロセス全体が劇的に速くなり、最大で2倍の速さを達成しました。全体再インデックスは、例えばデータベース修復後や.4dindxファイルが削除された後では必須です。

GET PICTURE FORMATS

GET PICTURE FORMATS (picture ; codecIDs)

引数	型		説明
picture	ピクチャー	➡	解析するピクチャーフィールドあるいは変数
codecIDs	テキスト配列	⬅	ピクチャーのコーデックID

説明

GET PICTURE FORMATSコマンドは、引数として渡された*picture*引数内に含まれている全てのコーデックIDの配列を返します。 4D ピクチャー (フィールドまたは変数) は、PNG、BMP、GIF など、複数の異なるフォーマットでエンコードされた同一の画像を格納することができます。

*picture*引数には、含まれるフォーマットを*codecIDs*配列内に取得したいピクチャーフィールドあるいは変数を渡します。

返されるコーデックIDは、**PICTURE CODEC LIST**コマンドと同様に4Dによって確立されます。これらは以下の形式で受け取ることが可能です:

- 拡張子(例 : ".gif")
- Mimeタイプ(例 : "image/jpeg")
- 4文字のQuickTimeコード

注:

- 4Dによって内部的に管理される以下のコーデックについては、必ず拡張子形式で返されます:JPEG、PNG、TIFF、GIF、BMP、SVG、PDF、EMF
- 4文字のQuickTimeコードは、QuickTime support互換性オプションが(**SET DATABASE PARAMETER**コマンドを使用して)設定されているデータベースにおいて返すことが可能です。しかしながら、QuickTimeは4Dではサポートされておらず、QuickTimeコーデックの使用は推奨されません。

ピクチャーコーデックIDについてのより詳細な情報については、**ピクチャ**の章を参照して下さい。

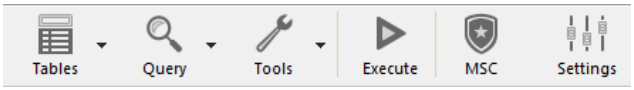
例題

カレントレコードのフィールド内に保存されているピクチャーフォーマットを知りたいという場合を考えます:

```
ARRAY TEXT($aTPictureFormats;0)
 //保存されている全てのフォーマットを取得
GET PICTURE FORMATS([Employees]Photo;$aTPictureFormats)
```

互換性ダイアログ

このダイアログに行くためには、まずメインのツールバーの "設定" をクリックします:



その後 "互換性" のタブをクリックします。

新しい互換性オプション

4D v16 では**互換性**ページに新しいオプションが追加されました: **アプリケーション配布では新しいアーキテクチャーを使用**

- 4D v15 R4以降に作成されたデータベースの場合: このオプションがデフォルトで選択されており、 "新しいアーキテクチャー" は自動で有効化されています。
- それ以前のバージョンから変換されたデータベースの場合: このオプションは互換性のためデフォルトで無効化されています。これらのデータベースで "新しいアーキテクチャー" の新機能を使用するにはオプションをチェックして有効化します。

4D v16 以降のすべてのアプリケーションでこのオプションは提供され、4D アプリケーションの運用に関わる新機構の有効化・無効化の操作を行えます (最終アプリケーションを生成するマシンで設定を確認してください)。このオプションによって管理される機構についての詳細は **最終アプリケーションでのデータファイルの管理** および **クライアントアプリケーションによる接続の管理** を参照してください。

その他のオプション

継続して提供されているその他の互換性オプションも同ダイアログボックスにて確認することができます。これら互換性オプションはバージョンを経て少しずつ追加されてきたものですので、最初にデータベースを作成してからの期間に応じてオプションリストは長くなります:

これらのオプションについての詳細は **互換性ページ** を参照ください。

削除されたオプション

"**ブラケットの代わりに4DVARコメントを使用する**" オプションは 4D v16 の**互換性**ページより削除されました。

最終アプリケーションでのデータファイルの管理

データファイルを開く

ユーザーが組み込みアプリ、またはアプリケーションのアップグレード(スタンドアロンアプリ、またはクライアントーサーバーアプリ)をローンチした場合、4Dは有効なデータファイルを選択しようとします。アプリケーションによって、複数の場所が順次検索されます。

組み込みアプリの起動のオープニングシーケンスは以下のようになっています:

1. 4Dは**最後に開かれたデータファイル**を、以下の説明のように開こうとします(これは初回起動時には適用されません)。
2. 見つからない場合、4Dは.4DCファイルの隣にある**default dataフォルダー**内のデータファイルを、読み込みのみモードで開こうとします(4D v15からの新機能、詳細は以下参照)。
3. これも見つからない場合、4Dは標準のデフォルトデータファイルを開こうとします(.4DCファイルと同じ場所にある、同じ名前のファイル)。
4. これも見つからない場合、4Dは標準の"データファイルを開く"ダイアログボックスを表示します。

最後に開かれたデータファイル

最後に開かれたファイルへのパス

アプリケーション配布では新しいアーキテクチャーを使用互換性オプションがチェックされているとき(**互換性ページ** を参照して下さい)、4Dでビルドされたあらゆるスタンドアロンまたはサーバーアプリケーションは、最後に開かれたデータファイルのパスをアプリケーションのユーザー設定フォルダ内に保存します。

互換性に関する注意: 以前のバージョンにおいては、この情報はストラクチャーファイル内に保存されていました。

アプリケーションのユーザー設定フォルダの場所は、以下の宣言で返されるパスに対応しています:

```
userPrefs:=Get 4D folder(Active 4D Folder)
```

データファイルパスは*lastDataPath.xml*という名前の専用のファイルに保存されています。

このアーキテクチャーのおかげで、アプリケーションのアップデートを提供するとき、ローカルユーザーデータファイル(最後に使用されたユーザーファイル)は初回の起動から自動的に開かれます。

この機構は通常標準の配布で安定しています。しかしながら特定の場合、例えば組み込みアプリケーションを複製した場合等において、データファイルとアプリケーションのリンクを変えたいことがあるかもしれません。より詳細な情報については、次の"データリンクモードを設定する"の章を参照して下さい。

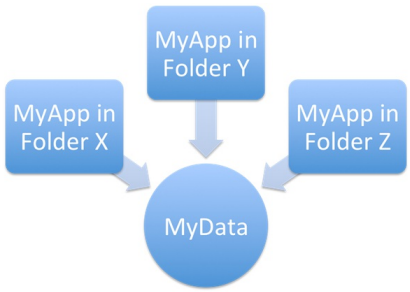
データリンクモードを設定する

コンパイルされたアプリケーションでは、4Dは最後に使用されたデータファイルを自動的にしようします。デフォルトで、新しいアーキテクチャーが有効化された場合(4D v15 R4以降、詳細は上記の章を参照して下さい)、データファイルのパスはアプリケーションのユーザー設定フォルダに保存され、**アプリケーション名**でリンクされます。

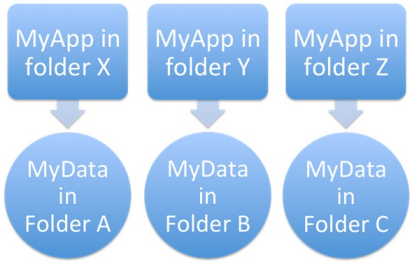
この場合、異なるデータファイルを使用するために組み込みアプリを複製したい場合には適さない事があります。複製されたアプリは同じアプリケーションユーザー設定フォルダを共有するため、同じデータファイルを常に使用します(アプリケーションが最後に使用したファイルが開かれるため、データファイルを解明した場合でも結果は同じです)。

そのため4Dでは**アプリケーションパス**を使用してデータファイルとリンクすることも可能です。このとき、データファイルは特定のパスを使用してリンクされるので、単に最後に開かれたファイルが使用されるとは限りません。

データがアプリケーション名でリンクされている場合の複製:



データがアプリケーションパスでリンクされている場合の複製:



このデータリンクモードはアプリケーションビルドプロセス時に選択することができます。以下の二つから選択可能です:

- アプリケーションをビルドダイアログボックスの**アプリケーションページ**または**クライアント/サーバーページ**を使用する。
- **LastDataPathLookup**(スタンドアロンアプリケーション時)または**LastDataPathLookup**(サーバーアプリケーション時)XMLキーを使用する。

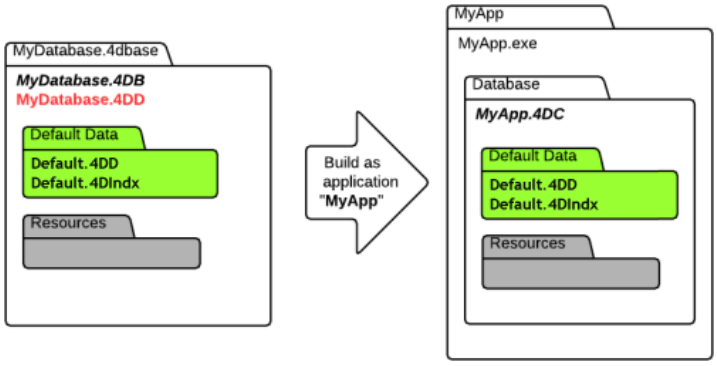
デフォルトのデータフォルダを定義する

4D v15 では組み込みアプリに簡単にデフォルトのデータファイルを埋め込むことが出来るようになりました。これにより、特別なダイアログボックスを表示させることなく、エンドユーザーのマシンにアプリケーションをインストールまたはアップデートさせることができるようになります。デフォルトのデータファイルを定義するためには、以下の用にします:

- デベロッパはデフォルトのデータファイルを、データベースパッケージ(4dbase)内のデフォルトのフォルダ内に保存する必要があります:
  - デフォルトのデータフォルダ名は"Default Data"である必要があります。
  - デフォルトのデータファイル名は"Default.4DD" である必要があり、でおフォルトのデータフォルダ内に保存されている必要があります。

  デフォルトのデータファイルには全ての必要なファイルもそろっている必要があります: インデックス、ジャーナル、外部BLOB、、、等です。
  (デベロッパが責任をもって有効なデフォルトデータファイルを用意して下さい)
- アプリケーションがビルドされたとき、アプリケーションのビルドプロセスが、このデフォルトデータフォルダを組み込みアプリ内に統合します。

以下はこの新機能を図示した画像です:



デフォルトのデータファイルが初回起動時に検知された場合、データファイルは自動的に読み込み専用モードで開かれるので、色んなカスタムのオペレーションが実行できるようになります。

互換性ページ

互換性ページには、以前の4Dバージョンとの互換性を管理するためのパラメーターがまとめられています。ここに表示されるオプションの数は、元のデータベースが作成されたバージョン (2004.x, v11, v12等) や、このデータベースで行われた設定の変更により異なります。

注: このページは現在のバージョンで作成されたデータベース (変換されていないデータベース) には表示されません。

- **ダイアログボックスでフィールドを入力可にする**: 以前のバージョンでは **DIALOG** コマンド等で表示されたダイアログボックスでフィールドに値を入力することができませんでした。この制限は 4D 2004 で取り除かれています。データを表示するだけの目的でダイアログにフィールドを表示している場合、以前の動作を保持することができます。バージョン2004に変換されたデータベースではこのオプションが選択され、v2004で作成されたデータベースでは選択されていません。

- **ラジオボタンを名前でグループ化する**: 以前のバージョンではラジオボタンの排他制御はボタンに割り当てる変数名の先頭バイトで判定されていました (例えば *m_button1*, *m_button2*, *m_button3* など)。4D 2004以降はフォームエディター上でオブジェクトをグループ化することで排他制御が行われるようになっています。この点については **ラジオボタンとピクチャーラジオボタン** を参照してください。
  この新しいモードはラジオボタン、3Dラジオボタン、ピクチャーラジオボタンで有効です。互換性を保つため、変換されたデータベースでは以前のモードが使用されています。しかしこのオプションの選択を解除すれば新しいモードを使用できます。v2004 で作成されたデータベースでは新しいモードが使用されます。
- **PRINT SELECTION中、レコード毎にフォームをリロード**: 以前のバージョンの 4Dでは **PRINT SELECTION** コマンドを使用した印刷中に使用されるフォームは、各レコード毎にリロードされていました。これにより On printing detail フォームイベントで開発者が言語を使用して変更したかもしれないオブジェクトの設定がすべて自動的に再初期化されていました。
  パフォーマンスを最適化するためにこのメカニズムは 4D 2004 で取り除かれました。今後はフォームメソッドを使用して 4D開発者が初期化を行わなければなりません。この動作は On display detail フォームイベントを使用するリストフォームと同じです。しかしながらこのオプションを使用して以前の動作を保持することができます。v2004で作成されたデータベースは新しいモードを使用します。
  オプションが選択されていない場合、ブラケット記法 ([MAVAR]) を使用します。この記法は以前のバージョンの4D Webサーバーで使用されていたプロプリエタリな方法であり、推奨されません。
- **新しいコンテキスト参照モードを使用しない**: このオプションが選択されていない場合、4D WebサーバーはHTMLのベースURLにコンテキスト番号を挿入します。
  以前のモードでは、 4D Webサーバーはブラウザーに送信する各項目にコンテキスト番号を送信しており、結果処理が遅くなっていました。しかしながら互換性のためこのオプションが選択されているかもしれません。このオプションを変更した後は設定を有効にするためにデータベースを再起動しなければなりません。
- **未知のURLから"/"を取り除く**: 以前の4Dではディスク上に存在しないファイルがURLとしてリクエストされた場合、On Web Authentication や On Web Connection データベースメソッドの$1引数に先頭の"/"が取り除かれたURLが渡されていました。この動作は4D 2004で変更されました。しかし以前の動作に基づいた実装を行っている場合にはこのオプションを選択します。
- **外部からのドラッグ＆ドロップを拒否**: v11以降、ピクチャーなどのファイルや選択されたテキストオブジェクトなどを4Dにドラッグ＆ドロップできるようになりました。変換されたデータベースではこの動作を想定したメソッドが書かれていないために期待した動作とならないかもしれません。このオプションを選択すると外部オブジェクトを4Dフォームにドロップできなくなります。ただしこの場合でも**自動ドロップ**オプションを使用すると外部オブジェクトの挿入が可能である点に留意してください。アプリケーションはドロップされたテキストやピクチャーを解釈します (**ドラッグ＆ドロップ**参照)。
- **QUERY BY FORMULAをサーバー上で実行**と**ORDER BY FORMULAをサーバー上で実行**: 4D v11より最適化の目的で、フォーミュラによるクエリや並び替えコマンドがサーバー上で実行されるようになりました。そして結果だけがクライアントマシンに返されます。この動作は以下のコマンドで有効です: **QUERY BY FORMULA**、**QUERY SELECTION BY FORMULA**、**ORDER BY FORMULA**。変数が直接フォーミュラ内で使用されている場合、クライアントマシン上の変数値を使用してフォーミュラが呼び出されます。例えば

  ```
  QUERY BY FORMULA([aTable];[aTable]aField=theVariable)
  ```

  このコードがサーバー上で実行された場合でも、*myvariable*変数値はクライアントマシン上のものが使用されます。他方この原則はフォーミュラにメソッドが使用され、そのメソッド内で変数が参照されている場合には当てはまりません。この場合サーバー上で変数が解釈されます。
  変換されたデータベースではこの点が考慮されていない可能性があるため、デフォルトでこれらのコマンドはクライア

ントマシン上でフォーミュラを実行します。新しいモードを使用したい場合はこれらのオプションを明示的に選択します。
注: このオプションは**SET DATABASE PARAMETER**コマンドで設定することもできます。

- **QUERY BY FORMULAでSQL JOINを使用**: 4D v11より **QUERY BY FORMULA** や **QUERY SELECTION BY FORMULA** コマンドはSQLの結合モデルに基づくJOINを実行するようになりました。これによりストラクチャーエディターで自動リレーションが設定されていなくても [Table_A]field_X=[Table_B]field_Y のようなフォーミュラを使用できるようになりました。
既存のデータベースでこの動作が考慮されていない場合、予期しない動作となることがあるため、変換されたデータベースではこの機能がデフォルトで無効にされています。データベースコードを見直した後、このモードを有効にすることを推奨します。
注:
  - "SQL JOIN"モードが有効な場合でも、以下のケースでは QUERY BY FORMULA や QUERY SELECTION BY FORMULA コマンドはストラクチャーエディターで設定された自動リレーションを使用します:
    - フォーミュラを{field ;comparator ;value}形式に分解できない場合
    - 同じテーブルの 2つのフィールドが比較されている場合
  - **SET DATABASE PARAMETER** コマンドを使用してプロセス毎にこのオプションを設定できます。
- **トランザクションのネストを許可する**: マルチレベルトランザクションのサポートを有効にします。4D v11以降、マルチレベルのトランザクションがサポートされるようになりました。既存のデータベースでこの動作が考慮されていない場合、予期しない動作となることがあるため、変換されたデータベースではこの機能がデフォルトで無効にされています (トランザクションは1レベルに制限されます)。マルチレベルのトランザクションを使用したい場合、このオプションを選択します。
注: このオプションはSQLエンジンで実行されるトランザクションには影響しません。SQLのトランザクションは常にマルチレベルです。
- **Unicodeモード**: カレントデータベースのUnicodeモードの有効/無効を切り替えます。Unicodeモードではデータベースエンジン、言語、メニューなどでネイティ ブにUnicode文字が処理されます。非Unicodeモードでは日本語環境の場合Shift_JISが使用されます。
この設定に関わらずデータファイルはUnicodeが使用され、文字列の評価にはICUが使用されます。
v2004以前から変換されたデータベースではこのオプションが選択されていません。しかしながらこのオプションを選択し、必要なコードの修正を適用することを強く推奨します。非Unicodeモードを使用しても、過去のバージョンとの完全な互換性は提供されません。
注:
  - このオプションのスコープはデータベースごとです。インタープリターモードではUnicodeモードのデータベースに非Unicodeモードのコンポーネントをインストールしたり、あるいはその逆を行ったりすることが可能です。
  - **SET DATABASE PARAMETER** コマンドを使用してUnicodeモードを設定することができます。

  4DにおけるUnicodeサポートについては **EXPORT TEXT** を参照してください。
- **ピリオドとカンマを数値フォーマットのプレースホルダーとして使用する**: v11以降、4Dは数値の表示フォーマットにシステムの地域設定パラメーターを使用するようになりました (**表示フォーマット** の"数値フォーマット"参照)。4Dは自動で数値表示フォーマット中の","を千の位区切り文字、"."を小数点として解釈し、シス テムに設定された記号で置き換えます。以前のバージョンでは数値表示フォーマットでシステムの地域設定は考慮されていませんでした。例えば "###,##0.00" フォーマットは日本語システムでは有効ですが、フランス語システムでは結果が異なっていました。変換されたデータベースでは互換性保持のためこの新しいメカニズムが無効になっています。
- **Web変数に値を自動的に代入する**: 以前のバージョンの4Dでは、Webサーバーの標準機構によって、HTTPフォームやGET type URL を使用して送信された変数の値を自動的に4Dプロセス変数へと代入をしていました。インタープリタモードでは、変数同士が同じ名前であれば、受け取った変数の値はどんな値でも4Dプロセス変数へとコピーされました。コンパイルモードでは、変数は事前に*COMPILER_WEB* プロジェクトメソッドを使用して宣言しておく必要がありました。
v13.4 以降この機構は廃止予定となり、新しいデータベースでは使用できなくなりました。変換されたデータベースではこの機能は互換性のために、残されてはいますが、互換性のオプションでチェックを外すことによって無効化することができます。今後は代わりに **WEB GET VARIABLES** または **WEB GET BODY PART** コマンドの使用が推奨されます。
- **旧式ネットワークレイヤーを使用する(OS X用64-bit版では使用不可)**: v14 R5以降、4Dアプリケーションは 4D Server とリモートの 4Dマシン間の通信に、*ServerNet* という新しいネットワークレイヤーを使い始めました。以前のネットワークレイヤーは廃止予定となりますが、既存のデータベースとの互換性を保つために保持は されます。このオプションを使用すると、変換された4D Serverアプリケーションにおいて、必要に応じていつでも以前のネットワークレイヤーを有効化・無効化することができます。例えば、クライアントアプリケーションを移行させるとき(**設定 (環境設定)** の章を参照して下さい)などに使えます。*ServerNet* は新規に作成されたデータベースにおいては自動的に使用され、変換されたデータベースにおいては無効化されます(このオプションがチェックされます)。
この設定を変更する際には、その変更が反映されるためにはアプリケーションを再起動する必要があるという点に注意して下さい。接続していたクライアントアプリケーションも新しいネットワークレイヤーで接続するためには全て再起動しなければなりません(*ServerNet* を使用するために必要な最小限のクライアントのバージョンはあ4D v14 R4です。**設定 (環境設定)** の章を参照して下さい)。

注:

- このオプションは、**SET DATABASE PARAMETER** コマンドを使うことによってプログラミングによって管理することもできます。
- タイトルにあるように、このオプションはOS X用4D Server 64-bit版においては無視されます。このプラットフォームでは*ServerNet* のみが使用できます。

- **メソッドをUnicodeで保存:** このオプションを使用すると、4Dメソッドコード文字列をUnicodeで保存できるようになります。4D v15以前のバージョンでは、4Dメソッドコード文字列(式、変数、メソッド名、コメント、等)はカレントのローカルエンコーディングを使用して保存されていました。このエンコーディングは特に4Dコードが異なる国のデベロッパ間で共有されている場合に問題を引き起こす可能性がありました。例えば、フラン スのデベロッパがアクセントを含む4Dコードを書き、イギリスのデベロッパへとデータベースを送った場合、このアクセントは失われていました。コードが日本語版で書かれた場合にも、深刻な問題を引き起こす可能性がありました。メソッドをUnicodeとして保存することにより、こういった問題を全て解決 し、4Dコードを特定のローカルな文字を含んだまま交換することを可能にします。既存のデータベースにおいても、可及的速やかにメソッドに対して Unicodeオプションを有効化することが推奨されます(国際的な環境で仕事をしているのならなおさらです)。

  注:

  - この機能はランゲージそのものとその解釈に対して適用されます。一部の4Dエディターウィンドウ (プロパティリスト等) では引き続きローカルなエンコーディングを使用するため、一部の文字列が正確に表示されない可能性があります。しかしながら、これはコードの実行には関係しません。
  - このオプションを変更した場合、その変更が適用されるためにはアプリケーションを再起動する必要があります。このオプションはいつでもチェックをしたり外したりすることができます。変更後に保存したメソッドのみが影響を受けます。

- **アプリケーション配布では新しいアーキテクチャーを使用:** このオプションは4D v15 R4以降のアプリケーション全てで利用可能です。これは4Dアプリケーションの配布に関連した新しい機構を有効化または無効化します(これは最終アプリケーションを生成するマシン上で設定する必要があります)。このオプションで管理される機構については、**最後に開かれたデータファイル** と **クライアントアプリケーションによる接続の管理** の章で説明されています。このオプションは変換されたアプリケーションではデフォルトではチェックが外されています。この新しい機構を利用するためには、このオプションを明示的にチェックする必要があります。

振る舞いの変更

ライセンス

4D v16 では 4D 製品のライセンス管理がより簡単になりました:

初回アクティベーションの簡素化: "**ライセンスマネージャー**" ダイアログボックスに新しい 4D Server のライセンス番号を入力すると、一回の操作でそのサーバーライセンスにリンクしているすべてのエクスパンション (追加クライアント、プラグイン等) も自動でアクティベーションされます。

新しい **更新** ボタン: "**ライセンスマネージャー**" ダイアログボックスに新しく更新ボタンが追加されました**:**

このボタンをクリックして、4D のログイン情報 (アカウントとパスワード) を入力すると、カスタマーデータベースに接続して自動で利用中のライセンスを更新します (利用中のライセンスは "**有効なライセンス**" タブに太字で表示されています)。この**更新**ボタンは次のような場合に使用します:

- 追加購入したエクスパンションのアクティベーションを行いたい
- 延長した期限切れライセンスを更新したい

新しい自動アクティベーション機能: 特定の場合に、製品のアクティベーションを促す機構がトリガーされます。具体的には、次の場合が対象となります:

- アクティベーションされていない 4D Developer Edition を使って、インタープリターモードのローカルデータベースを開く、または新規作成すると、自動アクティベーション機構が作動します。ダイアログボックスが表示され、4Dのカスタマーデータベースに接続してライセンスのアクティベーションを行うことを知らせます (ご利用の 4Dアカウントのパスワードを入力する必要があります)。
- 4D Server アプリケーションを起動した場合、サーバーライセンスの自動更新が行われます。アプリケーションをサービスとして運用することを可能にするため、サーバーライセンスの自動更新処理はダイアログボックスなどの表示もなく完全に自動で行われ、ユーザーから見えることはありません。

ランゲージ

OBJECT SET FORMAT / **OBJECT Get format**: これらのコマンドはリストボックスヘッダーのアイコンをサポートするようになりました**。**

METHOD GET CODE: このコマンドはコードをインデント付きテキストを返すようになりました。

DELETE FOLDER: このコマンドは空でないフォルダーも削除できるようになりました。

フォント

FONT LIST コマンドは Windows上において、スケーラブルなフォントのみを返すようにアップデートされました。

印刷

64-bit版のみ: 本章で紹介する新機能は 4D v16 64-bit版 (4D Developer Edition および 4D Volume Desktop、詳細については **印刷アーキテクチャー(新デザイン)** を参照ください) でのみ提供されています。

64-bit版の4Dでは印刷アーキテクチャーが完全に書き換えられ、これにより最新のOSに基づいた印刷ライブラリとダイアログボックスの恩恵を受けられるようになりました。この内部的なアップデートは大部分において4Dユーザーからは透過的ですが、以下の変更については注意する必要があります:

- "**印刷ジョブ**" ダイアログボックス (WindowsとOS X) はアップグレードされ、両プラットフォームにおいて標準のシステムダイアログボックスとなりました。
- "**ページ設定**" ダイアログボックス (Windows) がアップデートされ、現在は OSにより提供されます。
- **PRINT SETTINGS** コマンド: 印刷コマンドを呼び出したときに、"**ページ設定**" ダイアログボックスが自動的には表示されなくなりました。ダイアログを表示させるには、*dialType* パラメーターに Page setup dialog 定数を指定する必要があります。
  また、このコマンドには二つ目の定数が追加されました: Print dialog によって印刷ダイアログボックスを表示するかどうかを指定することができます。

- 以下の印刷オプション (**GET PRINT OPTION** あるいは **SET PRINT OPTION** コマンドで使用) が変更されました:

オプション(定数)	OS	4D v16での状況	コメント
2 (Orientation option)	Windows と OS X	アップデート済み	印刷ジョブ内から呼び出し可能、つまり同じ印刷ジョブにおいて縦向きと横向きを切り替えることができます。
8 (Color option)	Windows のみ	削除済み	廃止されました。
13 (Mac spool file format option)	OS X のみ	削除済み	**SET CURRENT PRINTER** コマンドの新しいオプションに置き換えられました

。

注: **OPEN PRINTING JOB**、 **CLOSE PRINTING JOB**、 **SET PRINT OPTION**、および **SET PRINT OPTION** コマンドは 4D Write Pro の **WP PRINT** コマンドと互換性があります (詳細については、**WP PRINT** を参照ください)。Paper option と Orientation option 以外のすべてのオプションが、**WP PRINT** によって印刷される 4D Write Proドキュメントに対してサポートされています。Paper option と Orientation option については、**WP USE PAGE SETUP** を利用してページサイズと向きの属性を設定することを推奨しています。

リストボックス

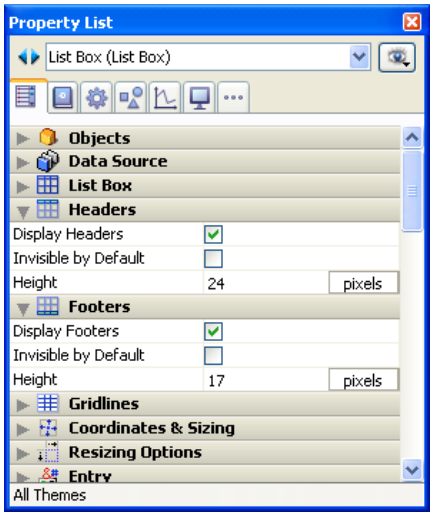
行管理配列

新しい **行管理配列** プロパティを使って、次のインターフェースプロパティを管理することができます:

- "表示" または "非表示" (デフォルトは "表示")
- "有効化" または "無効化" (デフォルトは "有効化")
- "選択可能" あるいは "選択不可" (デフォルトは "選択可能")

行管理配列プロパティは **LISTBOX SET ARRAY** と **LISTBOX Get array** を使用して設定あるいは読み込みをすることができます。またこの配列は、**LISTBOX GET ARRAYS** コマンドによっても返されます。
以前のバージョンの4Dでは、このプロパティは "**非表示行配列**" という名前でブール型の配列を受け取りました。互換性のため、行管理配列に対してもブール型の配列は利用可能です。この場合、それぞれの要素はリストボックス内で対応する行の表示/非表示ステータスを示します。**True**は行が非表示状態であり、**False**は行が表示状態であることを意味します。

ヘッダーとフッター



ピクセル単位でのヘッダーの最小高さはシステムにより異なります。最小値よりも小さな値を指定した場合、システムが定義するヘッダーの最小値で置き換えられます。行とフッターには最小値がありません。

Windows 7ではヘッダーの最小高さは 24ピクセルです。変換されたデータベースで高さがこれよりも小さな値に設定されていた場合、自動でリサイズされます。
ヘッダーやフッターの高さは **LISTBOX SET HEADERS HEIGHT** や **LISTBOX SET FOOTERS HEIGHT** コマンドを使用して動的に設定することもできます。

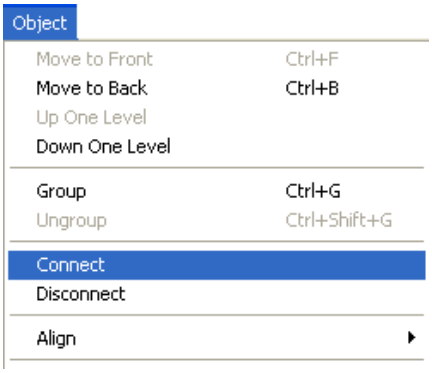
上述の影響でレイアウトが崩れてしまう場合がありますので、フォームを点検するときの確認事項として留意ください。

変換されたリストボックス

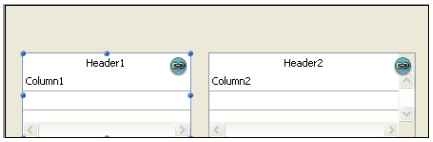
グループ化されたスクロールエリアがリストボックスに変換されると、それらは**接続**されます。接続されたリストボックスは連携して動作します:

- ひとつのリストボックス上で行を選択すると、接続されたグループに属するすべてのリストボックスの同じ行が選択されます。
- リストボックスをスクロールすると、接続されたグループに属するすべてのリストボックスがスクロールされます。

注: 変換されたリストボックスはフォーム上で**グループ化**されています。
フォームエディターの**オブジェクト**メニューから、接続・切断コマンドを使用して、これらのリストボックスの接続 / 切断を切り替えることができます:



これらのコマンドはフォームエディター上で適切なリストボックスが選択されている場合に有効となります。接続されたリストボックスが選択されている場合、そのリストボックスに接続されているすべてのリストボックス上に特別なバッジが表示されます：

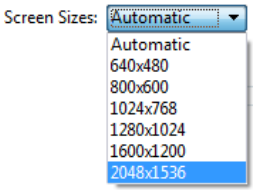


この動作原理を用いて、グループ化されたスクロールエリアの動作を再現します。しかしこの動作を標準のリストボックス機能に置き換えることが推奨されています。

フォーム

4D 製品およびハードウェアの進化に基づいて、フォームウィザードの詳細設定オプションが更新されました:

- **フォームサイズ** のリストに "2048x1536" が項目として追加されました:



- 生成されたフォームにおいて、ナビゲーションボタンの**変数名**プロパティは空欄のままになっています。

Replace string の最適化

4D v15 R3 で採用された新しい内部アルゴリズムにより、 **Replace string** コマンドを使用して特定の文字列を異なる長さの文字列で置換する際の実行速度がとても速くなりました。これは以下のような置換を行う場合があてはまります:

```
vResult:=Replace string(Source_Text;"a";"aa") //文字で置き換え
vResult2:=Replace string(Source_Text2;"à";"aa";*) //文字コードで置き換え
```

新しいアルゴリズムはどちらのシンタックスにおいても最適化されています。ソースとなるテキストが長く、置換箇所が多いほど、この最適化の差は顕著に現れます。

私たちが行ったベンチマークでは、以前のアルゴリズムと比較して以下のような結果を得ました:

文字コードによる置き換え(* を渡した場合)	文字による置き換え(* 省略時)
950倍の速さ	4400倍の速さ

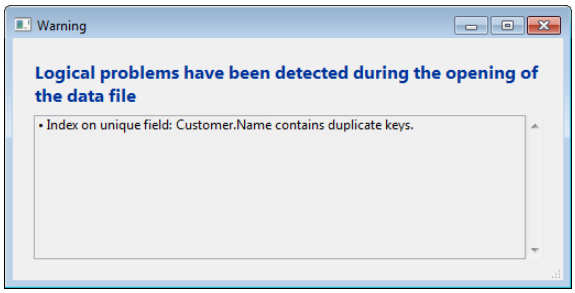
このテストはファイル内において32,000箇所の "a" を "aa" に置き換えるという内容で行われました。

注: 同じ長さの文字列で置換する場合は、以前のアルゴリズムと同じ速度になります。

重複不可フィールドにおける重複

重複不可フィールドにおいて値の重複が検知された際に、より詳細な情報が提供されます:

- MSC、または **VERIFY DATA FILE** のようなコマンドを使用すると、それによって生成されたログには違反しているテーブルとフィールド、そして重複している値が表示されます。
- データ入力時、"重複キー" エラーダイアログボックスにも違反しているテーブルとフィールド、そして重複している値が表示されます。
- **GET LAST ERROR STACK** コマンドは、あらゆる重複についての詳細な情報を含みます。
- 4D がデータファイルを開く際にインデックスを構築 (あるいは再構築) する必要がある場合、重複不可に設定されている関連フィールドでの重複が自動で検知されます。この場合、データベースを開く前に特別な警告ダイアログボックスが表示され、重複した値を特定・削除するために必要な情報をユーザーに提供します:



WEB: 4D タグと小数点

4Dタグ (**4DTEXT**、**4DVAR**、**4DHTML**、**4DHTMLVAR**、そして **4DEVAL**) を使用した数値表現を評価する際、4Dは常にピリオド文字(.)を小数点に使用するようになりました。このコンテキストにおいてはリージョン設定が無視されます。

この機能により、4Dの言語設定とバージョンが異なっていてもメンテナンスが用意となり互換性が保たれます。

例えば、以下のコードはリージョン設定に関わらず使用可能です:

```
value:=10/4
input:="<!--#4DTEXT value-->"
PROCESS 4D TAGS(input;output)
 // 例え小数点が ',' に設定されていた場合でも、出力は常に2.5になります。
```

使用コードにおいて、リージョン設定を遵守したうえで4Dタグの数値表現を評価するには、**String** コマンドを使って修正する必要があります:

- ピリオドを小数点として使用した*value*を取得する場合: *<!--#4DTEXT value-->*
- リージョン設定に基づく小数点を使用した*value*を取得する場合: *<!--#4DTEXT String(value)-->*

詳細については **4D 変換タグ** を参照ください。

HTTP サーバー

HTTP TRACE メソッドは 4D Webサーバーにおいてデフォルトでは無効化されています。HTTP TRACE メソッドを有効化する必要がある場合には、**WEB SET OPTION** コマンドに対して Web HTTP TRACE オプションを使用します。

タイムスタンプ付きのメンテナンスログファイル

MSCまたは4D Server管理ウィンドウを通して行うメンテナンスオペレーションがログファイルを生成するとき、その名前にはタイムスタンプがつくようになりました。以前のバージョンでは、ログファイルは常に同じファイル名で保存されていたため、既存のログファイルがあれば、それは新しいもので上書きされていましたが、今後ログファイルは生成されるたびに固有の名前でディスクに保存されます。このため、4Dにおいても4D Serverにおいても、データベース管理者が責任を持って、古いログファイルを必要に応じて削除する必要があります。

4D Internet Commands

バージョン v15.x/v15Rx と v16 の変更点: エンコーディングと文字セットの扱い (特にメール送信時の添付ファイル名について) が変わりました。このため、**これらの変更が動作に影響しないことを確認する必要があります。**

2つのコマンドが更新されました:

- **SMTP_Charset**
  添付ファイル名は base64 でエンコードされます。
  - 引数として 0 を受け渡すと、デフォルト値が使用されます:

encodeHeaders パラメーター: 題名は "UTF-8"、その他のフィールドは "ISO-8859-1"
bodyCharset パラメーター: base64 でエンコードされた UTF-8 文字セット
- 引数として 1 を受け渡すと、SMTP_SetPrefs コマンドで指定した値が使用されます。

- **SMTP_SetPrefs**
charset&Encoding に改名された二つめのパラメーターで、送信するメッセージ本文、ヘッダー、および添付ファイル名で使用される文字セットに加えて、メッセージ本文に適用されるエンコーディングを以下の表の値に応じて指定します:

値	本文の文字セット & エンコーディング	ヘッダーおよび添付ファイル名の文字セット (エンコーディングは常にbase64)
-1	変更しない	変更しない
0	Application & binary; エンコーディングなし	ISO-8859-1
1	デフォルト: UTF-8 & base64	デフォルト: 題名は UTF-8、その他のフィールドは ISO-8859-1
2	US-ASCII & 7bit	ISO-8859-1
3	US-ASCII & quotable-printable	ISO-8859-1
4	US-ASCII & base 64	ISO-8859-1
5	ISO-8859-1 & quotable-printable	ISO-8859-1
6	ISO-8859-1 & base64	ISO-8859-1
7	ISO-8859-1 & 8bit	ISO-8859-1
8	ISO-8859-1 & binary	ISO-8859-1
9	Reserved	Reserved
10	ISO-2022-JP (Japanese) & 7bit	ISO-2022-JP
11	ISO-2022-KR (Korean) & 7 bits	ISO-2022-KR
12	ISO-2022-CN (Traditional & Simplified Chinese) & 7 bit	ISO-2022-CN
13	HZ-GB-2312 (Simplified Chinese) & 7 bit	HZ-GB-2312
14	Shift-JIS (Japanese) & base64	Shift-JIS
15	UTF-8 & quoted-printable	UTF-8
16	UTF-8 & base64	UTF-8

OBJECT SET FORMAT

OBJECT SET FORMAT ({* ;} object ; displayFormat)

引数	型		説明
*	演算子	⇒	指定時, Objectはオブジェクト名 (文字列) 省略時, Objectはフィールドまたは変数
object	フォームオブジェクト	⇒	オブジェクト名 (* 指定時), または フィールドまたは変数 (* 省略時)
displayFormat	文字	⇒	オブジェクトに設定する表示フォーマット

説明

OBJECT SET FORMAT は、*object*で指定したオブジェクトの表示フォーマットを*displayFormat*で渡したフォーマットに設定します。新しいフォーマットは現在の表示にのみ有効です。フォームには保存されません。

オプションの * 引数を指定した場合、*object*はオブジェクト名です (文字列) 。オプションの * 引数を省略すると、*object*はフィールドまたは変数です。この場合、文字列ではなくフィールドまたは変数参照 (フィールドまたは変数のみ) を指定します。オブジェクト名に関する詳細はを参照してください。

OBJECT SET FORMAT は入力および出力フォーム (表示または印刷) 両方で使用でき、(入力可/不可) フィールドや変数に適用できます。

オブジェクトのデータタイプに適応する表示フォーマットを使用しなければなりません。

ブール

ブールフィールドをフォーマットするには二つの方法があります:

- 一つの値を*displayFormat*に渡す。この場合、フィールドはチェックボックスとして表示され、指定した値がラベルになります。
- セミコロン (*;*) で区切った二つの値を*displayFormat*に渡す。この場合フィールドは2つのラジオボタンとして表示されます。

日付

日付フィールドや変数をフォーマットするには、**Char**(n)を*displayFormat*に渡します。このとき*n*は4Dにより提供される以下の定義済み定数のうちいずれかです:

定数	型	値	コメント
Blank if null date	倍長整数	100	0の代わりに""
Date RFC 1123	倍長整数	10	Fri, 10 Sep 2010 13:07:20 GMT
Internal date abbreviated	倍長整数	6	Dec 29, 2006
Internal date long	倍長整数	5	December 29, 2006
Internal date short	倍長整数	7	2006/12/29
Internal date short special	倍長整数	4	06/12/29 (しかし 1986/12/29 または 2096/12/29)
ISO Date	倍長整数	8	2006-12-29T00:00:00
ISO Date GMT	倍長整数	9	2010-09-13T16:11:53Z
System date abbreviated	倍長整数	2	
System date long	倍長整数	3	
System date short	倍長整数	1	

Note: Blank if null は他の定数に加算されなければなりません。この定数は日付がヌル値の時、00/00/00ではなく空のエリアとして表示するよう4Dに指示します。

時間

時間フィールドや変数をフォーマットするには、**Char**(n)を*displayFormat*に渡します。このとき*n*は4Dにより提供される以下の定義済み定数のうちいずれかです:

定数	型	値	コメント
Blank if null time	倍長整数	100	0の代わりに""
HH MM	倍長整数	2	01:02
HH MM AM PM	倍長整数	5	1:02 AM
HH MM SS	倍長整数	1	01:02:03
Hour min	倍長整数	4	1時2分
Hour min sec	倍長整数	3	1時2分3秒
ISO time	倍長整数	8	0000-00-00T01:02:03
Min sec	倍長整数	7	62分3秒
MM SS	倍長整数	6	62:03
System time long	倍長整数	11	1:02:03 AM HNEC (Macのみ)
System time long abbreviated	倍長整数	10	1:02:03 AM (Macのみ)
System time short	倍長整数	9	01:02:03

Note: Blank if null は他の定数に加算されなければなりません。この定数は日付がヌル値の時、00:00:00ではなく空のエリアとして表示するよう4Dに指示します。

ピクチャ

ピクチャフィールドや変数をフォーマットするには、**Char**(n)を*displayFormat*に渡します。このとき*n*は4Dにより提供される以下の定義済み定数のうちいずれかです:

定数	型	値
On background	倍長整数	3
Replicated	倍長整数	7
Scaled to fit	倍長整数	2
Scaled to fit prop centered	倍長整数	6
Scaled to fit proportional	倍長整数	5
Truncated centered	倍長整数	1
Truncated non centered	倍長整数	4

文字と数値

文字や数値のフィールドや変数をフォーマットするには、*displayFormat* 引数に直接フォーマットラベルを渡します。

表示フォーマットに関する詳細は4D Design Referenceマニュアルの**数値フォーマット**や**文字フォーマット**を参照してください。

Note: ツールボックスであらかじめ定義した表示フォーマットを*displayFormat*に使用するには、表示フォーマット名の前に縦棒(|)を挿入します。

ピクチャボタン

ピクチャボタンをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
cols;lines;picture;flags{;ticks}

- *cols* = ピクチャの列数
- *lines* = ピクチャの行数
- *picture* = 使用するピクチャ (ピクチャライブラリ、ピクチャ変数):
  - ピクチャライブラリのピクチャを使用する場合、クエスチョンマークの後にその番号を指定します (例: "?250")。
  - ピクチャ変数ののピクチャを使用する場合、変数名を指定します。
- *flags* = ピクチャボタンの表示モードと処理。この引数には以下の値を指定できます: 0, 1, 2, 16, 32, 64, 128。これらの値はそれぞれ表示モードと処理モードを表します。これらの値は累計することができます。例えばモード1と64を有効にするには、65を*flags* 引数に渡します。それぞれの値の意味は以下のとおりです:
  - *flags* = 0 (オプションなし)
    ユーザがピクチャをクリックすると、順番に次のピクチャを表示します。Shiftキーを押しながらクリックすると、前のピクチャを順に表示します。最後のピクチャに達すると、クリックしてもピクチャは変更されません。つまり最初のピクチャには戻りません。
  - *flags* = 1 (連続してスイッチ)
    前のオプションと同様ですが、ユーザがマウスを押したままにするとピクチャは連続して (アニメーションのように) 変更されます。最後のピクチャに到達すると、そこでピクチャの変更は停止します。
  - *flags* = 2 (最初のフレームにループバック)
    前のオプションと同様ですが、ピクチャが連続したループで表示される点が異なります。最後のピクチャに達っ

し、さらにクリックすると、最初のピクチャが表示されます。
    - *flags* = 16 (ロールオーバー時にスイッチ)
      ピ クチャボタンのコンテンツは、マウスカーソルがボタンの上に来たときに変更されます。マウスがボタンエリアから離れると、最初のピクチャに戻ります。この モードはマルチメディアアプリケーションやHTMLドキュメントでしばしば使用されます。ロールオーバー時に表示されるピクチャはサムネールテーブルの最 後のピクチャです。ただし最後のフレームを無効として使用オプション (128) を使用した場合は、最後の前のフレームがロールオーバー時に使用されます。
    - *flags* = 32 (リリース後に元に戻す)
      このモードは2つのピクチャで動作します。これはユーザがクリックしたときを除き、常に最初のピクチャを表示します。この場合2番目のピクチャがマウスク リックの間表示され、マウスがリリースされると一番目のピクチャに戻ります。このモードを使用すれば、アイドルとクリック状態を表示するアクションボタン を作成できます。このモードを3D効果を作成したり、アクションを表現するピクチャを表示するために使用できます。
    - *flags* = 64 (透過)
      バックグランドピクチャを透過させるために使用します。
    - *flags* = 128 (最後のフレームを無効として使用)
      このモードは、最後のサムネールフレームをボタンが無効時に表示させるために使用します。このモードが選択されているとき、4Dは最後のサムネールを、ボタ ンが無効にされているときに表示します。このモードが、モード 0, 1 および 2とともに使用されていると、最後のサムネールは一連の表示には組み込まれません。子のサムネールはボタンが無効のときにのみ表示されます。
- *ticks* = “n チック毎に表示”モードを有効にし、それぞれのピクチャを表示する間隔を設定します。このオプション引数が渡されると、指定された速度でピクチャボタンの コンテンツが繰り返し表示されます。例えば"2;3;?16807;0;10"と指定すると、ピクチャボタンは10tickごとに異なるピクチャを表示し ます。このモードが有効の時は透過モード (64) のみが使用できます。

ピクチャポップアップメニュー

ピクチャポップアップメニューをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
cols;lines;picture;hMargin;vMargin;flags

- *cols* = ピクチャの列数
- *lines* = ピクチャの行数
- *picture* = 使用するピクチャ (ピクチャライブラリ、ピクチャ変数):
    - ピクチャライブラリのピクチャを使用する場合、クエスチョンマークの後にその番号を指定します (例: "?250")。
    - ピクチャ変数ののピクチャを使用する場合、変数名を指定します。
- *hMargin* = メニューの水平境界とピクチャの間のマージン (ピクセル)
- *vMargin* = メニューの垂直境界とピクチャの間のマージン (ピクセル)
- *flags* = ピクチャポップアップメニューの透過モード (0 または 64):
    - *mode* = 0: ピクチャポップアップメニューは透過でない
    - *mode* = 64: ピクチャポップアップメニューは透過

サーモメーターおよびルーラー

サーモメータやルーラーをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
min;max;unit;step;flags{;format{;display}}

- *min* = インジケータの最初の目盛り値
- *max* = インジケータの最後の目盛り値
- *unit* = インジケータの目盛りの間隔
- *step* = インジケータ中でカーソル移動の最小間隔
- *flags* = インジケータの表示モードと動作。この引数は0, 2, 3, 16, 32, 128を受け入れます。これらの値は128を除き、加算して複数のオプションを設定できます:
    - *flags* = 0: 単位を表示しない
    - *flags* = 2: インジケータの右または下に単位を表示
    - *flags* = 3: インジケータの左または上に単位を表示
    - *flags* = 16: 単位に隣接して目盛りを表示
    - *flags* = 32: ユーザがインジケータを調整している間、On Data Changeを実行する。この値が使用されない場合、On Data Changeはユーザがインジケータの調整を終了したときにのみ発生します。
    - *flags* = 128: "バーバーショップ" (連続したアニメーション) モードを有効にします。この値をほかの値と一緒に使用することはできません。このモードでは、他の引数は無視されます (*display*引数が渡された場合を除く)。このモードに関する詳細は、 *Design Reference*マニュアルを参照してください。
- *format* = インジケータの目盛りの表示フォーマット

インジケータオブジェクトのサイズが小さいため単位やメモリが正しく表示できない場合、それらは隠されます。
- *display* = 特別な表示オプション。サーモメーターの場合、この引数は*flags*サブ引数が128のときにのみ考慮されます。
  - *display* = 0 (または省略時): 標準のルーラを表示 / "バーバーショップ"タイプの連続したアニメーションを表示。
  - *display* = 1 : ルーラーの"ステッパー"モードを有効にする / サーモメーターの"非同期進捗"モードを有効にする。これらのオプションに関する詳細は*Design Reference*マニュアルを参照してください。

ダイアル

ダイアルをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
min;max;unit;step{;flags}

- *min* = インジケータの最初の目盛り値
- *max* = インジケータの最後の目盛り値
- *unit* = インジケータの目盛りの間隔
- *step* = インジケータ中でカーソル移動の最小間隔
- *flags* = ダイアルの処理モード (オプション)。この引数は32鑿を受け入れます: ユーザがインジケータを調整している間、On Data Changeを実行する。この値が使用されない場合、On Data Changeはユーザがインジケータの調整を終了したときにのみ発生します。

ボタングリッド

ボタングリッドをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
cols;lines

- *cols* = グリッドの列数
- *lines* = グリッドの行数

Note: フォームオブジェクトの表示フォーマットに関する詳細は、4D Design Referenceマニュアルを参照してください。

3D buttons

3Dボタンをフォーマットするには、*displayFormat* 引数に以下のシンタックスを使用した文字列を渡します:
title;picture;background;titlePos;titleVisible;iconVisible;style;horMargin;vertMargin;iconOffset;popupMenu;hyperlink;numStates iconOffset;popupMenu

- *title* = ボタンタイトル。この値はテキストまたはリソース番号 (例: ":16800,1") で指定できます。
- *picture* = ボタンにリンクするピクチャ。ピクチャライブラリ、ピクチャ変数、またはResourcesフォルダのピクチャを使用できます:
  - ピクチャライブラリのピクチャを使用する場合、クエスチョンマークの後にその番号を指定します (例: "?250")。
  - ピクチャ変数ののピクチャを使用する場合、変数名を指定します。
  - データベースのResourcesフォルダのピクチャを使用する場合、"#{folder/}picturename" または "file:{folder/}picturename"タイプのURIを指定します。
- *background* = ボタンにリンクするバックグランドピクチャ (カスタムスタイル)。ピクチャライブラリ、ピクチャ変数、PICTリソース、Resourcesフォルダのファイル (上記参照) を使用できます。
- *titlePos* = ボタンタイトルの位置:
  - *titlePos* = 1: 左
  - *titlePos* = 2: 上
  - *titlePos* = 3: 右
  - *titlePos* = 4: 下
  - *titlePos* = 5: 中央
- *titleVisible* = タイトルの表示/非表示:
  - *titleVisible* = 0: タイトルを非表示
  - *titleVisible* = 1: タイトルを表示
- *iconVisible* = アイコンの表示/非表示:
  - *iconVisible* = 0 : アイコンを非表示
  - *iconVisible* = 1 : アイコンを表示
- *style* = ボタンスタイル。このオプションで指定したスタイルにより、バックグランドなど他のオプションが有効になるかどうかが決まります。以下の値が使用できます:
  - *style* = 0: なし
  - *style* = 1: バックグランドオフセット
  - *style* = 2: プッシュボタン
  - *style* = 3: ツールバーボタン

    - *style* = 4: カスタム
    - *style* = 5: サークル
    - *style* = 6: システムスクエア (小)
    - *style* = 7: Office XP
    - *style* = 8: ベベル
    - *style* = 9: 角の丸いベベル
    - *style* = 10: 折り畳む/展開
    - *style* = 11: ヘルプ
    - *style* = 12: OS X テクスチャー
    - *style* = 13: OS X グラデーション
- *horMargin* = 水平マージン。ボタン内部の左右マージン (アイコンやテキストが描画されないエリア) をピクセル単位で指定します。
- *vertMargin* = 垂直マージン。ボタン内部の上下マージン (アイコンやテキストが描画されないエリア) をピクセル単位で指定します。
- *iconOffset* = 右および下方向へのアイコンのシフト。ピクセル単位で指定されるこの値は、ボタンがクリックされた際のボタンアイコンの右下方向へのシフトを指定します (同じ値が両方向に使用されます)。
- *popupMenu* = ボタンへのポップアップメニューの関連付け:
    - *popupMenu* = 0: ポップアップメニューなし
    - *popupMenu* = 1: リンクしたポップアップメニュー
    - *popupMenu* = 2: 分離したポップアップメニュー
- *hyperlink* = タイトルはマウスオーバー時にハイパーリンクである事を強調するために下線が付けられます(廃止予定の機構)。二つの値が使用可能です:
    - *hyperlink* = 0: タイトルはマウスオーバー時でも下線がつかない
    - *hyperlink* = 1: タイトルはマウスオーバー時に下線がつく
- *numStates* = 3Dボタンのアイコンとして使用しているピクチャー内に存在する状態の数。この数字はまた、標準のボタンの状態を表示するために4Dによって使用されます。

いくつかのオプションは、すべての3Dボタンで有効というわけではありません。また特定のオプションを設定したくない場合もあるでしょう。あるオプションを渡さないようにするには、対応する値を省略します。例えば*titleVisible*と*vertMargin*オプションを省略するには、以下のように書きます:

```
OBJECT SET FORMAT(myVar;"NiceButton;?256;:562;1;;1;4;5;;5;0;;2")
```

リストボックスヘッダー

リストボックス内でのアイコンをフォーマットする場合、以下のシンタックスに従う文字列を*displayFormat*引数に渡します:
picture;iconPos

- *picture* = ヘッダーのピクチャー。ピクチャーライブラリー、ピクチャー変数、ピクチャーファイルから取得可能:
    - ピクチャーライブラリーから取得する場合、"?"に続けてその番号を入力します(例: "?250")。
    - ピクチャー変数から取得する場合、その変数名を入力します。
    - データベースのResourcesフォルダーから取得する場合、そのURLを"#{folder/}picturename" あるいは "file:{folder/}picturename"という形式で入力します。
- *iconPos* = ヘッダー内でのアイコンの位置。二つの値がサポートされます:
    - *iconPos* = 1: 左
    - *iconPos* = 2: 右

この機能は例えば、カスタマイズされた並び替えアイコンを使用したい場合などに有効です。

例題 1

以下のコードは[Employee]DateHired フィールドをInternal date longにフォーマットします。

```
OBJECT SET FORMAT([Employee]DateHired;Char(Internal date long))
```

例題 2

以下のコードは *[Company]ZIP Code* フィールドのフォーマットを、フィールドデータ長に基づいて変更します:

```
If(Length([Company]ZIP Code)=9)
```

```
   OBJECT SET FORMAT([Company]ZIP Code;"#####-####")
Else
   OBJECT SET FORMAT([Company]ZIP Code;"#####")
End if
```

例題 3

以下のコードは *[Stats]Results* フィールド値を、値の正負およびゼロであるかに応じてフォーマットします:

```
OBJECT SET FORMAT([Stats]Results;"### ##0.00;(### ##0.00);")
```

例題 4

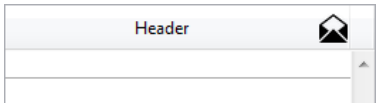
以下のコードはブールフィールドのフォーマットを変更して、Married または Unmarriedが表示されるようにします:

```
OBJECT SET FORMAT([Employee]Marital Status;"Married;Unmarried")
```

例題 5

データベースのResourcesフォルダーに"envelope_open.png"という名前のピクチャーファイルを保存してあった場合、以下のように書く事ができます:

```
vIcon:="#envelope_open.png"
vPos:="2" // 位置は右
OBJECT SET FORMAT(*;"Header1";vIcon+";"+vPos)
```



例題 6

以下のコードはブールフィールドのフォーマットを変更して、"Classified"というラベルのチェックボックスが表示されるようにします:

```
SET FORMAT([Folder]Classification;"Classified")
```

例題 7

1行4列のサムネイルがあります。このサムネイルをピクチャボタンに使用します (“デフォルト”, “クリック”, “ロールローバー” そして “無効時”)。ロールオーバー時にスイッチとリリース後に元に戻す、そして最後のフレームを無効として使用オプションを有効にします:

```
OBJECT SET FORMAT(*;"Picture Button";"4;1;?15000;176")
```

例題 8

サーモメータをバーバーショップモードにします:

```
OBJECT SET FORMAT($Mythermo;";;;;128")
$Mythermo:=1 `Start animation
```

METHOD GET CODE

METHOD GET CODE (path ; code {; option} {; *})

引数	型		説明
path	テキスト, テキスト配列	➡	メソッドパスを格納したテキストまたはテキスト配列
code	テキスト, テキスト配列	⬅	指定したメソッドのコード
option	倍長整数	➡	0 または省略時 = 単純な書き出し(トークンなし)、1 = トークンを使用して書き出し
*	演算子	➡	指定時 = コンポーネントで実行されたとき、コマンドはホストデータベースに適用される (コンポーネントのコンテキスト以外ではこの引数は無視されます)

説明

METHOD GET CODEコマンドは*path*引数で指定したメソッドの内容を*code*に返します。このコマンドはデータベースメソッド、トリガー、プロジェクトメソッド、フォームメソッド、そしてオブジェクトメソッド等すべてのタイプのメソッドコードを返すことができます。

テキスト配列またはテキスト変数に基づく2つのシンタックスを使用できます:

```
C_TEXT(tVpath) // テキスト変数
C_TEXT(tVcode)
METHOD GET CODE(tVpath;tVcode) // 1つのメソッドのコード
```

```
ARRAY TEXT(arrPaths;0) // テキスト配列
ARRAY TEXT(arrCodes;0)
METHOD GET CODE(arrPaths;arrCodes) // 複数メソッドのコード
```

2つのシンタックスを混合して使用することはできません。

無効なパス名を渡すと、*code* 引数は空のままエラーが生成されます。

このコマンドから*code*に返されるテキストは以下のとおりです:

- 4Dコマンド名はフランス語バージョンで"リージョンシステム設定を使用"オプションをチェックしていた場合除き、全てのバージョンの4D英語で記述されます(**メソッドページ** を参照して下さい)。*option*引数を使用する場合、コードにランゲージトークンを含める事によって4Dプログラミング言語とバージョンに関わりなくすることもできます(以下を参照)。
- コードの可読性を高めるために、メソッドエディター同様、テキストはプログラミング言語に基づいたタブ文字でインデント付けがされています。
- 読み込み時にメタデータが含まれていたコードには、先頭に行が追加されます。例えば:

  ```
  // %attributes = {"lang":"fr","invisible":true,"folder":"Web3"}
  ```

  読み込み時、この行は読み込まれず、それに対応する属性を設定する目的でのみ使用されます(指定されていない属性はデフォルト値へとリセットされます)。"lang"属性は書き出し言語を設定し、異なる言語への読み込みを防止する目的で使用されます(この場合、エラーが生成されます)。「フォルダ」属性にはメソッドの親フォルダの名前が入ります。メソッドが親フォルダを持っていない場合は表示されません。
  追加で属性を定義することができます。詳細な情報に関しては、**METHOD SET ATTRIBUTES** コマンドの詳細を参照して下さい。

*option*引数を使用する際、メソッドのトークナイズドされたランゲージ要素についてのコード書き出しモードを選択する事ができます:

- *option*引数に0を渡すか省略した場合、メソッドコードはトークンなしで書き出されます。つまり、メソッドエディターに表示されているのと同じように書き出されます。
- 1または定数Code with tokensを渡した場合、メソッドコードはトークンとともに書き出されます。つまり*code*引数に書き出されたコンテンツにおいて、トークナイズドされた要素はその直後に内部参照がつくようになります。例えば、"**String**(a)"という表記は"**String**:C10(a)"という形で書き出されます。ここで、"C10"とは**String**コマンドの内部参照を表しています。

トークナイズドされたランゲージ要素には以下のものが含まれます:

- 4Dコマンドと定数
- テーブルとフィールド名
- 4Dプラグインコマンド

トークンで書き出されたコードは以降のランゲージ要素のどのような改名にも影響されません。トークンのおかげで、**METHOD SET CODE**コマンドあるいはコピー/ペーストを用いた場合でも、テキストとして提供されたコードは4Dによって常に正常に解釈されます。4D トークンのシンタックスについての詳細は **フォーミュラ内でのトークンの使用**を参照ください。

コマンドがコンポーネントから実行されると、デフォルトでコンポーネントメソッドに適用されます。* 引数を渡すとホストデータベースにアクセスします。

例題 1

METHOD SET CODEコマンドの例題参照。

例題 2

この例では、*option*引数を使用した際の影響について説明します。

以下の"simple_init"メソッドを書き出したい場合を考えます:

```
Case of
   :(Form event=On Load)
      ALL RECORDS([Customer])
End case
```

以下のコードを実行した場合:

```
C_TEXT($code)
C_TEXT($contents)
$code:=METHOD Get path(Path project method;"simple_init")
METHOD GET CODE($code;$contents;0) //トークンなし
TEXT TO DOCUMENT("simple_init.txt";$contents)
```

書き出されたドキュメントには以下が含まれます:

```
   //%attributes = {"lang":"fr"} 4Dによってコメントが追加・保存
Case of
    : (Form event=On Load)
        ALL RECORDS([Customer])
End case
```

以下のコードを実行した場合:

```
C_TEXT($code)
C_TEXT($contents)
$code:=METHOD Get path(Path project method;"simple_init")
METHOD GET CODE($code;$contents;Code with tokens) //トークンを使用
TEXT TO DOCUMENT("simple_init.txt";$contents)
```

書き出されたドキュメントには以下が含まれます:

```
   //%attributes = {"lang":"fr"} 4Dによってコメントが追加・保存
Case of
    : (Form event:C388=On Load:K2:1)
        ALL RECORDS:C47([Customer:1])
End case
```

DELETE FOLDER

DELETE FOLDER (folder {; deleteOption})

引数	型		説明
folder	文字	➡	削除されるフォルダーの名称またはフルパス
deleteOption	倍長整数	➡	フォルダー削除オプション

説明

DELETE FOLDER コマンドは*folder* に渡したフルパスまたは名前を持つフォルダーを削除します。

deleteOption パラメーターを省略した場合のデフォルトでは、安全のため **DELETE FOLDER** は空のフォルダーのみ削除します。空でないフォルダーを削除するには *deleteOption* を使います。*deleteOption* には "**System Documents**" テーマの次の定数を受け渡すことができます:

定数	型	値	コメント
Delete only if empty	倍長整数	0	フォルダの中身が空の場合のみ削除する
Delete with contents	倍長整数	1	フォルダを中身ごと削除する

- Delete only if empty (0) を受け渡した場合、または *deleteOption* を省略した場合:
  - *folder* パラメーターに指定したフォルダーは、空の場合に限り削除されます。空でない場合には削除処理は行わず、エラー -47 (ファイルが既に開かれている、あるいはフォルダが空ではありません。) が発生します。
  - 指定したフォルダーがが存在しない場合にはエラー -120 (存在しないディレクトリを指定するパス名を使用してファイルにアクセスしようとしました。) が生成されます。

- Delete with contents (1) を受け渡した場合:
  - *folder* パラメーターに指定したフォルダーは、格納されている要素ごと削除されます。
    警告: フォルダーや格納要素がロックされていたり、読み取り専用に設定されていても、カレントユーザーが処理に必要な権限を持っていれば削除されます。
  - フォルダーや格納要素を削除できない場合、最初の削除不能な要素を検知した時点で処理は中断され、エラー* が返されます。この場合、フォルダーは一部削除済みの可能性があります。処理が中断された場合には、**GET LAST ERROR STACK** を使って、問題となったファイルの名称とパスを取得できます。
  - *folder* パラメーターに指定したフォルダーがが存在しない場合、このコマンドは処理を行わず、エラーも発生しません。

    (*) Windows: -54 (ロックされたファイルを書き込みのために開こうとしました。)
    OS X: -45 (ファイルがロックされている、あるいはパス名が不正です。)

これらのエラーは **ON ERR CALL** コマンドによって実装したメソッドでインターセプトすることができます。

FONT LIST

FONT LIST (fonts {; listType | *})

引数	型		説明
fonts	テキスト配列	⇐	フォント名の配列
listType | *	倍長整数, 演算子	⇒	取得したいフォント型のリスト、フォント名を取得するために*を指定(OS Xのみ)

説明

FONT LISTコマンドは、テキスト配列の*fonts*引数を作成し、システム上で使用可能なスケーラブルなフォントの名前を格納します。

listType 引数は取得したいフォントリストの型を指定します。指定するためには、 "**Font Type List**"テーマ内の以下の定数のいづれかを *listType* 引数に渡して下さい:

定数	型	値	コメント
Favorite fonts	倍長整数	1	*fonts* にはお気に入りのフォント(マシン上で最もよく使われているフォント)のリストが返されます。 - Windows 環境下では、Windowsコントロールパネル内のアクティブなフォントファミリー名のリストが表示されます。 - OS X 環境下では、コントロールパネル内にフォントファミリー名のリストが表示されます。英語では"Favorites" 、フランス語では "Favoris" 、ドイツ語では"Favoriten" というように名前がついています。このコレクションは、ユーザーがお気に入りのフォントを何も追加していない場合には空であることがあります。
Recent fonts	倍長整数	2	*fonts* には最近のフォント(4Dセッション中に使用されたフォント)のリストが含まれます。このリストは特にマルチスタイルテキストエリアによって使用されます。
System fonts	倍長整数	0	*fonts* 全てのシステムフォントのリストが含まれます。*listType* が省略されていた場合のデフォルトのオプションです。

OS X環境下では、任意の * 引数を渡す事によって *fonts* 配列を作成し、フォントファミリーの名前ではなくフォント自身の名前を入れます。デフォルトの操作ではリッチテキストエリアのプログラムの管理を簡略化し、フォントファミリーを使用します。つまり * 引数を渡すと、"Arial"、 "Arial black" や "Arial narrow" といったフォントファミリー名ではなく、 "Arial bold"、 "Arial italic"、 "Arial narrow italic" といったフォント名が返されるようになります。
Windows 環境下では、 * 引数は何の効力も持ちません。渡したとしてもコマンドはフォントファミリー名を返します。

注: Mac OS上で、このコマンドから返される結果を**ST SET ATTRIBUTES**コマンドで使用する場合、 * 引数を渡してはなりません。

スケーラブルフォントについて

このコマンドはスケーラブルなフォントのみを返します。スケーラブルでないフォント(ビットマップフォントなど)は古いテクノロジーに基づいており、サイズ変化における制約が問題となりうることから、デザインインターフェースでの使用は推奨されていません。4D Write Proエリアなどの4Dの最新機能ではこれらはサポートされていません。

OS X環境下では、OS X 10.4からこの原理が採用されています(このバージョンから*QuickDraw*ビットマップフォントが廃止予定となっています)。

Windows環境下では、この原理は4D v15 R4から採用されています。デベロッパーがインターフェースにおいて現代的なフォントのみを選択できるように、"trueType"または"openType"スケーラブルフォントのみが表示されています。例えば、"ASI_Mono"、"MS Sans Serif"、"System"フォントは表示されなくなっています。これに加えて、GDI名も表示されないようになり、DirectWriteフォントファミリー名前のみがサポートされるようになりました。例えば、"Arial Black"、"Segoe UI Black"フォントファミリーはリストには含まれず、"Arial"と"Segoe"のみが返されるようになります。

Windowsでの互換性上の注意:

- ビットマップフォントは、4Dフォームにおいて引き続きご利用いただけます(ただし4D Write Proエリアは除く)。このコマンドで返される一覧からは除外されているだけということです。しかしながら、将来のバージョンにおける4DとWindowsでの互換性の確保のためには、DirectWriteフォントファミリーを使用することが推奨されます。
- Windows上ではビットマップフォントは*fonts*引数からフィルターされているため、以前のリリースと比較すると4D v15 R4以降のアプリケーションで返されるリストは異なります。このコマンドを使用してスケーラブルでないホントを

選択していた場合には、必ずコードを修正するようにして下さい。

例題 1

フォーム上に、システム上で使用可能なフォントリストを表示するドロップダウンリストを作成したいとします。その場合、以下のようなドロップダウンリストのメソッドを記述します。

```
Case of
   :(Form event=On Load)
      ARRAY TEXT(asFont;0)
      FONT LIST(asFont)
   ` ...

End case
```

例題 2

最近使用したフォントのリストを取得したい場合:

```
FONT LIST($arrFonts;Recent fonts)
```

GET PRINT OPTION

GET PRINT OPTION (option ; value1 {; value2})

引数	型		説明
option	倍長整数	➡	オプション番号
value1	倍長整数, テキスト	⬅	オプションの値1
value2	倍長整数, テキスト	⬅	オプションの値2

説明

GET PRINT OPTION コマンドは、プリントオプションの現在の値を返します。

*option*引数を使用して取得するオプションを指定することができます。標準のオプション(倍長整数)か、PDFオプションコード(文字列)を取得することができます。コマンドは、*value1* と*value2* (任意)引数に、*option*引数で指定されたカレント値を入れて返します。

標準の印刷オプションを指定するには、“**Print Options**”テーマ内にある以下の定義済み定数を使用します:

定数	型	値	コメント
Paper option	倍長整数	1	*value1* のみを使用した場合、ここには用紙の名前のみが含まれます。両方の引数を使用した場合、*value1* には用紙の幅が、*value2* には用紙の高さが含まれます。幅と高さはどちらもスクリーンピクセルで表現されます。プリンターで使用できる全ての用紙フォーマットの名前、高さと幅を取得する場合には**PRINT OPTION VALUES** コマンドを使用して下さい。
Orientation option	倍長整数	2	*value1* のみ:1=縦向き、2=横向き。異なるページ方向が使用されている場合、**GET PRINT OPTION**コマンドは*value1*に0を返します。 **64-bit 版のみ**: このオプションは印刷ジョブ内から呼び出す事が可能なので、同一印刷ジョブ中において縦向きを横向きに、あるいはその逆へと切り替えることが可能です。
Scale option	倍長整数	3	*value1* のみ: 拡大縮小の倍率の値(パーセント)。一部のプリンターでは倍率の変更を許可していないものもあるという点に注意して下さい。無効な値を渡した場合、プロパティは印刷時に100%へとリセットされます。
Number of copies option	倍長整数	4	*value1* のみ: 印刷する部数
Paper source option	倍長整数	5	(Windows のみ) *value1* のみ: コマンドで返されるトレイの配列の中で、使用される予定の用紙トレイのインデックスに対応する番号。このオプションはWindowsでのみ使用可能です。
Color option	倍長整数	8	(Windows のみ) *value1* のみ: カラーを管理するモードを指定するコード: 1=白黒(モノクロ)、2=カラー **64-bit 版**: このオプションは64-bit版の4Dではサポートされていません。(廃止予定)
Destination option	倍長整数	9	*value1*: 印刷先のタイプを指定するコード: 1=プリンター、2=(PC)/PS ファイル(Mac)、3=PDFファイル、5=スクリーン(OS X ドライバーオプション)。 *value1* が1あるいは5以外であった場合、*value2* には生成されたドキュメントへのパス名が含まれます。このパスは他のパスが指定されるまでは使用され続けます。保存先に同じ名前のファイルが既に存在していた場合には、それは置き換えられます。**GET PRINT OPTION**の場合、カレントの値が既定のリスト内にない場合、*value1* には-1が返され、OKシステム変数は1に設定されます。エラーが起きた場合、*value1* とOKシステム変数は0に設定されます。 **注**: Windowsにおいては、PDF Creatorドライバーがインストールされていた場合には印刷先を3(PDFファイル)に設定することができます。*(9;3;path)* の値が渡された場合、4Dは自動的に"サイレント"PDF印刷を開始します。この場合には、渡されたオプションコードであればどれでも受け取ります(*value2* に空の文字列を渡すかこの引数を省略した場合、印刷時にファイルを保存ダイアログが表示されるという点に注意して下さい)。印刷後、カレントの設定は保存されます。これは4DのPDF印刷の管理を簡略化し、ユーザーがマルチプラットフォームなコードを書けるようにします。*(9;3;path)*値が渡されなかった場合、印刷は4Dによって管理されず、渡されたPDF Creatorオプションコードはどれも無視されます。
Double sided option	倍長整数	11	(Windows のみ) *value1*: 0=片側印刷あるいは標準、1=両面印刷。*value1*=1のとき、*value2* にはページ綴じの設定が含まれます: 0=左綴じ(デフォルト値)、1=上綴じ **注:** このオプションはWindows環境においてのみ使用可能です。
Spooler document name option	倍長整数	12	*value1* のみ: スプーラドキュメントの一覧に表示される、カレントの印刷ドキュメント名。この宣言によって定義される名前は、新しい名前あるいは空の文字列が渡されない限りはセッションで印刷される全てのドキュメントに対して使用されます。標準のオペレーション(メソッドの場合にはメソッド名を、レコードの場合にはテーブル名を使用)を使用あるいは復元するためには、空の文字列を*value1* に渡して下さい。
Mac spool file format option	倍長整数	13	(Mac のみ) *value1* のみ: 0=PDFモードでジョブを印刷(デフォルト値)　1=PostScriptモードでジョブを印刷 **注:** - このオプションはWinodws環境下では何の効力も持ちません。 - OS Xでは、印刷はデフォルトでPDFで行われます。しかしながら、PDF印刷ドライバは格納されたPostScript情報をもつPICTフォーマットのピクチャーをサポートしません。これらのピクチャーは具体的にはヴェクター式の描画ソフトウェアによって生成されます。このような問題を避けるため、このオプションではOS X環境下のカレントのセッションで使用するために、印刷モードを変更することができます。ただしPostScriptモードでの印刷には予期せぬ副作用を引き起こす可能性がある点に注意して下さい。

			64-bit 版: このオプションはサポートされていません。代わりに、**SET CURRENT PRINTER** コマンドのGeneric PDF driver オプションで置き換えられています。
Hide printing progress option	倍長整数	14	*value1* のみ: 1=進捗ウィンドウを非表示、0=進捗ウィンドウを表示(デフォルト)。このオプションは、特にOS XでのPDF印刷の際に有用です。 **注:** データベース設定ダイアログボックス内には、既に印刷プログレスオプションがあります(インターフェースページ内)。しかしながら、この設定は全体に適用され、OS X環境下でのウィンドウを全て非表示にするわけではありません。
Page range option	倍長整数	15	4D Write Pro 専用のオプション
Legacy printing layer option	倍長整数	16	(Windows用4D 64-bit版のみ) *value1* のみ: 1=以降の印刷ジョブに対してはGDI-ベースの旧式の印刷レイヤーを選択。0=D2D印刷レイヤーを選択(デフォルト)。 **64-bit 版:** このセレクターはWindows用64-bit版4Dのシングルユーザーアプリケーションでのみサポートされます。他のプラットフォームでは無視されます。これは主に64-bit版4Dアプリケーションの4Dジョブ内で旧式プラグインが印刷できるようにするためにものです。

PDFオプションコードは2つの部分、*OptionType*と*OptionName*からなり、"*OptionType:OptionName*"のように組み合わせて使用します。 PDFオプションコードとそれらの取り得る値についての詳細な情報については、**SET PRINT OPTION**コマンドの説明を参照して下さい。

注: GET PRINT OPTION コマンドは主にPostScript プリンターをサポートします。このコマンドは他のタイプのプリンター、例えばPCLやlinkなどにも使用できますが、その場合一部のオプションが使用できない可能性があります。

システム変数およびセット

このコマンドが正しく実行されると、OKシステム変数に1が、そうでなければ0が設定されます。

OPEN PRINTING JOB

OPEN PRINTING JOB
このコマンドは引数を必要としません

説明

OPEN PRINTING JOB コマンドはプリントジョブを開き、**CLOSE PRINTING JOB** コマンドが呼ばれるまで、続くすべてのプリント命令をスタックします。このコマンドはプリントジョブのコントロールを可能にし、特に印刷中に他のプリントジョブが予期せず挿入されないようにします。

OPEN PRINTING JOB コマンドはすべての4D印刷コマンド、クイックレポート、4D Writeや4D Viewプラグインの印刷コマンドで使用できます。他方このコマンドは4D Chartや4D Drawプラグイン、およびほとんどのサードパーティープラグインとは互換がありません。

このコマンドでプリントジョブが開かれると、プリントが実際に起動されるまで、プリンタは“busy”モードに置かれます。このコンテキストで互換のないプラグインがプリントジョブを起動すると、“printer busy”エラーが返されます。

CLOSE PRINTING JOB コマンドを呼び出してプリントジョブを終了し、印刷ドキュメントをプリンタに送信しなければなりません。このコマンドを呼び出さないと、印刷ドキュメントはスタックに置かれたままとなり、4Dアプリケーションを終了するまでプリンタは利用可能になりません。

プリントジョブはプロセスに対しローカルですが、印刷はグローバルレベルで実行されます。4D内で一度にひとつだけプリントジョブを開くことができます。

OPEN PRINTING JOBはカレントの印刷設定を使用します (デフォルト設定または**PAGE SETUP**や**SET PRINT OPTION**コマンドで設定された設定)。印刷設定を変更するコマンドは**OPEN PRINTING JOB**が呼ばれる前に実行されなければなりません。そうでなければエラーが生成されます。

LISTBOX SET ARRAY

LISTBOX SET ARRAY ({* ;} object ; arrType ; arrPtr)

引数	型		説明
*	演算子	➡	指定時:objectはオブジェクト名(文字列)省略時:objectは変数
object	フォームオブジェクト	➡	オブジェクト名(* 指定時)、または変数(* 省略時)
arrType	倍長整数	➡	配列のタイプ
arrPtr	ポインター	➡	プロパティに関連付ける配列を指定

説明

注: このコマンドは配列型のリストボックスに対してのみ有効です。

LISTBOX SET ARRAYコマンドは、 *object* and * によって指定されたリストボックスもしくはリストボックスコラムに、*arrType*配列を関連付けます。

注: デザインモードのプロパティリストを使用することによって、配列型のリストボックスにスタイル、文字色、背景色、行管理の配列を関連付けることができます。

任意の * 演算子を渡した場合、 *object* 引数でオブジェクト名を文字列で指定します。省略時には *object* 引数で変数を指定します。対象がリストボックスなのか列なのかを、 *object* 引数で指定します。

arrType 引数にはリストボックスまたは列に関連付けたい配列の種類を、 "**List Box**" のテーマ内にある以下の定数を渡すことによって指定します。

定数	型	値
lk background color array	倍長整数	1
lk control array	倍長整数	3
lk font color array	倍長整数	0
lk row height array	倍長整数	4
lk style array	倍長整数	2

arrPtr 引数には、制御したいプロパティを制御するための配列に対するポインターを渡します。

例題 1

4列目のフォントカラー配列を10列目にも使いたいという場合を考えます。

```
 // 4列目で使用している配列に対するポインターを取得します。
$Pointer:=LISTBOX Get array(*;"Col4";lk font color array)
 // 存在するかどうかをチェックします
If(Not(Nil($Pointer)))
 // 10列目へ適用します。
   LISTBOX SET ARRAY(*;"Col10";lk font color array;$Pointer)
End if
```

例題 2

リストボックス用に行高さ配列を設定します:

```
LISTBOX SET ARRAY(*;"LB";lk row height array;->RowHeightArray)
```

重要な注記: リストボックスの **行高さ配列** プロパティを利用するには有効な 4D View Pro ライセンスが必要です。有効なライセンスが存在しない場合には、ランタイムでリストボックスの内容は表示されず、代わりにエラーメッセージが表示されます。

LISTBOX SET HEADERS HEIGHT

LISTBOX SET HEADERS HEIGHT ({* ;} object ; height {; unit})

引数	型		説明
*	演算子	➡	指定時objectはオブジェクト名 (文字列) 省略時objectは変数
object	フォームオブジェクト	➡	オブジェクト名 (* 指定時) または変数 (* 省略時)
height	倍長整数	➡	ヘッダーの高さ
unit	倍長整数	➡	高さを指定する単位: 0または省略時 = ピクセル、1 = 行

説明

LISTBOX SET HEADERS HEIGHTコマンドは*object*と* 引数で指定したリストボックスのヘッダー行の高さを変更します。

オプションの * 引数を渡した場合、*object* 引数はオブジェクト名 (文字列) です。この引数を渡さない場合 *object* は変数です。この場合文字列ではなく変数参照を渡します。

リストボックスあるいはリストボックスヘッダーのいずれかを指定できます。

設定する高さを *height* 引数に渡します。*unit* 引数を省略するとデフォルトで高さはピクセル単位であらわされます。単位を変更するには *unit* 引数に**List Box**テーマの以下の定数を渡します:

定数	型	値	コメント
lk lines	倍長整数	1	高さを行数で指定。4Dはフォント設定に応じて高さを計算します。
lk pixels	倍長整数	0	高さをピクセルで指定 (デフォルト)。

ヘッダー高さの最小値はシステムに設定された値を使用します。

これはWindowsで24ピクセル、Mac OSで17ピクセルです。

*height*引数にこれより小さな値を渡すと、最小値が適用されます。

注: 行の高さの計算についてはデザインリファレンスマニュアルを参照してください。

Replace string

Replace string (source ; oldString ; newString {; howMany}{; *}) -> 戻り値

引数	型		説明
source	文字	⇒	元の文字列
oldString	文字	⇒	置き換対象の文字列
newString	文字	⇒	置き換え後の文字列 (空文字の場合オカレンスは削除)
howMany	倍長整数	⇒	置き換え 省略時、すべてのオカレンスを置き換え
*	演算子	⇒	渡されると、文字コードに基づいて評価
戻り値	文字	⇐	結果の文字列

説明

Replace stringは、*source*に存在するすべての*oldString*を*newString*で*howMany*回数だけ置き換えます。

*newString*が空の文字列 ("") の場合は、**Replace string**は*source*の中の*oldString*をすべて削除します。

*howMany*を指定した場合、**Replace string**関数は*source*の最初の文字から探して、その回数分だけ*oldString*を置き換えます。指定しない場合、発見した*oldString*をすべて置き換えます。

*oldString*が空の文字列の場合は、**Replace string**はなにも変更せず、元の文字列を返します。

デフォルトでこのコマンドはグローバルな比較を行い、言語上の特性と、1つ以上の文字で記述される文字 (例 æ = ae) を考慮に入れます。他方、発音区分符号 (a=A, a=à等) は無視され、文字コードが9未満の制御コードは考慮されません (Unicodeの仕様)。
この動作を変更するには、最後の引数にアスタリスク * を渡します。この場合、比較は文字コードベースで行われます。* 引数は以下のようなケースで必要となります:

- **Char**(1)など特別な文字を考慮に入れたい場合、
- 文字の評価で大文字小文字の区別やアクセント文字を考慮したい場合 (a#A, a#a 等)。

このモードでは、単語が書かれた方法のバリエーションが評価されないことに留意してください。

注: 4D v15 R3 以降、使用するシンタックスに関わらず、文字列を異なる長さの文字列で置き換える際にこのコマンドが使用するアルゴリズムに対し大幅な最適化が行われました。その結果、このコンテキストにおける処理が飛躍的に早くなりました。

例題 1

Replace stringの使用例を次に示します。結果を変数*vtResult*に代入します。コメントは、変数*vtResult*に代入される内容についての説明です。

```
vtResult:=Replace string("Willow";" ll";"d") `Resultは"Widow"
vtResult:=Replace string("Shout";"o";"") `Resultは"Shut"
vtResult:=Replace string(vtOtherVar;Char(Tab);",";*) `vtOtherVar の中の全てのタブをコンマ(,) に置き換える
```

例題 2

以下の例は、*vtResult*のテキストからキャリッジリターンとタブを取り除きます。

```
vtResult:=Replace string(Replace string(vtResult;Char(Carriage return);"";*);Char(Tab);"";*)
```

例題 3

この例では発音区分符号を区別するために、* 引数の使用する例を示します。

```
vtResult:=Replace string("Crème brûlée";"Brulee";"caramel") `Result gets "Crème caramel"
vtResult:=Replace string("Crème brûlée";"Brulee";"caramel";*) `Result gets "Crème brûlée"
```

String

String (expression {; format {; addTime}}) -> 戻り値

引数	型		説明
expression	式	➡	文字列式を返したい式 (実数、整数、倍長整数、日付、時間、文字列、 テキスト、ブールを指定可能)
format	文字, 倍長整数	➡	表示フォーマット
addTime	時間	➡	expressionが日付の時、文字列に追加する時間
戻り値	文字	⮌	式の文字列式

説明

Stringコマンドは、*expression*に渡した数値、日付、時間、文字列、またはブールを文字列に変換します。

オプションの*format*を指定しない場合、適当なデフォルトの形式で文字列が返されます。*format*を指定すると、結果の文字列は指定した形式になります。

オプションの*addTime*引数は、日付に時間を複合フォーマットで追加します。この引数は*epression*引数が日付型の時にのみ使用できます (後述)。

数値式

*expression*が数値（実数、整数、倍長整数）である場合、オプションで文字列フォーマットを渡すことができます。次に例を示します。

例題	結果	コメント
String(2^15)	"32768"	デフォルトフォーマット
String(2^15;"###,##0 Inhabitants")	"32,768 Inhabitants"	
String(1/3;"##0.00000")	"0.33333"	
String(1/3)	"0.3333333333333 "	デフォルトフォーマット(*)
String(Arctan(1)*4)	"3.14159265359 "	デフォルトフォーマット(*)
String(Arctan(1)*4;"##0.00")	"3.14"	
String(-1;"&x")	"0xFFFFFFFF"	
String(-1;"&$")	"$FFFFFFFF"	
String(0 ?+ 7;"&x")	"0x0080"	
String(0 ?+ 7;"&$")	"$80"	
String(0 ?+ 14;"&x")	"0x4000"	
String(0 ?+ 14;"&$")	"$4000"	
String(50,3;"&xml")	"50.3"	必ず "." を小数点として使用
String(Num(1=1);"True;;False")	"True"	
String(Num(1=2);"True;;False")	"False"	
String(Log(-1))	""	未定義の数値
String(1/0)	"INF"	正の無限数
String(-1/0)	"-INF"	負の無限数

(*) 4D v14 R3 以降、実数をテキストへと変換するアルゴリズムは、有効数字13桁に基づいて計算されています(それ以前のバージョンの4Dは15桁)

フォーマットは、フォームの数値フォーマットと同じ方法で指定します。数値フォーマットの詳細については4D Design Referenceマニュアルの**表示フォーマット**を参照してください。カスタムフォーマットの書式名を*format*に渡すことができます。カスタムフォーマットの名前は“|”で始めなければなりません。

注: コンパイルモードにおいて、**String** 関数は "整数64bit" 型フィールドと互換性がありません。

日付式

*expression*が日付の場合、デフォルトフォーマット（YY.MM.DD）を使用して文字列が返されます。*format* 引数には、以下で説明する定数のいずれか１つを渡すことができます (**Date Display Formats**テーマ) 。

この場合*addTime*引数に時間を渡すことができます。この引数を使用すれば日付と時間を合成し、標準形式のタイムスタンプを生成することができます (ISO Date、ISO Date GMT、Date RFC 1123定数)。このフォーマットはXMLやWebの処理の際特に有効です。*addTime*引数は*expression*が日付型の場合のみ使用できます。

定数	型	値	コメント
Blank if null date	倍長整数	100	0の代わりに""
Date RFC 1123	倍長整数	10	Fri, 10 Sep 2010 13:07:20 GMT
Internal date abbreviated	倍長整数	6	Dec 29, 2006
Internal date long	倍長整数	5	December 29, 2006
Internal date short	倍長整数	7	2006/12/29
Internal date short special	倍長整数	4	06/12/29 (しかし 1986/12/29 または 2096/12/29)
ISO Date	倍長整数	8	2006-12-29T00:00:00
ISO Date GMT	倍長整数	9	2010-09-13T16:11:53Z
System date abbreviated	倍長整数	2	
System date long	倍長整数	3	
System date short	倍長整数	1	

注: 表示フォーマットはシステム設定に応じて変化する可能性があります。

以下は今日が2006/12/29の場合の例です。

```
$vsResult:=String(Current date) // $vsResultは"06/12/29"
$vsResult:=String(Current date;Internal date long) // $vsResultは"December 29, 2006"
$vsResult:=String(Current date;ISO Date GMT) // $vsResultは日本の場合"2006-12-28T15:00:00Z"
```

日付/時間複合フォーマットに関するメモ:

- ISO Date GMTフォーマットはISO8601標準に対応します。日付と時間を含みタイムゾーン (GMT) を考慮します。

  ```
  $mydate:=String(Current date;ISO Date GMT;Current time) // 2010-09-13T16:11:53Zを返します。
  ```

  上記の例の最後の"Z"はGMTフォーマットの終了を表します。
  *addTime*引数を渡さない場合、ローカルタイムの午前０時として扱いGMTに変換します (例題参照)。そのため日付がローカルタイムに対して前後することがあります。

  ```
  $mydate:=String(Current date;ISO Date GMT) // 2010-09-12T15:00:00Zを返す
  ```

- ISO DateフォーマットはISO Date GMTフォーマットと同様に日付と時間を含みますが、タイムゾーンを考慮しません。当初よりこのフォーマットはISO8601標準に準拠しておらず、非常に特殊な目的のために予約されたものです。

  ```
  $mydate:=String(!13/09/2010!;ISO Date) // 2010-09-13T00:00:00 を返す(タイムゾーンに依存しない)
      $mydate:=String(Current date;ISO Date;Current time) // 2010-09-13T18:11:53 を返す
  ```

- Date RFC 1123フォーマットは日付と時間の組み合わせをRFC 822と1123で定義された標準に基づきフォーマットします。このフォーマットはたとえばHTTPヘッダーでcookieの有効期限を設定する際に必要となります。

  ```
  $mydate:=String(Current date;Date RFC 1123;Current time) // Fri, 10 Sep 2010 13:07:20 GMTを返す
  ```

  表現される日時は、タイムゾーンが考慮されるためローカルのタイムゾーンにより日付が前後にずれることになります。日付のみを渡すと、コマンドはローカルタイムの00:00をGMT時間で表現して返します:

  ```
  $mydate:=String(Current date;Date RFC 1123) // Thu, 09 Sep 2010 15:00:00 GMTを返す
  ```

時間式

*expression*が時間の場合、デフォルトフォーマット(HH:MM:SS)を使用して文字列が返されます。*format* 引数には、以下の表に示す定数のいずれか１つを渡すことができます (**Time Display Formats**テーマ) 。

定数	型	値	コメント
Blank if null time	倍長整数	100	0の代わりに""
HH MM	倍長整数	2	01:02
HH MM AM PM	倍長整数	5	1:02 AM
HH MM SS	倍長整数	1	01:02:03
Hour min	倍長整数	4	1時2分
Hour min sec	倍長整数	3	1時2分3秒
ISO time	倍長整数	8	0000-00-00T01:02:03
Min sec	倍長整数	7	62分3秒
MM SS	倍長整数	6	62:03
System time long	倍長整数	11	1:02:03 AM HNEC (Macのみ)
System time long abbreviated	倍長整数	10	1:02:03 AM (Macのみ)
System time short	倍長整数	9	01:02:03

Notes:

- ISO Dateフォーマットは ISO8601標準に対応し、日付と時間を含みます。このフォーマットは日付と時間の複合をサポートしないため、日付部分は0で埋められます。このフォーマットはローカル時間を表します。
- Blank if null**定数**はフォーマットに他のフォーマットに加算して使用しなければなりません。追加することにより、Null値の場合、4Dは0の代わりに空の文字列を返します。

以下の例は、現在時刻が5:30 PM45秒であるものとします。

```
$vsResult:=String(Current time) // $vsResultは"17:30:45"
$vsResult:=String(Current time;Hour Min Sec) // $vsResultは"17時30分45秒"
```

文字列式
*expression*が文字列またはテキスト型の場合、コマンドは引数に渡した値と同じ値を返します。これは特にポインタを使用している汎用プログラミングで有効です。
この場合、*format*引数は渡されても無視されます。

ブール式
*expression*がブール 型の場合、コマンドはアプリケーションのランゲージに文字列 "True" または"False"を返します(例えば、4Dのフランス語バージョンでは、“Vrai”または“Faux”)。
この場合、*format*引数は渡されても無視されます。

4D 変換タグ

4Dでは、参照を4D変数や式に挿入したり、異なる種類の処理をソーステキストに対して実行したりするための変換タグのセットを用意しています。こ れは別名”テンプレート”とも呼ばれます。これらのタグはソーステキストが実行されてアウトプットテキストが生成されたときに解釈されます。

これらの原理は特に4D Web サーバーにおいて**セミダイナミックページ** を生成するのに使用されます。

これらのタグは原則としてHTMLコメント(*<!--#Tag Contents-->*)として挿入される必要があります。しかしながら、 *<!--Beginning of list-->* などの他のコメントも使用可能です。

注: 値を返すタグに対してはXMLに準拠させるために特定の場合において$-ベースの代替シンタックスが使用されます。より詳細な情報については、以下の段落**4DTEXT、4DHTML、4DEVALにおける代替シンタックス**を参照して下さい。

以下の4D変換タグを使用できます:

- 4DTEXT: 4D変数や4D式を挿入する為に使用します。
- 4DHTML: HTMLコードを挿入するために使用します。
- 4DEVAL: 4D式を評価するために使用します。
- 4DSCRIPT: 指定された4Dメソッドを実行します。またメソッドの戻り値を挿入します。
- 4DINCLUDE: ページに他のページを挿入するために使用します。
- 4DBASE: 4DINCLUDEで使用されるデフォルトフォルダーを変更します。
- 4DCODE: 4Dコードのブロックを挿入するために使用します。
- 4DIF, 4DELSE, 4DELSEIF そして 4DENDIF: HTMLコードに条件分岐を挿入するために使用します。
- 4DLOOP と 4DENDLOOP: HTMLコードでループを行うために使用します。

複数のタイプのタグを混用することも可能です。例えば、以下のHTML構造は、何の問題もなく実行可能です:

```
<HTML> ... <BODY> <!--#4DSCRIPT/PRE_PROCESS-->          (メソッド呼び出し) <!--#4DIF (myvar=1)--
>                (If 条件)    <!--#4DINCLUDE banner1.html-->   (サブページ挿入) <!--#4DENDIF--
>                    (End if) <!--#4DIF (mtvar=2)-->    <!--#4DINCLUDE banner2.html--> <!--
#4DENDIF--> <!--#4DLOOP [TABLE]-->               (カレントセレクションでのループ) <!--#4DIF
([TABLE]ValNum>10)-->          (If [TABLE]ValNum>10)    <!--#4DINCLUDE subpage.html-->   (サブ
ページの挿入) <!--#4DELSE-->                      (Else)    <B>Value: <!--#4DTEXT [TABLE]ValNum--
></B><BR>                                  (フィールド表示) <!--#4DENDIF--> <!--#4DENDLOOP--
>                  (End for) </BODY> </HTML>
```

テンプレートの実行について

"テンプレート"ページの中身をパースするのは、二通りのやり方があります:

- **PROCESS 4D TAGS** コマンドを使用する方法: このコマンドは"テンプレート"に加えて引数(任意)をを入力として受付け、処理の結果のテキストを返します。
- 4Dの統合されたHTTPサーバーを使用する: **WEB SEND FILE**(.htm, .html, .shtm, .shtml)、**WEB SEND BLOB** (text/html型BLOB)、または **WEB SEND TEXT** コマンド、またはURLの呼び出しを使用して送信された**セミダイナミックページ** を使用して解析を行います。URLの呼び出しを使用する場合、最適化のために、 “.htm” と “.html” で終わるページに関しては**解析されません**。この場合においてHTMLページを強制的に解析させるためには、終わりに“.shtm” または“.shtml”にする必要があります(例 http://www.server.com/dir/page.shtm)。この点についての詳細に関しては、**Webサーバ** 内の **セミダイナミックページ** の章を参照して下さい。

4DTEXT

シンタックス: *<!--#4DTEXT VarName-->* または *<!--#4DTEXT 4DExpression-->*
代替シンタックス: *$4DTEXT(VarName)* または *$4DTEXT(4DExpression)* (**4DTEXT、4DHTML、4DEVALにおける代替シンタックス** を参照して下さい)

タグ *<!--#4DTEXT VarName-->* を使用して4D変数や値を返す式への参照を挿入します。例えば(HTMLページ内にて)以下のように記述すると:

<P>Welcome to *<!--#4DTEXT vtSiteName-->*!</P>

4D変数 *vtSiteName* の値がHTMLページに送信時に挿入されます。値はテキストとして挿入されます。">"のようなHTMLの

特殊文字は、自動的にエスケープされます。

4DTEXT タグを使用して、変数だけでなく4D式も挿入できます。フィールドの値を直接挿入できますし (例 *<!--#4DTEXT [tableName]fieldName-->*) 、または配列要素の値も挿入できますし (例 *<!--#4DTEXT tabarr{1}-->*) 、値を返すメソッドも使用できます (*<!--#4DTEXT mymethod-->*)。

式の変換は変数のそれと同じルールが適用されます。さらに式は4Dのシンタックスルールに適合していなければなりません。

注: Webリクエストからの *4DTEXT* での4Dメソッドの実行は、メソッドプロパティの“4DHTMLタグとURL (4DACTION) で利用可能”属性の設定に基づきます。詳細は **接続セキュリティ** を参照してください。

式に4D関数への呼び出しを直接含めることができますが、これはローカライズの観点から推奨されません。例えば *<!--#4DTEXT Current date-->* は英語バージョンの4Dでは正しく解釈されますが、フランス語バージョンの4Dでは解釈されません。同じことが実数値にも言えます (小数点は言語により異なります)。どちらの場合にもプログラムを使用して変数値を割り当てることを強くお勧めします。

使用する4Dランゲージやバージョンに関係なく式が正常に評価される事を保証するためには、異なるバージョン間で名前が変化する可能性のある要素(コマンド、テーブル、フィールド、定数)に対してトークンシンタックスを使用することが推奨されます。例えば、**Current time**コマンドを挿入するためには、'**Current time:C178**' と入力します。この点の詳細な情報に付いては、**フォーミュラ内でのトークンの使用**を参照して下さい。

解釈エラーの場合、“*<!--#4DTEXT myvar-->* : ## error # error code” のように表示されます。

注:

- プロセス変数を使用できます。
- セキュリティ上の理由から、悪意あるコードの侵入・挿入を防ぐために、アプリケーション外から導入されたデータを処理するときにはこのタグを使用する事が推奨されます(以下の**使用上の注意**の章を参照して下さい)。
- ピクチャーーフィールドの内容を表示できます。しかしピクチャー配列の項目内容を表示することはできません。
- 4D式を使用して、オブジェクトフィールドの中身を表示させることができます。例えば、以下の様に記述します。 *<!--#4DTEXT **OB Get**:C1224([Rect]Desc;¥"color¥")-->*
- 通常はテキスト変数を使用します。しかしBLOB変数を使用することもできます。この場合長さ情報なしのテキストBLOBを使用します。

4DHTML

シンタックス: *<!--#4DHTML VarName-->* または *<!--#4DHTML 4DExpression-->*
代替シンタックス: *$4DHTML(VarName)* または *$4DHTML(4DExpression)* (**4DTEXT、4DHTML、4DEVALにおける代替シンタックス**を参照して下さい)

4DTEXTタグ同様、この4DHTMLタグを使用すると、変数や値を返す4D式をHTML式として挿入できます。4DTEXTタグとは異なり、このタグはHTML特殊文字をエスケープしません。

例えば4Dのテキスト変数 *myvar* を4Dタグを使用して処理した結果は以下の様になります:

myvar 値	タグ	Webページに挿入されるテキスト
myvar:="<B>"	*<!--#4DTEXT myvar-->*	<B>
myvar:="<B>"	*<!--#4DHTML myvar-->*	<B>

使用する4Dランゲージやバージョンに関係なく式が正常に評価される事を保証するためには、異なるバージョン間で名前が変化する可能性のある要素(コマンド、テーブル、フィールド、定数)に対してトークンシンタックスを使用することが推奨されます。例えば、**Current time**コマンドを挿入するためには、'**Current time:C178**' と入力します。この点の詳細な情報に付いては、**フォーミュラ内でのトークンの使用**を参照して下さい。

解釈エラーの場合、“*<!--#4DHTML myvar-->* : ## エラー # エラーコード”のように表示されます。

注:

- Webリクエスト経由の *4DHTML*での4Dメソッドの実行は、メソッドプロパティの“4D HTMLタグおよびURL (4DACTION) で利用可能”属性の設定に基づきます。詳細は **接続セキュリティ** を参照してください。
- セキュリティ上の理由から、悪意あるコードの侵入・挿入を防ぐために、アプリケーション外から導入されたデータを処理するときには4DTEXTタグを使用する事が推奨されます(以下の**使用上の注意**の章を参照して下さい)。

4DEVAL

シンタックス: *<!--#4DEVAL VarName-->* または *<!--#4DEVAL 4DExpression-->*
代替シンタックス: *$4DEVAL(VarName)* または *$4DEVAL(4DExpression)* (**4DTEXT、4DHTML、4DEVALにおける代替シンタックス**を参照して下さい)

4DEVAL タグを使用すると、変数や4D式を挿入できます。既存の4DHTMLタグのように、4DEVALタグはテキストを返す際にHTML特殊文字をエスケープし ません。しかしながら、4DHTMLや4DTEXTと異なり、4DEVALは有効な4D宣言であればどれでも実行することができます(値を返さない割り当 てや式も含まれます)。

例えば、以下の様なコードを実行することができます:

```
$input:="<!--#4DEVAL a:=42-->" //割り当て
$input:=$input+"<!--#4DEVAL a+1-->" //計算
PROCESS 4D TAGS($input;$output)
 //$output = "43"
```

使用する4Dランゲージやバージョンに関係なく式が正常に評価される事を保証するためには、異なるバージョン間で名前が変化する可能性のある要素(コマンド、テーブル、フィールド、定数)に対してトークンシンタックスを使用することが推奨されます。例えば、**Current time**コマンドを挿入するためには、'**Current time:C178**' と入力します。この点の詳細な情報に付いては、**フォーミュラ内でのトークンの使用**を参照して下さい。

解釈エラーの場合、"*<!--#4DEVAL expr-->* : ## エラー # エラーコード"のように表示されます。

注:

- Webリクエスト経由の *4DHTML*での4Dメソッドの実行は、メソッドプロパティの"4D HTMLタグおよびURL (4DACTION) で利用可能"属性の設定に基づきます。詳細は **接続セキュリティ** を参照してください。
- セキュリティ上の理由から、悪意あるコードの侵入・挿入を防ぐために、アプリケーション外から導入されたデータを処理するときには4DTEXTタグを使用する事が推奨されます(以下の**使用上の注意**の章を参照して下さい)。

4DSCRIPT/

シンタックス: *<!--#4DSCRIPT/MethodName/MyParam-->*

4DSCRIPT タグを使用して、スタティックなHTMLページを処理する際に4Dメソッドを実行することを可能にします。*<!--#4DSCRIPT/MyMethod/MyParam-->* タグがHTMLコメントとしてページに現れると、***MyMethod***メソッドが*$1*引数に *Param* を受け取って実行されます。

注: タグがWebプロセスのコンテキストにおいて呼び出された場合、Webページがロードされると、4Dは **On Web Authenticationデータベースメソッド** を (存在すれば) 呼び出します。このメソッドが**True**を返すと、4Dはメソッドを実行します。

メソッドは*$0*にテキストを返す必要があります。文字列がコード1から始まっていると、それは　HTMLソースとして扱われます (*4DHTML*のときと同じ原則)。
注: *4DSCRIPT*での4Dメソッドの実行は、メソッドプロパティの"4D タグおよびURL (4DACTION) で利用可能"属性の設定に基づきます。詳細は **接続セキュリティ** を参照してください。

例えば、以下のコメントをセミダイナミックWebページに挿入したとしましょう "Today is *<!--#4DSCRIPT/MYMETH/MYPARAM-->*" 。ページをロードする際、4Dは **On Web Authenticationデータベースメソッド** を (存在すれば) 呼び出し、そして ***MYMETH*** メソッドの*$1*引数に文字列 "/MYPARAM" を渡して呼び出します。

メソッドは*$0*にテキストを返します (例えば "14/12/31")。コメント "Today is *<!--#4DSCRIPT/MYMETH/MYPARAM-->*" は結果 "Today is 14/12/31" となります。

MYMETHメソッドは以下のとおりです:

```
C_TEXT($0)\\This parameter must always be declared
C_TEXT($1)\\This parameter must always be declared
$0:=String(Current date)
```

警告: *$0* と *$1* 引数を宣言しなければなりません。

注: *4DSCRIPT*から呼び出されるメソッドはインタフェース要素 (**DIALOG**, **ALERT**...) を呼び出してはいけません。

4Dはメソッドを見つけた順に実行するので、ドキュメント中で離れて参照される変数の値を設定するメソッドを呼び出すことも可能です。モードは関係ありません。テンプレートには必要なだけ *<!--#4DSCRIPT...-->* コメントを記述できます。

4DINCLUDE

シンタックス: *<!--#4DINCLUDE Path-->*
このタグは主に、このコメントを使用して、(*path*引数で指定された) 他のHTMLページを、これから送信するHTMLに含めるためにデザインされました。デフォルトでHTMLのボディー部、つまり<body> と</body>タグの間の内容だけが統合されます (bodyタグは含められません)。これによりヘッダーに含まれるメタタグ関連の衝突が回避されます。しかし*path*で指定されたHTML中に<body></body>タグが含まれない場合、ページ全体が統合されます。この場合メタタグの整合性を管理するのは開発者の役割です。

<!--#4DINCLUDE --> コメントは、テスト (*<!--#4DIF-->*) やループ (*<!--#4DLOOP-->*) と使用するととても便利です。条件に基づきあるいはランダムにタグを含める便利な方法です。
このタグを使用してページをインクルードするとき、拡張子にかかわらず、4Dは呼び出されたページを解析してから、内容を*4DINCLUDE*呼び出し元のページに挿入します。

<!--#4DINCLUDE --> コメントで挿入されたページはロードされ、URLで呼ばれて**WEB SEND FILE**コマンドで送信されたファイルと同じように、Webサーバーキャッシュに格納されます。

*path*には、含めるドキュメントへのパスを記述します。

警告: *4DINCLUDE*呼び出しの場合、パスは解析される親ドキュメントからの相対パスです。フォルダ区切り文字にはスラッシュ (/) を使用し、レベルをさかのぼるには2つのドットを使用します (HTMLシンタックス)。
注:

- **PROCESS 4D TAGS**コマンドで*4DINCLUDE*タグを使用する場合、デフォルトフォルダはデータベーストラクチャを含むフォルダーです。
- *4DINCLUDE* タグで使用されるデフォルトフォルダーを*<!--#4DBASE -->*タグを使用して変更できます (後述)。

ページ内の*<!--#4DINCLUDE path-->*数に制限はありません。しかし*<!--#4DINCLUDE path-->*の呼び出しは１レベルのみ有効です。つまり、例えばmydoc1.html内の*<!--#4DINCLUDE mydoc2.html-->*で呼ばれるmydoc2.htmlページのボディに、*<!--#4DINCLUDE mydoc3.html-->* を含めることはできません。

さらに、4Dは組み込み呼び出しが再帰的でないか確認します。

エラーの場合、“*<!--#4DINCLUDE path-->*：ドキュメントを開けません”のように表示されます。

例題

```
<!--#4DINCLUDE subpage.html--> <!--#4DINCLUDE folder/subpage.html--> <!--#4DINCLUDE
../folder/subpage.html-->
```

4DBASE

シンタックス: *<!--#4DBASE folderPath-->*

<!--#4DBASE --> タグは*<!--#4DINCLUDE-->*タグで使用されるワーキングディレクトリを指定します。
Webページ内でこのタグが使用されると、*<!--#4DBASE -->* タグはこのページ内でそのあとに続くすべての*<!--#4DINCLUDE-->* 呼び出しのディレクトリを変更します。組み込まれたファイル内で*<!--#4DBASE -->*フォルダーが変更されると、親のファイルから元となる値を取得します。

folderPath 引数には現在のページに対する相対パスを指定し、パスは"/"で終わっていなければなりません。指定するフォルダーはWebフォルダー内になければなりません。
WEBFOLDER キーワードを渡すと、デフォルトパスに戻されます (そのページに対して相対)。

4D v12では以下のように各呼び出しごとに相対パスをしていなければなりませんでした:

```
<!--#4DINCLUDE subpage.html--> <!--#4DINCLUDE folder/subpage1.html--> <!--#4DINCLUDE folder/subpage2.html--> <!-
-#4DINCLUDE folder/subpage3.html--> <!--#4DINCLUDE ../folder/subpage.html-->
```

<!--#4DBASE --> タグを使用すれば以下のように記述できます:

```
<!--#4DINCLUDE subpage.html--> <!--#4DBASE folder/--> <!--#4DINCLUDE subpage1.html--> <!--#4DINCLUDE
subpage2.html--> <!--#4DINCLUDE subpage3.html--> <!--#4DBASE ../folder/--> <!--#4DINCLUDE subpage.html--> <!--
#4DBASE WEBFOLDER-->
```

Example

<!--#4DBASE --> タグを使用してホームページのディレクトリを設定する:

```
/* Index.html */ <!--#4DIF LangFR=True-->     <!--#4DBASE FR/--> <!--#4DELSE-->     <!--
#4DBASE US/--> <!--#4DENDIF--> <!--#4DINCLUDE head.html--> <!--#4DINCLUDE body.html--> <!--
#4DINCLUDE footer.html-->
```

head.html ファイル内でカレントフォルダーが*<!--#4DBASE -->*を使用して変更されているが、index.html内では変更されない:

```
/* Head.htm */ /* ここでのワーキングディレクトリはインクルードされるファイルに対して相対的 (FR/ または US/) */ <!--#4DBASE
Styles/--> <!--#4DINCLUDE main.css--> <!--#4DINCLUDE product.css--> <!--#4DBASE Scripts/-->
<!--#4DINCLUDE main.js--> <!--#4DINCLUDE product.js-->
```

4DCODE

4DCODEタグを使用すると、テンプレートに複数行の4Dコードのブロックを挿入する事ができます。

"<!--#4DCODE" シークエンスとそれに続くスペース、CRまたはLF文字が検知されると、4Dは次の"-->"シークエンスまで

のコードを解釈します。コードブロック自体はキャリッジリターンもラインフィードも、あるいはその両方も含む事ができ、4Dによってシーケンシャルに解釈されます。

例えば、4DCODEタグを使用して、以下のようにテンプレート内に書く事ができます:

```
<!--#4DCODE
//PARAMETERS 初期化

$graphType:=1
If(OB Is defined:C1231($graphParameters;"graphType")) //US言語のみ
    $graphType:=OB GET:C1224($graphParameters;"graphType")
    If($graphType=7)
        $nbSeries:=1
        If($nbValues>8)
            DELETE FROM ARRAY:C228 ($yValuesArrPtr{1}->;9;100000)
            $nbValues:=8
        End if
    End if
End if
-->
```

注: 4DCODEタグ内では、4Dコードは必ずEnglish-US言語で書かれている必要があります。そのため、4DCODEは4Dランゲージの"リージョナルシステム設定を使用"のユーザー設定を無視します(**コマンドと定数のためのランゲージ**を参照して下さい)。

4DCODEタグの機能は以下の通りです:

- **TRACE**コマンドはサポートされており、4Dデバッガを起動するので、テンプレートコードをデバッグすることができます。
- どのようなエラーであろうと、標準のエラーダイアログが表示され、ユーザーがコードの実行を中止したりデバッグモードに入ったりすることができます。
- <!--#4DCODE と --> の間のテキストは改行され、どのような改行コードでも受け取ります(cr、lf、またはcrlf)
- テキストは **PROCESS 4D TAGS**を呼び出したデータベースのコンテキストにてトークナイズドされます。これは例えばプロジェクトメソッドの認識等において重要です。
  注: "4DタグとURLの4DACTION経由で利用可"メソッドプロパティは考慮されません(以下の**セキュリティに関する注意**も参照して下さい)。
- テキストが常にEnglish-US設定であったとしても、あるバージョンの4Dから他のバージョン間においてコマンドや定数名が改名されていることによる問題を避けるために、コマンド名や定数名はトークンシンタックスを使用する事が推奨されいます。
  注: *:Cxxx*シンタックスに関する詳細な情報については、**フォーミュラ内でのトークンの使用**の章を参照して下さい。

セキュリティに関する注意: 4DCODEタグがどのような4Dランゲージコマンドあるいはプロジェクトメソッドでも呼び出せると言う事実は、特にデータベースがHTTP経由で使用可能な場合等に、セキュリティ上の問題になり得ます。しかしながら、タグはサーバー側でのコードをテンプレートファイルから実行するため、タグそのものはセキュリティ上の問題にはなりません。このようなコンテキストにおいては、どのようなWebサーバーと同様、セキュリティは主にサーバーファイルへのリモートアクセスレベルにおいて管理されています。

4DIF, 4DELSE, 4DELSEIF and 4DENDIF

シンタックス: *<!--#4DIF expression--> {<!--#4DELSEIF expression2-->...<!--#4DELSEIF expressionN-->} {<!--#4DELSE-->} <!--#4DENDIF-->*

<!--#4DELSEIF--> (オプション), *<!--#4DELSE-->* (オプション) と *<!--#4DENDIF-->* コメントと共に使用することで、*<!--#4DIF expression-->* コメントはコードの一部に条件分岐を実行させることを可能にします。

expression 引数はブール値を返す有効な4D式です。式は括弧の中に記述され、4Dのシンタックスルールに準拠していなければなりません。

使用する4Dランゲージやバージョンに関係なく式が正常に評価される事を保証するためには、異なるバージョン間で名前が変化する可能性のある要素(コマンド、テーブル、フィールド、定数)に対してトークンシンタックスを使用することが推奨されます。例えば、**Current time**コマンドを挿入するためには、'**Current time:C178**' と入力します。この点の詳細な情報に付いては、**フォーミュラ内でのトークンの使用**を参照して下さい。

<!--#4DIF expression--> ... *<!--#4DENDIF-->* を複数レベルでネストできます。4Dのようにそれぞれの *<!--#4DIF expression-->* は *<!--#4DENDIF-->*とマッチしなければなりません。

解釈エラーの場合、*<!--#4DIF -->* と *<!--#4DENDIF-->*“の間のコンテンツの代わりに、*<!--#4DIF expression-->*: ブール式が必要です”が挿入されます。

同様に、*<!--#4DIF -->*が同じ数の*<!--#4DENDIF-->*で閉じられていない場合、“*<!--#4DIF expression-->*: 4DENDIFが

必要です” が*<!--#4DIF -->* と *<!--#4DENDIF-->*の間のコンテンツの代わりに挿入されます。

<!--#4DELSEIF--> タグを使用すると、条件テストの記述が容易になります。最初に**True**と判定されたブロック内にあるコードだけが実行されます。Trueブロックがない場合、文は実行されません (*<!--#4DELSE-->*がなければ)。
最後の*<!--#4DELSEIF-->*の後に*<!--#4DELSE-->*タグを記述できます。すべての条件が**False**の場合、*<!--#4DELSE-->*ブロックの文が実行されます。

以下の2つのコードは同等です

- 4DELSEのみを使用する場合:

```
<!--#4DIF Condition1-->   /* Condition1 is true*/ <!--#4DELSE-->     <!--#4DIF Condition2-->         /*
Condition2 is true*/     <!--#4DELSE-->         <!--#4DIF Condition3-->            /* Condition3 is true
*/         <!--#4DELSE-->             /*None of the conditions are true*/         <!--#4DENDIF-->     <!--
#4DENDIF--> <!--#4DENDIF-->
```

- 同じ内容を4DELSEIFタグを使用して記述した場合:

```
<!--#4DIF Condition1-->      /* Condition1 is true*/ <!--#4DELSEIF Condition2-->      /* Condition2 is
true*/ <!--#4DELSEIF Condition3-->     /* Condition3 is true */ <!--#4DELSE-->     /* None of the
conditions are true*/ <!--#4DENDIF-->
```

例題 1

スタティックなHTMLページに書かれたこの例題のコードは、vname#"" 式の結果に応じ、異なるラベルを表示します:

```
<BODY> ... <!--#4DIF vname#""--> Names starting with <!--#4DVAR vname-->. <!--#4DELSE--> No
name has been found. <!--#4DENDIF--> ... </BODY>
```

例題 2

この例題は接続したユーザーに基づき異なるページを返します:

```
<!--#4DIF LoggedIn=False-->     <!--#4DINCLUDE Login.htm --> <!--#4DELSEIF User="Admin"-->
<!--#4DINCLUDE AdminPanel.htm --> <!--#4DELSEIF User="Manager"-->     <!--#4DINCLUDE
SalesDashboard.htm --> <!--#4DELSE-->     <!--#4DINCLUDE ItemList.htm --> <!--#4DENDIF-->
```

4DLOOP と 4DENDLOOP

シンタックス: <!--#4DLOOP condition--> <!--#4DENDLOOP-->

このコメントを使用して、条件を満たす間、コードの一部を繰り返すことができます。繰り返し部のコードは<!--#4DLOOP--> と <!--#4DENDLOOP-->で挟まれます。

<!--#4DLOOP condition--> ... <!--#4DENDLOOP--> ブロックはネストできます。4Dと同様、それぞれの<!--#4DLOOP condition--> は同じ数の <!--#4DENDLOOP-->で閉じられていなければなりません。

条件には5種類あります:

- **<!--#4DLOOP [テーブル]-->**
  このシンタックスは、指定された *table* のカレントプロセスのカレントセレクションに基づき、レコードごとにループします。2つのコメントの間のコードはカレントセレクションレコード毎に繰り返されます。

注: 4DLOOPタグがテーブルを条件として使用されると、レコードが読み込みのみでロードされます。

以下のコードは:

```
<!--#4DLOOP [People]--> <!--#4DTEXT [People]Name--> <!--#4DTEXT [People]Surname--><BR> <!--
#4DENDLOOP-->
```

4Dランゲージで表すと以下のとおりです:

```
FIRST RECORD([People])
While(Not(End selection([People])))
 // ...
   NEXT RECORD([People])
End while
```

- **<!--#4DLOOP 配列-->**

このシンタックスは、配列項目ごとにループします。2つのコメントの間のコードが繰り返されるたびに、配列のカレント項目がインクリメントされます。

注: このシンタックスで二次元配列を使用することはできません。この場合、ネストしたループでメソッドを条件に使用します。

以下のコードは:

```
<!--#4DLOOP arr_names--> <!--#4DTEXT arr_names{arr_names}--><BR> <!--#4DENDLOOP-->
```

4Dランゲージで表すと以下のとおりです:

```
For($Elem;1;Size of array(arr_names))
   arr_names:=$Elem
 //...
End for
```

- **<!--#4DLOOP method-->**
  このシンタックスでは、メソッドがTrueを返す間ループが行われます。メソッドは倍長整数タイプの引数を受け取ります。まずメソッドは引数0を渡されます。これは (必要に応じて) 初期化ステージとして使用できます。その後、Trueが返されるまで1, 2, 3と渡される引数値がインクリメントされます。

セキュリティのため、Webプロセス内では、**On Web Authenticationデータベースメソッド**が初期化ステージ (引数に0が渡されて実行される) の前に一度呼び出されます。認証されると、初期化に進みます。

警告: コンパイルのため、**C_BOOLEAN**($0) と**C_LONGINT**($1) が宣言されていなければなりません。

例題以下のコードは:

```
<!--#4DLOOP my_method--> <!--#4DTEXT var--> <BR> <!--#4DENDLOOP-->
```

4Dランゲージで表すと以下のとおりです:

```
If(AuthenticationWebOK)
   If(my_method(0))
      $counter:=1
      While(my_method($counter))
 //...
         $counter:=$counter+1
      End while
   End if
End if
```

my_method は以下のようになります:

```
C_LONGINT($1)
C_BOOLEAN($0)
If($1=0)
 `Initialisation
   $0:=True
Else
   If($1<50)
 //...
      var:=...
      $0:=True
   Else
      $0:=False `Stops the loop
   End if
End if
```

- **<!--#4DLOOP 4D式-->**
  このシンタックスでは、4DLOOPタグは4D式が**True**を返す間ループが行われます。式は有効なブール式であれば問題はなく、無限ループを防ぐために、毎ループごとに評価されるための変数部分を含んでいる必要があります。
  例えば以下のコードは:

```
<!--#4DEVAL $i:=0--> <!--#4DLOOP ($i<4)--> <!--#4DEVAL $i--> <!--#4DEVAL $i:=$i+1--> <!--
#4DENDLOOP-->
```

以下の結果を生成します:

```
0
1
2
3
```

使用する4Dランゲージやバージョンに関係なく式が正常に評価される事を保証するためには、異なるバージョン間で名前が変化する可能性のある要素(コマンド、テーブル、フィールド、定数)に対してトークンシンタックスを使用することが推奨されます。例えば、**Current time**コマンドを挿入するためには、'**Current time:C178**' と入力します。この点の詳細な情報に付いては、**フォーミュラ内でのトークンの使用**を参照して下さい。

- **<!--#4DLOOP ポインター配列-->**
  この場合、4DLOOPタグは配列のときと同じように振るまいます:ポインターによって参照された配列の要素ごとにループを繰り返します。カレントの配列要素は、コードの部分が繰り返される度に増加していきます。
  このシンタックスは **PROCESS 4D TAGS** コマンドの引数に対して配列ポインターを渡した場合に有用です
  例えば:

```
ARRAY TEXT($array;2)
$array{1}:="hello"
$array{2}:="world"
$input:="<!--#4DEVAL $1-->"
$input:=$input+"<!--#4DLOOP $2-->"
$input:=$input+"<!--#4DEVAL $2->{$2->}--> "
$input:=$input+"<!--#4DENDLOOP-->"
PROCESS 4D TAGS($input;$output;"elements = ";->$array)
 // $output = "elements = hello world "
```

解釈エラーの場合、"*<!--#4DLOOP expression-->*: エラーの説明" が*<!--#4DLOOP -->* と *<!--#4DENDLOOP-->*の間のコンテンツの代わりに挿入されます。

以下のメッセージが表示されます:
- 予期しない式のタイプ (標準のエラー);
- テーブル名が正しくありません (テーブル名のエラー);
- 配列が必要です (変数が配列でないか、二次元配列が指定された);
- メソッドが存在しません;
- シンタックスエラー (メソッド実行時);
- アクセス権エラー (テーブルやメソッドにアクセスする権限がない)。
- 4DENDLOOP が必要です (*<!--#4DLOOP -->*が対応する*<!--#4DENDLOOP-->*で閉じられていない)。

4DTEXT、4DHTML、4DEVALにおける代替シンタックス

いくつかの既存の4D変換タグは、$-ベースのシンタックスを使用して表現する事ができます:
<!--#4dtag *expression*--> の代わりに、**$4dtag (*expression*)** という表記を使用する事ができます。

この代替シンタックスは、処理後の値を返すタグにおいてのみ使用可能です:

- 4DTEXT
- 4DHTML
- 4DEVAL

(その他のタグ、例えば4DIFや4DSCRIPTなどでは、通常のシンタックスを使用して書かなければなりません)。

例えば、以下のようなコードを:

```
<!--#4DEVAL(UserName)-->
```

このような表記で置き換える事ができます:

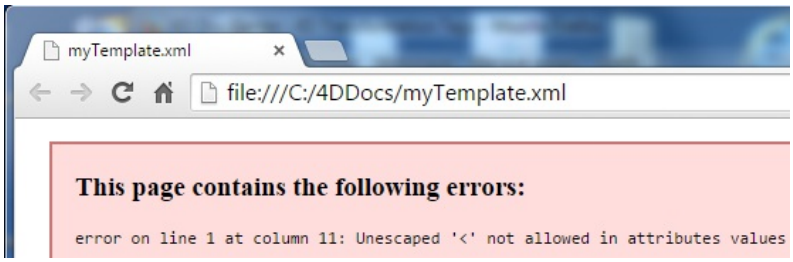
```
$4DEVAL(UserName)
```

このシンタックスの主な利点としては、**XML準拠のテンプレートを書く事ができる**という点です。一部の4Dデベロッパー

は、XML準拠のテンプレートを標準のXMLパーサーツールを使用して評価する必要があります。"<"文字はXML属性値としては無効なため、ドキュメントのシンタックスを破らずに4Dタグの"<!-- -->"シンタックスを使用する事はできませんでした。その一方で、"<"文字をエスケープしてしまうと、4Dがタグを正常に解釈できなくなってしまいます。

例えば、以下のコードは属性値の最初の"<"文字のためにXMLパースエラーを引き起こします:

```
<line x1="<!--#4DEVAL $x-->" y1="<!--#4DEVAL $graphY1-->"/>
```



$シンタックスを使用すると、パーサーによって以下のコードが評価されます:

```
<line x1="" y1=""/>
```

ここで、*$4dtag* と *<--#4dtag -->* は厳密には同じではないという点に注意して下さい。*<--#4dtag -->*とは異なり、*$4dtag*は4Dタグを繰り返し解釈する事はしません。$タグは常に一度だけ解釈され、その結果は標準テキストとして読まれます。

注: 繰り返し処理については、**再起的処理**の段落を参照して下さい。

この違いの理由は、悪意あるコードの侵入を防ぐためにあります。以下に説明されているように、ユーザーテキストを使用していてタグの不要な繰り返し処理を避けるためには、4DHTMLタグではなく4DTEXTタグの使用が強く推奨されます。4DTEXTを使用した場合、"<"などの特殊記号はエスケープされてしまうため、*<!--#4dtag expression -->* シンタックスを使用している4Dタグは全て元の意味を失ってしまいます。しかしながら4DTEXTは$記号はエスケープしないので、悪意あるコードの侵入を防ぐために*$4dtag (expression)* シンタックスにおける繰り返し解釈のサポートを取りやめる事にしました。

以下の例では、使用されるシンタックスとタグによる処理の結果の違いを表しています:

```
 // 例 1
myName:="<!--#4DHTML QUIT 4D-->" //悪意あるコードの侵入
input:="My name is: <!--#4DHTML myName-->"
PROCESS 4D TAGS(input;output)
 //4D は終了していまいます
```

```
 // 例 2
myName:="<!--#4DHTML QUIT 4D-->" //悪意あるコードの侵入
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
 //結果は"My name is:  & lt;!--#4DHTML QUIT 4D-->"
```

```
 // 例 3
myName:="" //悪意あるコードの侵入
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
 //結果は"My name is: ERROR: missing ')'"
```

シンタックスでは合致した引用符や括弧を閉じる事をサポートしているという点に注意して下さい。例えば、以下の(非現実的な)文字列を評価しなければならない場合:

```
String(1) + "\"(hello)\""
```

以下のように書く事ができます:

```
input:="$4DEVAL( String(1)+\"\\\"(hello)\\\"\")"
PROCESS 4D TAGS(input;output)
-->output is 1"(hello)"
```

使用上の注意

再起的処理

4D タグは繰り返し解釈されます。4Dは常に変換の結果を解釈しようとし、もし新しい変換が起きた際にはそれに伴い新しく解釈が実行され、取得した結果がこれ以上変換の必要がなくなるまで繰り返されます。例えば、以下のような宣言が合った場合:

```
<!--#4DHTML [Mail]Letter_type-->
```

もし[Mail]Letter_type text テキストフィールドにタグ(例えば*<!--#4DSCRIPT/m_Gender-->* など)が含まれていた場合、このタグは4DHTMLタグの解釈の後、それに伴って評価されます。

この強力な原則はテキスト変換に関連するほとんどの需要を満たす事ができます。しかしながら、これは場合に寄っては悪意のあるコードの侵入を許す事になる可能性があるという点に注意して下さい。この点についてのより詳細な情報については、以下の章を参照して下さい。

悪意あるコードの侵入を防止

4D変換タグは様々なタイプのデータを引数として受け入れます。テキスト、変数、メソッド、コマンド名、etc...。これらのデータが自分で書い たコードから提供される場合、そのインプットを自分でコントロールできるので、悪意あるコードの侵入のリスクは無いと言っていいでしょう。しかしながら、 データベースのコード扱うデータは、多くの場合外部ソース(ユーザー入力、読み込み、等)を通じて導入されたものです。
この場合、4DEVALや4DSCRIPTなどの変換タグを使用しない事が賢明です。なぜならこれらのタグはこういったデータを直接使って変数を評価するからです。
これに加え、繰り返しの原則(前章参照のこと)に従い、悪意あるコード自信が変換タグを含んでいる可能性もあります。この場合、4DTEXT タグを使用する必要があります。
例として、"Name"という名前のWebフォームフィールドがあり、ユーザーがそこに名前を入力しなければならない場合を考えてみましょう。この名前は*<!--#4DHTML vName-->* タグを使用してページ内に表示されます。もし"*<!--#4DEVAL **QUIT 4D**-->*" 型のテキストが名前の代わりに入力されたとしたら、このタグを解釈するとアプリケーションは終了してしまいます。
このリスクを避けるため、この場合にはシステム全体で4DTEXT タグを使用します。このタグは特殊HTML文字をエスケープするため、挿入された悪意ある再起的コードは、再解釈されることはありません。前の例でいうと、"Name"フィールドにはこの場合"*<!--#4DEVAL **QUIT 4D**-->*" が含まれ、これは変換されません。

"."を小数点として使用

v15 R4以降、4Dタグ(4DTEXT、4DVAR、4DHTML、4DHTMLVAR、そして4DEVAL)を使用した数値表現を評価する際には、4Dは常にピリオド文字(.)を使用するようになりました。今後はリージョン設定は無視される事になりました。

この機能により、4Dの言語設定とバージョンが異なっていてもメンテナンスが容易となり互換性が保たれます。

例えば、以下のコードはリージョン設定に関わらず使用可能です:

```
value:=10/4
input:="<!--#4DTEXT value-->"
PROCESS 4D TAGS(input;output)
 // 例え小数点が ',' に設定されていた場合でも、出力は常に2.5になります。
```

互換性に関する注意: ご自分のコードが4Dタグを使用した数値表現を、リージョン設定を遵守して評価する場合、そのコードは**String**コマンドを使用して適応させる必要があります:

- ピリオドを小数点として使用した*value*を取得する場合: *<!--#4DTEXT value-->*
- リージョン設定に基づいた小数点を使用した*value*を取得する場合: *<!--#4DTEXT String(value)-->*

WEB SET OPTION

WEB SET OPTION (selector ; value)

引数	型		説明
selector	倍長整数	➡	オプションコード
value	倍長整数, テキスト	➡	オプション値

説明

WEB SET OPTIONコマンドは4D Webサーバーの機能に関する様々なオプションのカレントの値を変更します。

*selector*引数には**Web Server**テーマの定数のうちひとつを指定し、*value*に新しい設定値を渡します:

定数	型	値	コメント
Web character set	倍長整数	17	**スコープ**: 4D ローカル, 4D Server **2セッション間で設定を保持**: Yes **説明**: データベースに接続するブラウザーとの通信に (ローカルモード4Dと4D Serverの) Webサーバーが使用する文字セット。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設定でも設定できます。 **値**: 取りうる値は、文字セットに関連するデータベースの動作モードによります。 • Unicode モード: アプリケーションがUnicodeモードで動作している場合、この引数に渡す値は文字セット識別子です。 (*MIBEnum*倍長整数または文字列。以下のアドレスを参照: http://www.iana.org/assignments/character-sets)。 以下は4D Webサーバがサポートする文字セットに対応する識別子のリストです: 4=ISO-8859-1 12=ISO-8859-9 13=ISO-8859-10 17=Shift_JIS 2024=Windows-31J 2026=Big5 38=euc-kr 106=UTF-8 2250=Windows-1250 2251=Windows-1251 2253=Windows-1253 2255=Windows-1255 2256=Windows-1256 **注**: IANAに定義されていない特別な文字セット (1258=x-mac-japanese) を使用することができます。これはWindows上ではコードページ10001に、Mac上ではkTextEncodingMacJapaneseにマップされています。 • *ASCII* 互換モード: 0: Western European 1: Japanese 2: Chinese 3: Korean 4: User-defined 5: Reserved 6: Central European 7: Cyrillic 8: Arabic 9: Greek 10: Hebrew 11: Turkish 12: Baltic
Web debug log	倍長整数	84	**スコープ:** ローカルWebサーバー **2セッション間で設定を保持:** No。ただしHTTPサーバーが再起動されても設定を保持(この場合新しいログファイルが使用されます)。 **説明:** 4D WebサーバーのHTTPリクエストログファイルの状態を設定または取得できるようにします。有効化された場合、"**HTTPDebugLog_*nn*.txt**"と命名されたこのファイルはアプリケーションの"Logs"フォルダに保存されます(*nn* にはファイル番号が入ります)。これはWebサーバーに関連した問題をデバッグするのに有用です。リクエストとレスポンスをrawモードで記録するからです。ヘッダーも含めて、リクエスト全体が記録されます。オプションとして、ボディ部分も記録することができます。 **値:** "wdl"の接頭辞が付いた定数のどれか一つ(このテーマ内のこれらの定数の詳細を参照して下さい) **デフォルト値**: 0 (有効化しない)
Web HTTP compression	倍長整数	50	**スコープ**: ローカル*Web*サーバー **2セッション間で設定を保持**: No **説明**: 4D HTTPサーバーで使用されるすべての圧縮されたHTTP通信 (クライアントのリクエスト、サーバーのレスポンス、WebおよびWebサービス) の圧縮レベル。このセレクターを使用すれば圧縮率を犠牲にして実行速度を速めるか、速度を優先して圧縮率を高めるかを選択できま

level	整数		す。値の選択は交換するデータのサイズやタイプに基づきます。*value*引数に1から9までの値を渡します。1は速度優先、9は圧縮率優先です。また-1を渡して圧縮速度と圧縮率の妥協点を指定できます。デフォルトの圧縮レベルは1 (速度優先) です。 **とりうる値**: 1 から 9 (1 = 速度優先, 9 = 圧縮優先) または -1 = 最適
Web HTTP compression threshold	倍長整数	51	**スコープ**: ローカルHTTPサーバー **2セッション間で設定を保持**: No **説明**: 最適化モードの内部的な4D Webサービス通信フレームワークにおいて、圧縮を行わないリクエストサイズの敷居値を設定できます。この設定は、小さなデータ交換時に圧縮を行うことによる、マシンの時間の浪費を避けるために有効です。 **とりうる値**: 任意の倍長整数値。*value*にはバイト単位でサイズを渡します。デフォルトで、圧縮の敷居値は1024バイトに設定されています。
Web HTTP TRACE	倍長整数	85	**スコープ**: ローカルWebサーバー **異なるセッション間で値を保持**: No **詳細**: 4D Web サーバー内のHTTP TRACEメソッドを無効化または有効化します。セキュリティ上の理由から、4D v15 R2以降、デフォルトで4D WebサーバーはHTTP TRACEリクエストをエラー405で拒否します(HTTP TRACEの無効化を参照して下さい)。必要であれば、この定数に値1を渡す事でそのセッションの間HTTP TRACEメソッドを有効化する事ができます。このオプションが有効化されると、4D WebサーバーはHTTP TRACEリクエストに対してリクエストライン、ヘッダー、そしてボディーを返信します。 **取り得る値**: 0 (無効化) または 1 (有効化) **デフォルトの値**: 0 (無効化)
Web HTTPS port ID	倍長整数	39	**スコープ**: 4D ローカル, 4D Server **2セッション間で設定を保持**: Yes **説明**: ローカルモード4Dおよび4D ServerのWebサーバがSSLによるセキュアな接続（HTTPSプロトコル）で使用するTCP ポート番号。HTTPS ポート番号はデータベース設定の"Web/設定"ページで指定できます。デフォルト値は443 (標準ポート番号) です。*value*引数に**TCP Port Numbers**テーマの定数を渡すこともできます。 **とりうる値**: 0~65535
Web inactive process timeout	倍長整数	78	**スコープ**: ローカルWebサーバー **2セッション間で値を保持**: No. しかしHTTPサーバーを再起動しただけの場合は保持されます。 **説明**: セッション管理のために使用されるプロセスのタイムアウトを設定します。タイムアウト後、プロセスは終了されます。 **取りうる値**: 倍長整数 (分) **デフォルト値**: 480分 (= 8時間) (0を渡すとデフォルト値に設定されます)
Web inactive session timeout	倍長整数	72	**スコープ**: ローカルWebサーバー **2セッション間で値を保持**: No. しかしHTTPサーバーを再起動しただけの場合は保持されます。 **説明**: セッション管理のために使用されるcookieのタイムアウトを設定します。 **取りうる値**: 倍長整数 (分) **デフォルト値**: 480分 (= 8時間) (0を渡すとデフォルト値に設定されます)

Web IP address to listen	倍長整数	16	**スコープ**: 4D ローカル, 4D Server
			2セッション間で設定を保持: Yes
			説明: ローカルモード4Dおよび4D ServerのWebサーバがHTTPリクエストを受信するIPアドレス。デフォルトで、特定のアドレスは定義されていません（*value*=0）。この引数はデータベース設定で設定できます。
			
			Web IP Address to listenセレクターは、コンパイルして4D Volume Desktop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)。
			
			*value*引数には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(Web IP address to listen;$addr)`

Web keep	倍長		**スコープ**: ローカルWebサーバー **2セッション間で設定を保持**: No. しかしHTTPサーバー再起動後も設定は有効。 **説明**: 4Dによる自動セッション管理モード (**Webセッション管理**) の有効/無効を設定する。

Web keep session	長整数	70	**取りうる値**: 1 (有効) / 0 (無効) **デフォルト値**: v13で作成されたデータベースでは1、変換されたデータベースでは0。このモードはリモートモードで一時的なコンテキストの再利用メカニズムも有効にする点に留意してください。このメカニズムに関する詳細は**Webサーバー設定**を参照してください。
Web log recording	倍長整数	29	**スコープ**: 4D ローカル, 4D Server **2セッション間で設定を保持**: Yes **説明**: ローカルモード4Dまたは4D ServerのWebサーバーが受け取るWebリクエストの記録を開始または停止します。デフォルト値は0（リクエストを記録しない）です。 Web リクエストのログは"logweb.txt"という名前のテキストファイルに保存されます。このファイルは自動でストラクチャファイルと同階層のLogsフォ ルダー内に作成されます。このファイルのフォーマットは、渡した値により決定されます。Webログファイルフォーマットに関する詳細は**Webサイトに関する情報**を参照してください。 このファイルはデータベース設定の"Web/ログ"ページからも有効にできます。 **とりうる値**: 0 = 記録しない (デフォルト), 1 = CLFフォーマットで記録, 2 = DLFフォーマットで記録, 3 = ELFフォーマットで記録, 4 = WLFフォーマットで記録 **警告**: フォーマット3および4はカスタムフォーマットであり、記録する内容を事前にデータベース設定で定義しなければなりません。事前定義せずにこれらのフォーマットを使用した場合、ログファイルは作成されません。
Web max concurrent processes	倍長整数	18	**スコープ**: 4D ローカル, 4D Server **2セッション間で設定を保持**: Yes **説明**: ローカルモード4Dならびに4D ServerのWebサーバーでサポートされる、任意のタイプの同時Webプロセス上限数を厳密に設定。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）をすべての新しいリクエストに返します。 このパラメーターにより、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、4D Webサーバが飽和状態になることを防ぐことができます。このパラメーターはデータベース設定でも設定できます。 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プロセス情報を監視する方法です。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を監視します。 **値**: 10から32 000までの任意の数。デフォルト値は100。
Web max sessions	倍長整数	71	**スコープ**: ローカルWebサーバー **2セッション間で設定を保持**: No. しかしHTTPサーバー再起動後も設定は有効。 **説明**: 4Dの自動セッション管理下のセッション上限数。設定した上限に達すると、もっとも古いセッションが閉じられます。 **取りうる値**: 倍長整数値 **デフォルト値**: 100 (0を渡すとデフォルト値が設定されます)
Web maximum requests size	倍長整数	27	**スコープ**: 4D ローカル, 4D Server **2セッション間で設定を保持**: Yes **説明**: Webサーバに処理を許可する受信HTTPリクエスト (POST) の最大サイズ (バイト単位)。デフォルト値は2,000,000 (2MBより少し少ない値) です。最大値（2,147,483,648）を渡すと、実際上は制限がなくなります。 この制限は、受信するリクエストが大きすぎるためにWebサービスが飽和してしまうことを回避するために使用します。リクエストがこの制限に達すると、4D Webサービスはリクエストを拒否します。 **とりうる値**: 500,000～2,147,483,648.
Web port ID	倍長整数	15	**スコープ**: 4D ローカル, 4D Server. **2セッション間で設定を保持:** No **説明:** ローカルモードの4Dと4D Serverで、4D Web server が使用するTCPポート番号を設定または取得します。デフォルトではこの値は80です。TCPポート番号は、データベース設定の"Web/設定"タブ内にて設定できます。*value* p引数には、**TCP Port Numbers** テーマ内にある定数の一つを使用することができます。このセレクターは、4D Desktop を使用して組み込み・コンパイルされた4D Web Server フレームワーク内(デザイン環境へのアクセスがない状態)において有用です。 **取り得る値:** TCPポート番号についての詳細な情報に関しては、 **Webサーバー設定** セクション

			を参照して下さい。 **デフォルトの値:** 80
Web session cookie domain	倍長整数	81	**スコープ:** ローカルWeb server **2セッション間で設定を保持:** No、ただしHTTPサーバーが再起動した場合でも有効なままです。 **説明:** セッションCookieの"ドメイン"フィールドの値を設定または取得します。このセレクター(とセレクター82)は、セッションCookieのスコープを管理するのに有用です。例えば、このセレクターに、 "/*.4d.fr" という値を設定した場合、クライアントは、リクエストが ".4d.fr" のドメイン宛てだった場合にのみCookieを送ります。これにより、外部の静的なデータをホストするサーバーを除外することができます。 **取り得る値:** テキスト
Web session cookie name	倍長整数	73	スコープ: ローカルWebサーバー 2セッション間で設定を保持: No. しかしHTTPサーバー再起動後も設定は保持される。 説明: 4Dの自動セッション管理機能で使用されるcookieのname属性値。 取りうる値: テキスト デフォルト値: "4DSID" (空文字を渡すとデフォルト値が設定されます。)
Web session cookie path	倍長整数	82	**スコープ:** ローカル Web server **2セッション間で設定を保持:** No、ただしHTTPサーバーを再起動後も有効。 **説明:** セッションCookieの"パス"フィールドの値を設定・または取得します。このセレクター(とセレクター81)は、セッションCookieのスコープを管理するのに有用です。例えば、このセレクターに"/4DACTION"という値を設定した場合、クライアントは、4DACTION から始まる動的なリクエストに対してのみCookieを送り、ピクチャやスタティックなページ等に対しては送りません。 **取り得る値:** テキスト
Web session enable IP address validation	倍長整数	83	**スコープ**: ローカルWebサーバー **2セッション間で設定を保持**: No **説明**: セッションcookieに対するIPアドレス認証を有効化・または無効化します。セキュリティ上の理由から、4D Webサーバーはセッションcookieを含んでいるそれぞれのリクエストのIPアドレスをデフォルトでチェックし、そのアドレスが、cookieを作成するのに使用されたIPアドレスと異なる場合にはリクエストを拒否します。一部の特定のアプリケーションにおいては、この認証を無効化して、IPアドレスが合致しないcookieも受け入れたい場合があるかもしれません。例えばWifiと3G/4Gネットワークを切り替えるモバイルデバイスでは、IPアドレスは合致しません。この場合、このオプションに0を渡してIPアドレスが変わった時でもクライアントがWebセッションを引き続き利用できるようにします。ただしこの設定はアプリケーションのセキュリティレベルを下げることになることに注意して下さい。 これが変更された際には、その設定は直ちに反映されます(HTTPサーバーを再起動する必要はありません)。 **とり得る値**: 0(無効化)または1(有効化) **デフォルト値**: 1 (IP アドレスはチェックされます)

*selector*引数にWeb debug logを使用する場合、*value*引数に以下の定数のうちどれか一つを渡す事ができます:

定数	型	値	コメント
wdl disable	倍長整数	0	Web HTTP debug log は無効化されています
wdl enable with all body parts	倍長整数	7	Web HTTP debug log はレスポンスとリクエスト両方をボディー部に含めた状態で有効化されます
wdl enable with request body	倍長整数	5	Web HTTP debug log はリクエストのボディー部のみ含めた状態で有効化されます
wdl enable with response body	倍長整数	3	Web HTTP debug log はレスポンスのボディー部のみを含めた状態で有効化されています。
wdl enable without body	倍長整数	1	Web HTTP debug log はボディ部なしで有効化されています(この場合ボディ部のサイズは提供されます)

例題

HTTP デバッグログをボディ部分なしで有効化する場合を考えます:

```
WEB SET OPTION(Web debug log;wdl enable without body)
```

ログエントリーは次のようになります:

```
# REQUEST
# SocketID: 1592
# PeerIP: 127.0.0.1
# PeerPort: 54912
# TimeStamp: 39089388
GET /4DWEBTEST HTTP/1.1
Connection: Close
Host: 127.0.0.1
User-Agent: 4D_HTTP_Client/0.0.0.0

# RESPONSE
# SocketID: 1592
# PeerIP: 127.0.0.1
# PeerPort: 54912
# TimeStamp: 39089389 (elapsed time: 1 ms)
HTTP/1.1 200 OK
Accept-Ranges: bytes
Connection: close
Content-Length: 3555
Content-Type: text/plain; charset=UTF-8
Date: Tue, 20 Jan 2015 10:51:29 GMT
Expires: Tue, 20 Jan 2015 10:51:29 GMT
Pragma: no-cache
Server: 4D/14.6.0

[Body Size: 3555]
```

名前の変更、テーマの変更

On Web Session Suspend データベースメソッド

以前の名前	新しい名前	コメント
On Web Session Suspend データベースメソッド	**On Web Close Process** データベースメソッド	**注:** 4D Mobile セッション (複数プロセスを生成できます) を利用している場合、Web プロセスを閉じる度に **On Web Close Process** データベースメソッドが呼び出されるため、4D Mobile セッションプロセスによって生成されたあらゆるデータ (変数、セレクション、他) をこの時点で保存することができます。

リストボックスの定数の改名

以前は"**Listbox**..."という接頭辞が付いていたリストボックスの定数は、"**lk**..."という接頭辞に変更になりました。**List Box**と**Listbox Footer Calculation**テーマをご覧下さい。

List Box

定数	型	値	コメント
lk add to selection	倍長整数	1	選択された行は既存の選択行に追加されます。指定した行が既存の選択に属している場合、コマンドは何も行いません。
lk all	倍長整数	0	コマンドはすべてのサブレベルに作用します (引数省略時のデフォルト値)。
lk background color	倍長整数	1	
lk background color array	倍長整数	1	
lk break row	倍長整数	2	コマンドは*row* と *column*引数で指定された"セル"に属するサブレベルに作用します。これらの引数は標準モードのリストボックスの行および列番号を表すことに留意してください。階層表現ではありません。*row* と *column* 引数が省略されると、コマンドは何も行いません。
lk control array	倍長整数	3	
lk display footer	倍長整数	8	0 = 非表示 1 = 表示
lk display header	倍長整数	0	0=非表示, 1=表示
lk display hor scrollbar	倍長整数	2	0=非表示, 1=表示
lk display ver scrollbar	倍長整数	4	0=非表示, 1=表示
lk font color	倍長整数	0	
lk font color array	倍長整数	0	
lk footer height	倍長整数	9	高さ (ピクセル)
lk header height	倍長整数	1	高さ (ピクセル)
lk hor	倍長		

scrollbar height	長整数	3	高さ (ピクセル)
lk hor scrollbar position	倍長整数	6	カーソルの位置 (ピクセル)
lk inherited	倍長整数	-255	
lk last printed row number	倍長整数	0	*info*に印刷された最後の行番号を返します。これにより次に印刷される行の番号が分かります。リストボックスに非表示行が存在したり**OBJECT SET SCROLL POSITION**コマンドが呼び出されていたりすると、返される値は実際に印刷された行数よりも、大きくなることがあります。例えば行番号1, 18そして20が印刷されると、*info*には20が返されます。
lk level	倍長整数	3	コマンドは*level*列に対応するすべてのブレーク行に作用します。この引数は標準モードのリストボックスの列番号を指定し、階層表現を考慮しません。*level*引数が省略されると、コマンドはなにも行いません。
lk lines	倍長整数	1	高さを行数で指定。4Dはフォント設定に応じて高さを計算します。
lk pixels	倍長整数	0	高さをピクセルで指定 (デフォルト)。
lk printed height	倍長整数	3	*info*に実際に印刷されたオブジェクトの高さをピクセル単位で返します (ヘッダーや線等を含む)。印刷する行数がリストボックスの高さに満たない場合、高さは自動で減らされます。
lk printed rows	倍長整数	1	さいごの**Print object**最後のコマンド呼び出し時に実際に印刷された行数を*info*に返します。この数値には階層リストボックスの場合に追加されたブレーク行も含まれます。例えばリストボックスに20行あり、奇数行が隠されている場合、*info*は10になります。
lk printing is over	倍長整数	2	リストボックスの最後の (表示) 行が印刷されたかどうかを示すブール値を*info*に返します。True = 行は印刷された; そうでなければFalse。
lk remove from selection	倍長整数	2	指定された行は既存の選択行から取り除かれます。指定した行が既存の選択に属さない場合、コマンドは何も行いません。
lk replace selection	倍長整数	0	選択された行が新しい選択行となり、既存のものと置き換えられます。このコマンドは、ユーザが行をクリックした場合と同じ結果になります。これは (*action* 引数が省略された時の) デフォルトの動作です。
lk row height array	倍長整数	4	(4D View Pro license required)
lk row is disabled	倍長整数	2	対応する行は無効化されています。テキストとチェックボックスなどのコントロール類は暗くなっているかグレーアウトされています。入力可能なテキスト入力エリアは入力可能ではありません。デフォルト値:有効化
lk row is hidden	倍長整数	1	対応する行は非表示です。行を非表示にするのはリストボックスでの表示にのみ影響します。非表示の行は配列内には存在し、プログラミングを通して管理可能です。ランゲージコマンド(具体的には**LISTBOX Get number of rows**または**LISTBOX GET CELL POSITION**)は行の表示/非表示のステータスを考慮しません。例えば10行あるリストボックスの、最初の9行が非表示になってい

hidden	整数		た場合、**LISTBOX Get number of rows**は10を返します。ユーザーからの視点では、リストボックス内での非表示行の存在というのは視覚的には認識できません。表示されいている行のみが(例えばすべてを選択コマンドなどで)選択可能です。デフォルト値:表示
lk row is not selectable	倍長整数	4	対応する行は選択可能になっていません(ハイライトができません)。入力可能なテキスト入力エリアは"シングルクリック編集"オプションが有効になっていない限り入力可能ではありません。しかしながらチェックボックスなどのコントロールとリストは機能しています。この設定はリストボックスセレクションモードが"なし"の場合には無視されます。デフォルト値:選択可能
lk selection	倍長整数	1	コマンドは選択されたサブレベルに作用します。
lk style array	倍長整数	2	
lk ver scrollbar position	倍長整数	7	カーソルの位置 (ピクセル)
lk ver scrollbar width	倍長整数	5	幅 (ピクセル)

廃止予定の機能

廃止予定のコマンド

データベース内で使用されている廃止予定のコマンドを探すには、編集メニューから**デザインモードを検索**を選択するか、ツールバーの**デザインモードを検索**エリアを使って、廃止予定コマンドの接頭辞である "_o_" を検索します。廃止予定のコマンドは新しいコマンドや機能に代替することが強く推奨されますが、"削除" されていないかぎり、これらの廃止予定コマンドは機能することにも留意してください。

廃止予定コマンドの完全な一覧については、**ランゲージ: 廃止予定または削除されたコマンド** の章を参照してください。

4D v16で廃止予定となったコマンド

以前の名前	新しい名前	コメント
C_GRAPH	**_o_C_GRAPH**	4D v15 R5 でこのコマンドは廃止予定として改名され、4D コマンドの一覧には表示されなくなりました。4D v14以降、グラフエリア型の変数は廃止されており、サポートされていません。代わりにピクチャー変数を使用する必要があります(**GRAPH** 参照)。
INTEGRATE LOG FILE	**_o_INTEGRATE LOG FILE**	**_o_INTEGRATE LOG FILE** コマンドは廃止予定として定義され、代わりに **INTEGRATE MIRROR LOG FILE** の使用が推奨されます。**_o_INTEGRATE LOG FILE** コマンドと違い、新しい **INTEGRATE MIRROR LOG FILE** コマンドはカレントログファイルを統合されたものと置き換えることなく、データベースのカレントログファイルは使用され続けます。つまり、統合中に発生した変更はすべてカレントログファイルに記録されます。
Open external window	**_o_Open external window**	**_o_Open external window** コマンドは 64-bit版の 4D および 4D Server では機能しません。また、今後リリースされる 4D のバージョンではサポートされなくなります。

廃止予定の 4D Pack コマンド

4D Pack コマンド	代替コマンド	廃止予定となったバージョン	現在の状況
_o_AP BLOB to print settings	**BLOB to print settings**	v16	廃止予定
_o_AP Print settings to BLOB	**Print settings to BLOB**	v16	廃止予定
_o_AP Is picture deprecated	**GET PICTURE FORMATS**	v16	廃止予定
_o_AP NORMAL SCREEN, _o_AP FULL SCREEN	-	v16	廃止予定
_o_AP Get field infos, _o_AP Get table infos	-	v16	廃止予定
_o_AP Get tips state, _o_AP SET TIPS STATE	-	v16	廃止予定

_o_C_GRAPH

_o_C_GRAPH ({method ;} variable {; variable2 ; ... ; variableN})

引数	型		説明
method	文字	⇒	メソッド名 (オプション)
variable	変数	⇒	宣言する変数名

互換性に関する注意

4D v14以降、グラフエリア型の変数は廃止されており、サポートされていません。代わりにピクチャー変数を使用する必要があります(**GRAPH** 参照)。

BLOB to print settings

BLOB to print settings (printSettings {; params}) -> 戻り値

引数	型		説明
printSettings	BLOB	➡	印刷設定を格納したBLOB
params	倍長整数	➡	0 = 部数と印刷範囲に関して、BLOB に保存された値を復元 ; 1 = デフォルト値にリセット
戻り値	倍長整数	⬅	1 = 処理に成功, 0 = カレントのプリンターがない, -1 = 引数が不正, 2 = プリンターが変更された

説明

BLOB to print settings コマンドは、4Dの現プリンタ設定を *printSettings* BLOBに格納された内容で置き換えます。このBLOBは **Print settings to BLOB** または **_o_AP Print settings to BLOB** 4D Pack コマンドで生成されていなければなりません (後述参照)。

params パラメーターには、"部数" および "印刷範囲" の基本設定の扱いを指定します:

- 0 または省略: BLOB に保存されている値が採用されます
- 1: これらの値をデフォルト値にリセットします (部数: 1; 印刷範囲: すべて)

新しいプリント設定はカレントプリンターに対して適用され、> 引数なしで呼び出された **PAGE SETUP** や **SET PRINT OPTION**、**PRINT SELECTION** などのコマンドが設定を変更するまで、すべてのセッションで有効です。具体的には **PRINT SELECTION**, **PRINT LABEL**, **PRINT RECORD**, **Print form** と **QR REPORT** コマンドのほか、4Dのメニューコマンド (デザインモード含む) で、この印刷設定使用されます。

BLOB to print settingsで定義した設定を保持するためには、**PRINT SELECTION**, **PRINT LABEL**, および **PRINT RECORD** コマンドは > 引数付きで呼び出さなければなりません。

このコマンドは次のいずれかの値を返します:

- -1: BLOB が不正です
- 0: カレントプリンターがありません (この場合、コマンドはなにもしません)
- 1: BLOB は正常にロードされました
- 2: BLOB は正常にロードされましたが、カレントプリンター名が変更されました(*)
  注: BLOB が **_o_AP Print settings to BLOB** 4D Pack コマンドによって生成されている場合には、当該BLOB にその情報が含まれていないため、プリンター名が実際には変更されていなくても、常に (2)が返されます。

(*) プリント設定は BLOB 生成時に選択されていたカレントプリンターに依存します。この印刷設定を、同じモデルの異なるプリンターに対して適用することができます。違うプリンターの場合には、共通の設定のみがロードされます。

Windows / OS X

printSettings BLOB はどちらのプラットフォームでも保存およびロードすることが可能ですが、印刷設定には共有されているものと、ドライバーやシステムバージョンに依存する特有のものがあります。そのため、同じ *printSettings* BLOB を異なるプラットフォームに流用した場合には、ロードされる情報が不完全の場合があります。異なるプラットフォームを併用する環境において印刷設定の復元を最適化するには、それぞれのプラットフォーム用に (つまり二つの) *printSettings* BLOB を管理することが推奨されます。

4D Pack コマンドとの互換性

4D Pack の旧 **_o_AP Print settings to BLOB** コマンドで生成された印刷設定 BLOB は **BLOB to print settings** で使用できますが、**Print settings to BLOB** で保存した BLOB を **_o_AP BLOB to print settings** で使用することはできません。

The **BLOB to print settings** コマンドは、**_o_AP Print settings to BLOB**コマンドに比べてより多くの印刷情報を格納する事ができます。

例題

4D の現在の印刷コンテキストに、以前ディスクに保存したプリント設定を適用します:

```
C_BLOB(curSettings)
DOCUMENT TO BLOB(Get 4D folder(Active 4D Folder)+"current4Dsettings.blob";curSettings)
 // current4Dsettings は Print settings to BLOB で生成されたものです
$err:=BLOB to print settings(curSettings;0)
```

```
Case of
   :($err=1)
 // 印刷設定は正常にロードされました
   :($err=2)
      CONFIRM("プリンターが変更されました。\n\n設定を確認しますか?")
      If(OK=1)
         PRINT SETTINGS
      End if
   :($err=0)
      ALERT("カレントプリンターがありません。")
   :($err=-1)
      ALERT("不正な設定ファイルです。")
End case
```

無効化された機能

- **_o_AP Save BMP 8 bit**: コマンドが削除されました
- 特定の関数は 64-bit版の 4D で無効化されています。詳細については次のドキュメントを参照ください:
  - **64-bit版の 4D Developer Edition** の **無効化された機能** および **サポートされない機能** の章
  - 印刷オプションについては **印刷** の章

64-bit版の 4D Developer Edition

4D v16では、4Dから**OS X用4D Developer Edition** および**OS X用64-bit版4D Volume Desktop**が提供されます。また、Windows 用の 4D Developer Edition 64-bit および 4D Volume Desktop 64-bit のプレビュー版も提供されます。

注:

- Windowsにおいては、64-bit版の4D Developer および4D Volume Desktopは**プレビュー版**にて提供されています。
- 4Dでは**Windows用4D Server 64bit版の使用** および **OS X用64bit版4D Serverの使用**が可能である点も忘れないで下さい。

これらの新しい製品によって 4D スタンドアロンアプリケーションと 4D リモートアプリケーションは 64-bit版 OSの利点を最大限活用することができます。64-bit テクノロジーの主な利点は、より多くのRAMメモリを割り当てられることです。

大幅な改定内容にもかかわらず、4D 64-bit版アプリケーションは今までの 4D データベースと高い互換性を持ちます。しかしながら、最新技術を利用するにあたり、いくつかの機能を更新し、またいくつかの機能についてはサポートを終了する必要が生じました。変更点についての詳細は**64-bit版のみの機能**にまとめてあります。x

システム要項

64-bit版4D Developerを動かすには、以下のスペックが必要です:

	Windows	**OS X**
OS	Windows 7 またはそれ以上(64-bit版)	OS X 10.10 (Yosemite) またはそれ以上
RAM	8 GB	8 GB

お使いのバージョンの4DがどのOSと対応しているのかを探すためには、4D Web サイトにある対応早見表を参照して下さい。

アーキテクチャー

64-bit アーキテクチャーを想定した 4Dアプリケーションは、この環境専用のバージョンとなります。言い換えるとそれらは32-bit版OSでは実行できません。

インタープリタモードでは、64-bit版・32-bit版のどちらのアプリケーションでも同じ4Dデータベースを開くことができます(サーバー・ローカルを問いません)。どちらのアプリケーションを使おうと、開発の段階では一切違いはありません (ただし以下の制限を除く)。

コンパイルモードでは、データベースは適切なプロセッサー向けにコンパイルされている必要があります。つまり、64-bit版アプリケーションで開くためには64-bitの、32-bit版アプリケーションで開くためには32-bitのプロセッサー向けにでコンパイルしなければなりません。32-bit用にコンパイルされた、インタープリタコードを含んでいないデータベースを 64-bitの4Dアプリケーションで開くことはできず、その逆もまたしかりです。データベースはどちらか特定のアーキテクチャーだけにコンパイルすることもできますし、両方にコンパイルすることもできます。コンパイルについての詳細は、次の章を参照してください。

以下の一覧は様々な4D実行環境と、そのデータベースのコードとの互換性をまとめたものです:

	利用可能コード	**32-bit 4D**	**64-bit 4D**
32-bit 4D Server	インタープリタ	OK	OK(*)
	32-bit コンパイル済みのみ	OK	-
	32-bit と 64-bit コンパイル済み	OK	OK(*)
64-bit 4D Server	インタープリタ	OK	OK(*)
	64-bit コンパイル済みのみ	-	OK(*)
	64-bit と 32-bit コンパイル済み	OK	OK(*)
Local database	インタープリタ	OK	OK
	32-bit コンパイル済みのみ	OK	-
	64-bit コンパイル済みのみ	-	OK
	32-bit と 64-bit コンパイル済み	OK	OK

(*) 64-bit版 4D は旧式ネットワークレイヤーを利用できないため、64-bit版 4D から 32-bit版 4D Server (Windows・OS X ともに) および Windows用 64-bit版 4D Serverに接続する場合においては、サーバー側において***ServerNet*** ネットワー

クレイヤーが有効化されていることを確認する必要があります。なお、Mac OS用 64-bit版 4D Server はそもそも旧式ネットワークレイヤーをサポートしていないため、この確認は不要です。より詳細な情報については、**新しい ServerNet ネットワークレイヤー(互換性)** の章を参照してください。

コンポーネントとプラグイン

以下のプラグインおよびコンポーネントは、32-bit版および 64-bit版の 4D Server、4D Developer Edition、そして 4D Volume Desktop によってロード・実行することが可能です:

- 4D for OCI
- 4D Internet Commands(*)
- 4D ODBC Pro
- 4D Pack
- 4D Progress
- 4D SVG
- 4D Widgets
- 4D Write Pro Interface

(*)Windows用4D Developer Edition 64-bit版では利用不可(プレバージョン)

4D View と 4D Write

4D View と 4D Write は 32-bit プラグインのため、これらを使用できるのは 32-bit 版の 4D のみとなりますが、64-bit版でも次のことが可能です:

- 64-bit版で開発を行って 32-bit版用にコンパイル / 運用するのを可能にするため、64-bit版 (OS X と Windows)でもプレースホルダーが提供されています。
- 4D Server 64-bit Windows では、これらのプラグインはオフスクリーンエリアに限り実行することができます。

64-bit版のみの機能

この章では現在の Windows と OS X 用の 64-bit版 4D Developer Edition の実装についての詳細を記載しています。

更新された機能

64-bit アーキテクチャーをサポートするために多くの4D機能とダイアログが更新、あるいは書き換えられました。ほとんどの変更は見えないところで行われており、 32-bit版と同じように動作します。しかし、一部のエディターについては 32-bit版とは異なる形に更新され、印刷などの基礎的な機能についても変更がありました。

機能	対象となる 4D のバージョン	補足
クイックレポートエディター	OS X & Win	完全に書き換えられました。**クイックレポート(64-bit版)** 参照。
ラベルエディター	OS X & Win	完全に書き換えられました。**ラベルエディター (64-bit)** 参照。
グラフ	OS X & Win	**GRAPH** コマンドにグラフ設定を指定する Object 型の引数を渡せます。
フォームエディタープロパティリスト	OS X & Win	新デザイン + 新機能。**プロパティリスト (新デザイン)** 参照。
印刷	OS X & Win	"印刷"ダイアログボックスの更新(標準のシステムダイアログボックスを使用)。"印刷設定"ダイアログボックスは今後は自動的には表示されません(**PRINT SETTINGS**コマンドを参照して下さい)。**SET CURRENT PRINTER** と **SET PRINT OPTION**コマンドの修正。
読み込み / 書き出しダイアログボックス	OS X & Win	32-bit版と同じに動作しますが、XML書き出のXSLサポート(XSLT はサポートされていません。後述参照)と、ODBCソース経由の場合は除きます(無効化、後述参照)。

無効化された機能

64-bit版 4D Developer Edition では一部の機能が無効化されています:

機能/テクノロジー	対象となる4Dのバージョン	補足
ODBC ソースを介した読み込み / 書き出し	OS X & Win	無効化
クイックレポートのクロステーブル	OS X & Win	無効化
クイックレポートエディター：境界線	OS X & Win	無効化
ラベルエディターの標準コード	OS X & Win	無効化
Webエリア内での統合 Web Kit	OS X & Win	無効化 (オプション使用時にはシステム Web エンジンに自動切替; OS X では 4D $4D メソッドへのアクセスを維持)
4D Internet Commands	Win	現在は利用不可

サポートされない機能

以下の機能またはテクノロジーは廃止され、64-bit版 4D Developer Editionではサポートされません:

機能/テクノロジー	対象となる4Dのバージョン	コメント
XSLT と Xalan	OS X & Win	**_o_XSLT APPLY TRANSFORMATION**、**_o_XSLT SET PARAMETER**、そして **_o_XSLT GET ERROR** は何もしません。代わりに **PROCESS 4D TAGS** を使用するか、PHP *libxslt* モジュールを使用してください。
PICT フォーマット	OS X & Win	'サポートされていない画像フォーマットです' ピクチャ＋画像の拡張子が代わりに表示されます。PICT フォーマットは 4D 全体において廃止予定となっています。**PICT フォーマットのピクチャー** も参照してください。
QuickTime	OS X & Win	QuickTime のサポートが廃止されました。QuickTime support データベースパラメーターは無視されます。
cicn アイコン	OS X & Win	**GET ICON RESOURCE** はサポートされていません。エラーを返します。
データベース .RSR ファイル	OS X & Win	データベース .RSR ファイルは自動的には開かれません。**Open resource file** を使用する必要があります。
書き込み可能リソースファイル	OS X & Win	**_o_Create resource file** はサポートされていません。リソースファイルは読み込みのみモードでしか開けません。
_o_Font number	OS X & Win	このコマンドはサポートされていません。エラーを返します。
_o_Open external window	OS X & Win	このコマンドはサポートされていません。エラーを返します。
旧式ネットワークレイヤー	OS X & Win	*ServerNet* のみがサポートされています。
ASCII 互換モード	OS X & Win	Unicode モードのみがサポートされています。
4D Write と 4D View プラグイン	Win	旧式プラグインは 64-bit版の 4D とは互換性がありません。**4D Write Proリファレンス** と **4D View Pro**を使用して下さい。
AP Print settings to BLOB / AP BLOB to print settings (4D Pack)	OS X & Win	**Print settings to BLOB** / **BLOB to print settings** コマンドによって置き換えられました。
OLE ツール	Win	サポートされていません

32-bitから64-bitへのアップグレード

OS X上で既存の4Dアプリケーションを32-bit版から64-bit版へとアップグレードするには、多少の準備が必要となります。

もしお使いのアプリケーションがWindowsあるいはOS X用の4D Server 64-bit版で実行できるなら、大部分の準備は既に済んでいると言ってよいでしょう。64-bit版のデスクトップアプリケーションの場合は、いくつかの追加の段階を踏む必要があるかもしれません。この章ではアップグレードの前と後、両方で必要なポイントをすべて検証するのに手助けするステップ・バイ・ステップ形式のチェックリストを用意しています。

製品の64-bit版への移行にともない、いくつかの機能がアップデートされていたり、無効化、あるいは廃止予定と宣言されています。すべての詳細はこちらのページにまとめてあります: **64-bit版のみの機能**

注: 通常のアップグレードプロセスと同様、大規模なステップの前にMSCを使用して検証を行い、データとストラクチャーの両方に問題がないことを確認するのが良いでしょう。

プラグインをチェックする

最初にすることは、プラグインがあれば、それらを64-bit版へとアップグレードすることです:

- **4D プラグイン:**
  4D Write と4D Viewを除き、すべてのプラグインは既に64-bit版で動作しています。
  - お使いのアプリケーションが4D Writeを使用している場合、そのコードを4D Write Proへと移行することを検討する必要があります。このとき、既存の32-bit版のコードはそのまま保持しておき、並行して新しい64-bitベースのモジュールを4D Write Proで作り始めるのが良いでしょう。
  - お使いのアプリケーションが4D Viewを使用する場合、4D View Pro機能あるいは別の選択肢を使用する必要があります。
- **サードパーティーのプラグイン:**
  プラグインのプロバイダーに連絡をして、64-bit版を取得してください。

32-bit版においてアップグレードに必要な準備

1. お使いのアプリケーションを最新の32-bitリリース (例えば4D v15 R5 32-bitなど) にアップグレードしてください。
2. Unicodeモードが有効化されていることを確認してください。
3. あらゆるPICT/cicn/QuickTimeピクチャーを変換してください。
   データ内にある廃止予定のピクチャー形式を検出するには、**GET PICTURE FORMATS** コマンドが使用できます。
   また、データベースのストラクチャー内にあるサポートされていないピクチャーもすべて置き換える必要があります。
   MSCで検証をすることで、resources 内にあるピクチャーや3Dボタン用のピクチャーや静的なピクチャーの中で古いピクチャーを検知することができます。
4. XSLTベースの機能 (**_o_XSLT APPLY TRANSFORMATION**、**_o_XSLT SET PARAMETER** あるいは **_o_XSLT GET ERROR** コマンド)を、例えば **PROCESS 4D TAGS** コマンドで置き換えてください。
5. **_o_Font number** の呼び出しをフォント名での呼び出しに置き換えてください。
6. リソースファイルを作成あるいは変更するコードをすべて削除してください。

この時点で、お使いのデータベースを 64-bit版の4Dで開く準備ができました。

データベースを64-bit版で開いてチェックする

1. お使いのアプリケーションを 4D Developer Edition 64-bit版で開いてください。
2. Webエリアにおいて統合された WebKit を使用している場合、これらは自動的にシステムエンジンに切り替わっているため、チェックしてみてください (4Dメソッド($4d) へのアクセスは引き続き有効です)。
3. コード内で **SET PRINT OPTION** コマンドの Mac spool file format option を使用している場合、それらを **SET CURRENT PRINTER** と Generic PDF driver 定数の組み合わせの呼び出しで置き換える必要があります。
4. ラベルエディターの呼び出しと使用をチェックしてください (**ラベルエディター (新デザイン)** 参照)。
5. クイックレポートの呼び出しと使用をチェックしてください (**クイックレポートエディター (新デザイン)** 参照)。

これでお使いのアプリケーションは完全に64-bit版で使用できるようになり、4D 64-bit版の新機能も使用できるようになりました。

64-bit版の機能の利点

- **64-bitアーキテクチャー**はデータベースキャッシュの限度を引き上げます。より大きいキャッシュを使用するだけでデータベースのパフォーマンスを向上させることができます。
  プリエンプティブプロセス、アニメーション付きフォームオブジェクト、新しい印刷機能など、**強力な64-bit機能**を使用してみましょう。
  4D Runtime Volumeライセンス64-bit版でアプリケーションをビルドしましょう。
  Win および Mac OS で 64-bit版 4D Server のファイナルバージョンを使用しましょう (**OS X用64bit版4D Serverの使用** および **Windows用4D Server 64bit版の使用** 参照)。
- **新しいクイックレポートエディター** は旧版で作成したレポートにも対応しています (**ラベルエディター (64-bit)** 参照)。
- **新しいラベルエディター** は旧版で作成したラベルファイルにも対応しています (**クイックレポート(64-bit版)** 参照)。
- **GRAPH** コマンドにオブジェクト型パラメーターを指定して**グラフ**を作成しましょう。

SET PRINT OPTION

SET PRINT OPTION (option ; value1 {; value2})

引数	型		説明
option	倍長整数	➡	オプション番号
value1	倍長整数, テキスト	➡	オプションの値1
value2	倍長整数, テキスト	➡	オプションの値2

説明

SET PRINT OPTIONコマンドを使用し、 プログラムから印刷オプションの値を変更することができます。プリントパラメーターを変更する他のコマンド (**PRINT SETTINGS**、> 引数を使用しない **PRINT SELECTION**) が呼び出されない限り、このコマンドを使用して定義された各オプションは、セッションの間、データベース全体に対して適用されます。印刷ジョブが開いている間はこのオプションを変更することはできません。

*option*引数を使用し、変更するオプションを指定することができます。“**Print Options**”テーマ内の定義済定数のいずれか、またはPDFオプションコード (WindowsのみでPDFCreatorドライバーで利用可能) を渡すことができます。

指定した*option*の新しい値は、*value1*と (オプションの) *value2*に渡します。渡す値の数と種類は、指定したオプションのタイプによって異なります。

option番号を使用する (定数)

以下の表でoptionとそれに対応するvalueを説明します:

定数	型	値	コメント
Paper option	倍長整数	1	*value1* のみを使用した場合、ここには用紙の名前のみが含まれます。両方の引数を使用した場合、*value1* には用紙の幅が、*value2* には用紙の高さが含まれます。幅と高さはどちらもスクリーンピクセルで表現されます。プリンターで使用できる全ての用紙フォーマットの名前、高さと幅を取得する場合には**PRINT OPTION VALUES** コマンドを使用して下さい。
Orientation option	倍長整数	2	*value1* のみ:1=縦向き、2=横向き。異なるページ方向が使用されている場合、**GET PRINT OPTION**コマンドは*value1*に0を返します。 **64-bit 版のみ**: このオプションは印刷ジョブ内から呼び出す事が可能なので、同一印刷ジョブ中において縦向きを横向きに、あるいはその逆へと切り替えることが可能です。
Scale option	倍長整数	3	*value1* のみ: 拡大縮小の倍率の値(パーセント)。一部のプリンターでは倍率の変更を許可していないものもあるという点に注意して下さい。無効な値を渡した場合、プロパティは印刷時に100%へとリセットされます。
Number of copies option	倍長整数	4	*value1* のみ: 印刷する部数
Paper source option	倍長整数	5	(Windows のみ) *value1* のみ: コマンドで返されるトレイの配列の中で、使用される予定の用紙トレイのインデックスに対応する番号。このオプションはWindowsでのみ使用可能です。
Color option	倍長整数	8	(Windows のみ) *value1* のみ: カラーを管理するモードを指定するコード: 1=白黒(モノクロ)、2=カラー **64-bit 版**: このオプションは64-bit版の4Dではサポートされていません。(廃止予定)
Destination option	倍長整数	9	*value1*: 印刷先のタイプを指定するコード: 1=プリンター、2=(PC)/PS ファイル(Mac)、3=PDFファイル、5=スクリーン(OS X ドライバーオプション)。 *value1* が1あるいは5以外であった場合、*value2* には生成されたドキュメントへのパス名が含まれます。このパスは他のパスが指定されるまでは使用され続けます。保存先に同じ名前のファイルが既に存在していた場合には、それは置き換えられます。**GET PRINT OPTION**の場合、カレントの値が既定のリスト内にない場合、*value1* には-1が返され、OKシステム変数は1に設定されます。エラーが起きた場合、*value1* とOKシステム変数は0に設定されます。 **注**: Windowsにおいては、PDF Creatorドライバーがインストールされていた場合には印刷先を3(PDFファイル)に設定することができます。*(9;3;path)* の値が渡された場合、4Dは自動的に"サイレント"PDF印刷を開始します。この場合には、渡されたオプションコードであればどれでも受け取ります(*value2* に空の文字列を渡すかこの引数を省略した場合、印刷時にファイルを保存ダイアログが表示されるという点に注意して下さい)。印刷後、カレントの設定は保存されます。これは4DのPDF印刷の管理を簡略化し、ユーザーがマルチプラットフォームなコードを書けるようにします。*(9;3;path)*値が渡されなかった場合、印刷は4Dによって管理されず、渡されたPDF Creatorオプションコードはどれも無視されます。
Double sided option	倍長整数	11	(Windows のみ) *value1*: 0=片側印刷あるいは標準、1=両面印刷。*value1*=1のとき、*value2* にはページ綴じの設定が含まれます: 0=左綴じ(デフォルト値)、1=上綴じ **注:** このオプションはWindows環境においてのみ使用可能です。
Spooler document name option	倍長整数	12	*value1* のみ: スプーラドキュメントの一覧に表示される、カレントの印刷ドキュメント名。この宣言によって定義される名前は、新しい名前あるいは空の文字列が渡されない限りはセッションで印刷される全てのドキュメントに対して使用されます。標準のオペレーション(メソッドの場合にはメソッド名を、レコードの場合にはテーブル名を使用)を使用あるいは復元するためには、空の文字列を*value1* に渡して下さい。
Mac spool file format option	倍長整数	13	(Mac のみ) *value1* のみ: 0=PDFモードでジョブを印刷(デフォルト値)　1=PostScriptモードでジョブを印刷 **注:** - このオプションはWinodws環境下では何の効力も持ちません。 - OS Xでは、印刷はデフォルトでPDFで行われます。しかしながら、PDF印刷ドライバは格納されたPostScript情報をもつPICTフォーマットのピクチャーをサポートしません。これらのピクチャーは具体的にはヴェクター式の描画ソフトウェアによって生成されます。このような問題を避けるため、このオプションではOS X環境下のカレントのセッションで使用するために、印刷モードを変更することができます。ただしPostScriptモードでの印刷には予期せぬ副作用を引き起こす可能性がある点に注意して下さい。

			64-bit 版: このオプションはサポートされていません。代わりに、**SET CURRENT PRINTER** コマンドのGeneric PDF driver オプションで置き換えられています。
Hide printing progress option	倍長整数	14	*value1* のみ: 1=進捗ウィンドウを非表示、0=進捗ウィンドウを表示(デフォルト)。このオプションは、特にOS XでのPDF印刷の際に有用です。 **注:** データベース設定ダイアログボックス内には、既に印刷プログレスオプションがあります(インターフェースページ内)。しかしながら、この設定は全体に適用され、OS X環境下でのウィンドウを全て非表示にするわけではありません。
Page range option	倍長整数	15	4D Write Pro 専用のオプション
Legacy printing layer option	倍長整数	16	(Windows用4D 64-bit版のみ) *value1* のみ: 1=以降の印刷ジョブに対してはGDI-ベースの旧式の印刷レイヤーを選択。0=D2D印刷レイヤーを選択(デフォルト)。 **64-bit 版:** このセレクターはWindows用64-bit版4Dのシングルユーザーアプリケーションでのみサポートされます。他のプラットフォームでは無視されます。これは主に64-bit版4Dアプリケーションの4Dジョブ内で旧式プラグインが印刷できるようにするためにものです。

このコマンドを使用して設定を行うと、4Dアプリケーション全体に対しセッションの間中、そのプリントオプションが保持されます。**PRINT SELECTION**、**PRINT RECORD**、**Print form**、**QR REPORT** と **WP PRINT** コマンドおよびデザインモードを含めた4Dの印刷全般に対して、この設定が使用されます。

注:

- SET PRINT OPTIONコマンドを用いて設定したプリントオプションがリセットされないように、**PRINT SELECTION**、**PRINT RECORD**、**PAGE BREAK**コマンドでは、任意の引数 > を必ず使用してください。
- **SET PRINT OPTION** コマンドは主にPostScript プリンターをサポートします。このコマンドは他のタイプのプリンター、例えばPCLやlinkなどにも使用できますが、その場合一部のオプションが使用できない可能性があります。

PDFオプションを使用する

option 引数でPDFオプションコードを使用できるようにするためには、4D環境にPDFCreatorドライバーがインストールされていなければなりません (詳細情報については**WindowsにおけるPDFCreatorドライバーの統合**を参照してください)。さらにオプションコードが効力を得るためには、以下の文を使用してPDF印刷の制御を有効にしなければなりません:

```
SET PRINT OPTION(Destination option;3;fileName)
```

そうしなければoptionコードは無視されます。

PDFoptionコードは2つの部分からなるテキストタイプのコードです。*OptionType*と*OptionName*を"*OptionType:OptionName*"のように組み合わせます。このコードの説明は以下の通りです:

- *OptionType* はネイティブなPDFCreatorオプションあるいは4D PDF管理オプションのいずれを設定するかを指定します。2つの値が受け入れられます:
  - **PDFOptions** = ネイティブオプション
  - **PDFInfo** = 内部オプション
- *OptionName* は設定するオプションを指定します (*OptionType* 値に基づきます)。
  - *OptionType* = **PDFOptions**の場合、*OptionName*には複数のPDFCreatorネイティブオプションのうち一つを渡せます。例えばUseAutosaveオプションは自動バックアップに影響します。このオプションを変更するには、*option* 引数に"PDFOptions:UseAutosave"を渡し、使用する値を*value1*引数に渡します。PDFCreatorネイティブオプションに関する完全な説明は、PDFCreatorドライバーの説明書を参照してください。
  - *OptionType* = **PDFInfo**の場合、*OptionName*には以下の特定のセレクターを渡せます:
    - **Reset print**: 特に無限ループから抜けるために、内部的な待ち状態をリセットするために使用します。この場合*value1*は使用しません。
    - **Reset standard options**: すべてのPDFCreatorオプションをデフォルト値にリセットするために使用します。印刷中の場合、デフォルト設定はその印刷が終了後に適用されます。この場合*value1*は使用しません。
    - **Start**: PDFCreatorスプーラーを開始または停止するために使用します。*value1*に0を渡すと停止、1を渡すと開始です。
    - **Reset options**: SET PRINT OPTIONコマンドおよび**PDFOptions**を使用して、セッションの開始以降変更されたすべてのオプションをリセットします。
    - **Version**: PDFCreatorドライバーの現在のバージョンを読み取るために使用します。このセレクターは**GET PRINT OPTION**コマンドでのみ使用できます。番号は*value1*引数に返されます。
    - **Last error**: PDFCreatorドライバーから最後に返されたエラーを読みとるために使用します。このセレクターは**GET PRINT OPTION**コマンドでのみ使用できます。エラー番号は*value1*引数に返されます。

- **Print in progress**: PDFCreatorにより、4Dが印刷を待っているかどうかを知るために使用します。このセレクターは**GET PRINT OPTION**コマンドでのみ使用できます。*value1*引数に1が返されると、4DはPDFCreatorを待っています。そうでなければ0が返されます。
- **Job count**: 印刷キューにいくつのジョブがたまっているかを知るために使用します。このセレクターは**GET PRINT OPTION**コマンドでのみ使用できます。ジョブ数は*value1*引数に返されます。
- **Synchronous Mode**: 4Dが送信した印刷リクエストとPDFCreatorドライバー間の同期モードを設定するために使用します。4Dは印刷キュー内にある印刷ジョブの現在の状態に関する情報を取得できないので、このオプションを使用して、PDFCreatorドライバーが空き状態のときにのみジョブを送信することで、その実行をよりうまく制御できます。この場合、4Dはドライバーと同期されています。*value1* に0を渡すと4Dは即座に印刷リクエストを送信します (デフォルト値)。そして1を渡すと4Dは同期を行い、他のリクエストを送信する前にドライバーがジョブを終了するのを待ちます。

注: それぞれの印刷後に、4DはPDFCreatorを使用する他のアプリケーションとの衝突を避けるために、自動でPDFCreatorドライバーの設定を以前のものに戻します。

例題 1

以下のメソッドはテーブル中の全レコードを印刷するために、PDFドライバーを有効にします。PDFはC:¥Test¥Test_PDF_X (Xはレコード位置番号) に書き出されます:

```
SET CURRENT PRINTER(PDFCreator Printer Name)
 // WindowsではPDFCreatorがインストールする仮想プリンターを選択
If(OK=1) // PDFCreatorが実際にインストールされていれば

   ALL RECORDS([Table_1])
   For($i;1;Records in selection([Table_1]))
      SET PRINT OPTION(Destination option;3;"C:\\Test\\Test_PDF_"+String($i))
 // Destination optionに3を指定することでPDFCreator印刷ジョブを起動
      PRINT RECORD([Table_1];*)
      NEXT RECORD([Table_1])
   End for
 // PDFCreatorドライバーのオプションをリセット
   SET PRINT OPTION("PDFInfo:Reset standard options";0)
End if
```

例題 2

64-bit版では、Orientation option の値を同一印刷ジョブ内で変更することができます (特例)。**PAGE BREAK** コマンドの呼び出しより先に、このオプションがあらかじめ設定されている必要があることに留意ください:

```
ALL RECORDS([People])
PRINT SETTINGS
If(OK=1)
   OPEN PRINTING JOB
   SET PRINT OPTION(Orientation option;1) // 縦向き
   Print form([People];"Vertical_Form")

   SET PRINT OPTION(Orientation option;2) // 横向き
   PAGE BREAK // 必ずオプションの後にコールします
   Print form([People];"Horiz_Form")
   CLOSE PRINTING JOB
End if
```

システム変数およびセット

コマンドが正しく実行されるとシステム変数OKに1が設定され、そうでなければ0が設定されます。

エラー管理

*option*に渡した値が無効であるか、そのプリンタで*option*が利用できない場合、コマンドはエラーを返し (**ON ERR CALL**コマンドでインストールされたエラー管理メソッドを用いて、このエラーをとらえることができます) 、オプションの現在の値がそのまま保持されます。

OS X用64bit版4D Serverの使用

4D v15.1以降、OS X用に64 ビット版の4D Server が提供されています。この新製品により、お使いの4D Server アプリケーションで64ビットのAppleマシンの実力を全て引き出す事ができるようになります。64-ビット版のテクノロジーの主な利点は、より多くのRAMメモリーを割り当てることができる、という点です。

このセクションでは、64ビット版の4D ServerをMac OS Xで導入・使用する際の注意点を扱っていきます。

OS Xの必須バージョン

OS X用の64ビット版4D Serverを使用するためには、**10.9** (Mavericks)以上のOSが必要になります。お使いの4D Serverのバージョンがどのでご利用いただけるかは、4D Webサイトにある4D-OS対応早見表を参照して下さい。

アーキテクチャー

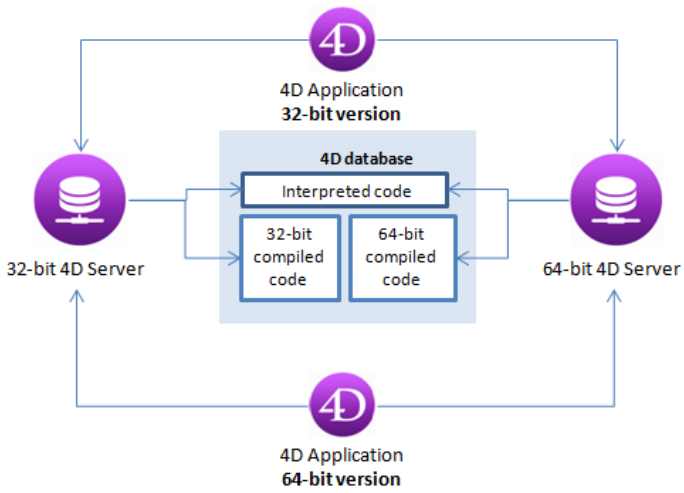
64ビットアーキテクチャー用の4D Server はその環境専用のバージョンです(32ビットOS上では動作しません)。

クライアント側からは、4Dアプリケーションは(OS X用・Win用どちらでも)64ビット版の4D Serverにアクセスする事ができます。この場合接続するのに使用する4Dは標準の32ビット版です(以下のダイアグラムを参照して下さい)。4Dクライアントアプリケーションとは、リモートモードの4Dと、4D Volume Desktopを使用して組み込まれたアプリケーションが含まれます。

インタープリタモードでは、同じ4Dデータベースを64ビット版の4D Serverでも32ビット版4D Serverでも実行する事ができます。開発は、使用するアプリケーションを問わず、同じように進めることが可能です(ただし以下の制約が付随します)。

コ ンパイルモードでは、64ビット版4D Serverで実行するためには64ビットプロセッサー用にコンパイルされている必要があります。32ビット用にコンパイルされていて、インタープリタ コードを含まないデータベースは64ビット版4D Serverで実行することはできません。

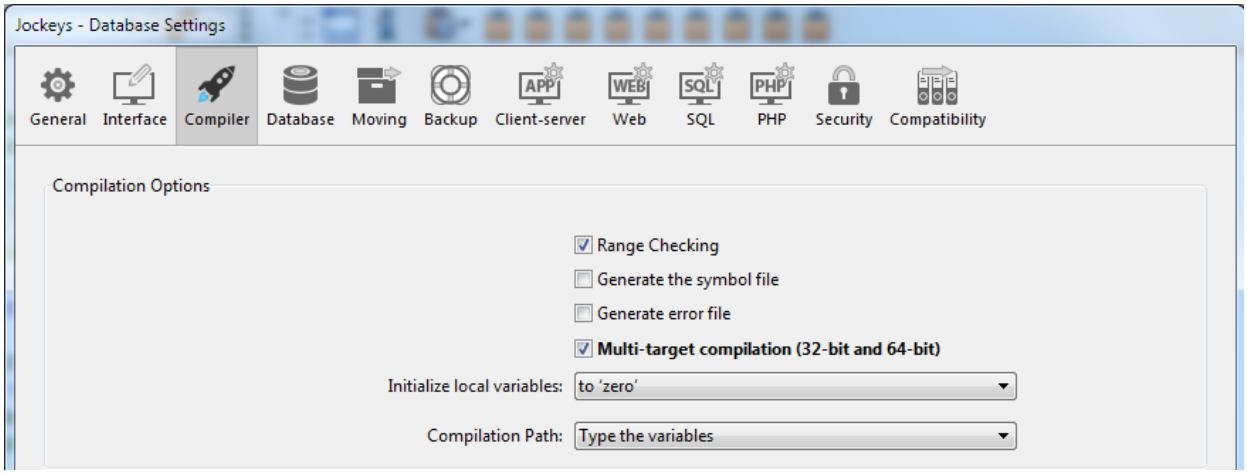
***4D Server 32-bit*版 および *64-bit*版アーキテクチャーの概要**



データベースがプラグインを使用する場合、OS X用64-bit版プラグインがサーバーマシン側においてインストールされていなければなりません。

64-bit 用コンパイル

4Dアプリケーションは、32ビット用と64ビット用とにコンパイルすることができます。これをするためには、データベース設定の「コンパイラー」ページにある、**マルチターゲットコンパイル(32-bit用と64-bit用)**オプションをチェックする必要があります:



このオプションがチェックされているとき、コンパイラーは64ビット用のコードと32ビット用のコードを.4DC と .4DBファイルに含めます。それにより、これらのファイルは32ビット版・64ビット版、両方の4D Serverで実行することが出来るようになります。デフォルトでは、このオプションはチェックされていません。

コンパイルされたコードの互換性

OS X 64-bit アーキテクチャーをサポートするために、4D内蔵のコンパイラーが変更されました。その結果、4D v15(またはそれ以降のバージョン)でコンパイルされたデータベースのみが OS X 64-bitで実行可能となります(**注**: コンパイラーは*4D v14 R3*以降変更されています)。つまり:

- 既存の4DデータベースをOS X 64-bitにおいてコンパイルモードで使用したい場合、そのデータベースを4D v15(またそれ以降のバージョン)で再コンパイルする必要があります。
- データベースがコンパイルされたのコンポーネントを使用する場合、そのコンポーネントを4D v15(またそれ以降のバージョン)で再コンパイルする必要があります。

プロセススタックのサイズ

64-bitバージョンの4D Server上で走るプロセスのスタックは、32-bitバージョンよりも多くのメモリーを必要とします (約2倍)。**Execute on server**や**New process**コマンドを使用して64-bitバージョンの4D Server上でプロセスを作成する場合、デフォルト値(0)、または少なくとも512KBを*stack*引数に渡すことを、呼び出し連鎖が大きくなる場合やスタックが足りないというエラーが発生する場合にはさらにそれを増やすよう推奨します。コードが64-bit 4D Server上で実行されるためのものである場合、この引数をチェックするようにしてください。

サポート対象外の機能

以下の機能・技術はカレントのOS X用の64bit版4D Serverではサポートされません:

機能/技術	コメント
XSLT と Xalan	**_o_XSLT APPLY TRANSFORMATION**、 **_o_XSLT SET PARAMETER**、 そして **_o_XSLT GET ERROR** は動作しません。代わりに PHP *libxslt* モジュールを使用して下さい。
PICT フォーマット	画像の代わりに、'サポート外の画像フォーマット'画像 + ファイル拡張子が表示されます(**利用不可能なピクチャーフォーマット**を参照して下さい)。PICTフォーマットは4D全体で使用廃止になっています。**_o_AP Is Picture Deprecated**も参照して下さい。
cicn アイコン	**GET ICON RESOURCE**コマンドはサーバーではサポートされていません(*)。
データベース .RSR ファイル	データベース .RSR ファイルは、自動的には開かれません。**Open resource file** を使用する必要があります。
書き込み可能なリソースファイル	**_o_Create resource file** はサーバー上ではサポートされていません(*)。リソースファイルは読み込み専用でのみ開く事ができます。 **注意:** *Mac OS* リソースファイルは、既に*4D v11*から廃止予定となっていました。
_o_Font number	このコマンドはサーバー上ではサポートされていません。(*)
ASCII 互換モード	Unicodeモードのみがサポートされています
旧式ネットワークレイヤー	*ServerNet* のみがサポートされています(**新しい ServerNet ネットワークレイヤー(互換性)**を参照して下さい)
読み込み/書き出しダイアログボックス	利用不可
ラベルエディター	利用不可
Web エリア内で統合Web Kit を使用する	利用不可

(*) サーバー側で実行した場合にはエラーが返されます

4D Write から 4D Write Pro へドキュメントを変換

4D Write ドキュメントの変換

4D Write Pro は、ほとんどのプロパティ情報をサポートしたまま、既存の 4D Writeドキュメントを開いて変換することができます。

上図では、左に 4D Write エリア、右に 4D Write Pro エリアを表示しています (新しいライブラリオブジェクトを使って作成 - 後述参照)。4D Write エリアの中身は **WP New** コマンドを使って簡単に変換することができます:

```
// 4D Write エリアのコンテンツを 4D Write Pro に復元します。
[WRITEAREAS]AreaNTWP:=WP New([WRITEAREAS]AreaNT_)
```

しかし、4D Write は 32-bit版の 4D v16 でのみ使用できることに注意が必要です。まず 4D Write ドキュメントを変換してから、64-bit に変換する必要があります。

4D Write と違い、4D Write Pro はプラグインではなく、4D に完全に統合されています。4D Write Pro には 4D Write と同じライセンスが使用されます。4D Write Pro を有効にするには、アプリケーションにこのライセンスをインストールする必要があります。

4D Write Proオブジェクトに 4D Writeドキュメントを読み込む方法は二つあります:

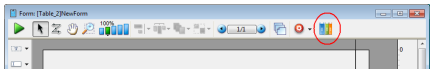
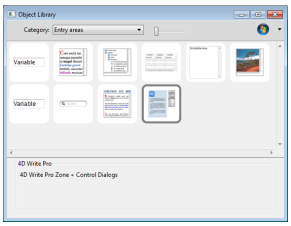
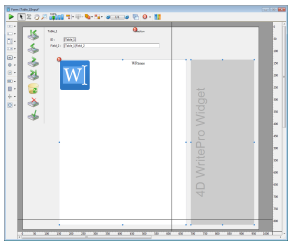
- ディスク上に保存されている 4D Write ファイルに関しては、**WP Import document** コマンドを使用します。
- BLOBフィールドに格納されている 4D Write エリアに関しては、**WP New** コマンドを使用します。

互換性に関する注意:

- サポートされるのは、4DWite ドキュメントのうち最後の世代 ("4D Write v7") のドキュメントに限られます。
- 4D Write Proエリアにインポートできるオブジェクトや機能についての詳細は **4D Write のプロパティで復元されるもの** を参照ください。
- 4D Write ドキュメントから 4D Write Pro エリアへのコピー・ペーストは現時点ではサポートされていません。4D Writeドキュメントの読み込みは 4D Write Pro ランゲージコマンドの使用によってのみ可能です。
- 4D Write Pro の機能は Windows において Direct2D に依存しています。必要となる Direct2D のバージョンを利用するため、Windows 7 または Windows Server 2008 のマシンでは *"Windows*プラットフォーム更新プログラム*"* のコンポーネントがインストールされていることを確認してください。

オブジェクトライブラリの 4D Write Pro オブジェクト

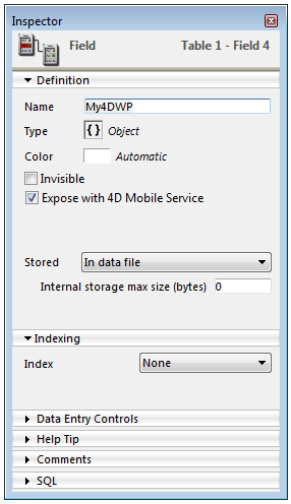
4D v16 では、フォームエディターにて提供されている設定済オブジェクトのライブラリに **4D Write Pro** フォームオブジェクトが新しく加わりました。このオブジェクトをフォーム上にドラッグ & ドロップすると、あらかじめ設定のされた 4D Write Pro エリアが挿入されます。このエリアには、コンテンツ操作用のコントロールパネルを持つ 4D Write Pro ウィジェットサブフォームも関連づいています:

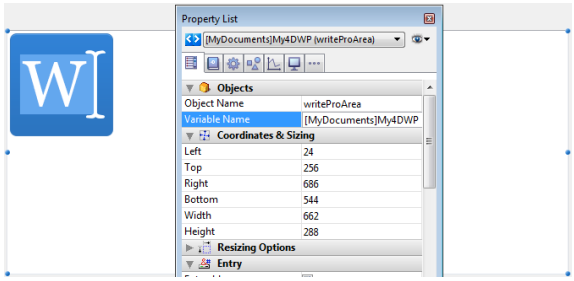




詳細については **4D Write Pro エリア** を参照ください。

4D Write Pro: オブジェクトフィールドの割り当て

データベースのストラクチャー内において、4D Write Proドキュメントを保存するのに4Dオブジェクトフィールドを使用することができます。
4D Write Proドキュメントを保存するためのオブジェクトフィールドを作成したら、あとは当該エリアを含んでいるフォーム内で参照するだけです。フォームエディターにて、4D Write Pro エリアのプロパティリストを開き、**変数名**の欄に標準の "[Table]Field" 表記でフィールド名を入力します:





これで4D Write Proエリアはオブジェクト型フィールドと関連付けされました。

4D式のフィルタリング

以前のバージョンにおける 4D Write Pro ドキュメント内では、フィルタリングが有効化されていませんでした。4Dメソッドを使用している 4D Write Pro ドキュメントを 4D v16以降へと変換した場合、これらは正しく評価されなくなり、代わりに "## Error # 48" メッセージが表示されます。
この場合、**SET ALLOWED METHODS** コマンドを使って、許可されたメソッドのリストに使用したいメソッドを追加する必要があります。

変更されたコマンド

4D Write Pro 用の新コマンドが追加されたほか、複数の既存コマンドが 4D Write Pro に対応するように進化をとげました:

- **OBJECT SET HORIZONTAL ALIGNMENT**: 4D Write Pro オブジェクトに対応しています。*alignment* パラメーターに、4D Write Pro エリアに限定された、新しい wk justify 定数が提供されており、両端揃えの設定が可能です。
- **OB SET**: このコマンドは4D Write Proオブジェクトに定義されている属性に対応しています (**WP SET ATTRIBUTES** と同様)。次のシンタックスがサポートされています:
  OB SET (objSel | wpDoc; attribName ; attribValue {; attribName2 ; valeurAttrib2 ; ... ; attribNameN ; attribValueN})
  制限: 属性値としてピクチャーフィールドおよび変数を直接受け渡すことはできません。
- **OB Get**:
  このコマンドは4D Write Proオブジェクトに定義されている属性に対応しています (**WP GET ATTRIBUTES** と同様)。次のシンタックスがサポートされています:
  OB Get (objSel | wpDoc; attribName) -> Function result
  このコマンドは **OB SET** と同様に制限されています: 属性値としてピクチャーフィールドおよび変数を直接使用することはできません。
- 4D Write Pro 属性の文字列変換: **JSON Stringify** コマンドを使って 4D Write Pro オブジェクトを JSON に変換すると、"題名" 属性のみが文字列として取得されます。
  カスタム属性を追加している場合には、これらも文字列として書き出されます (**4D Write Proドキュメントを4Dオブジェクトフィールドに保存する** の **カスタムの属性を使用** 参照)。
- **QUERY BY ATTRIBUTE**: **4D Write Proドキュメントを4Dオブジェクトフィールドに保存する** の章での記述のとおり、ドキュメントがオブジェクトフィールドに保存されている場合、**QUERY BY ATTRIBUTE** コマンドは 4D Write Proの内部属性およびカスタム属性をサポートします。

.4wp ドキュメントフォーマット

4D v16以降では、ネイティブな **.4wp** フォーマットを使用することによって 4D Write Proドキュメントをロスなくディスク上に保存したりディスクから開いたりすることができます。

.4wp フォーマットはドキュメント名と同じ名前を持つ ZIPフォルダーと、その中に格納された HTMLテキストと画像から構成されます:

- HTMLテキストは、通常の HTML と (計算されていない) 4D式に加え、4D特有のタグを組み合わせます。
- 画像は、HTMLファイルの隣にある、ドキュメント名と同じ名前を持つフォルダー内に保存されています。

.4wp ドキュメントは HTML に基づいているため、HTMLをサポートしているものであればどんな外部アプリケーションでも

それを読み込んだり開いたりすることができます。

注: 4D Write Pro の内部ドキュメントは HTML に 4D独自の拡張を加えたもので、HTML5/XHTML5 に準拠していながら、このマニュアルで説明されている独自の HTML/CSS属性のサブセットとタグもサポートしています。したがってデータ損失のリスクなく 4D Write Pro で開けるのは、4D Write Pro で書き出された HTMLドキュメントに限られます。

WP New

WP New {(source)} -> 戻り値

引数	型		説明
source	文字, BLOB, Object	➡	4D HTMLソースまたは4D Write Blob
戻り値	Object	⮌	4D Write Pro オブジェクト

説明

WP New コマンドは4D Write Pro オブジェクトを作成し、返します。

source 引数を省略した場合、コマンドはデフォルトで空の4D Write Proオブジェクトを返します。

また *source* 引数を使用した場合、新しい4D Write Pro オブジェクトは*source* 引数の中身をコンテンツとして返されます。渡せる内容は以下の通りです:

- 文字列 の引数: この場合、4D HTMLソースを渡します。つまり、wk web page html 4D オプションを使用した**WP EXPORT VARIABLE** で書き出されたテキストです。このテキストは参照(4Dタグと式)と埋め込まれた画像を含むことができます。
- *blob* 引数: この場合、いかのどちらかを渡す事ができます:
  - BLOBに保存された4D Write Pro(.4wp)フォーマットドキュメント。4D Write Proドキュメントフォーマットの詳細については、**.4wp ドキュメントフォーマット** を参照して下さい。
  - BLOBに読み込まれている以前の4D Writeエリア(.4w7 または .4wt を含んだBlobがサポートされます)。4D Write Proオブジェクト内で現在サポートされている4D Writeの機能の詳細な一覧については、**4D Write ドキュメントの読み込み** の章を参照して下さい。

  ディスク上に保存されている4D Write ドキュメント(.4w7 or .4wt) を読み込みたい場合、**WP Import document** コマンドの使用も検討してみて下さい。
- *object* 引数: この場合には、4D Write Pro レンジオブジェクトを渡します。**WP New** は指定したレンジから新規ドキュメントを作成して、これを返します。指定レンジがドキュメントの全レンジでない場合、最初のセクションだけがエクスポートされ、ブックマークなどが存在する場合には、これらは無視されます。

返されるオブジェクトは、例えば **MissingRef** コマンドに受け渡し可能な、完全なドキュメントです。

例題 1

空の4D Write Proオブジェクトを作成したい場合を考えます:

```
myWPObject:=WP New
```

例題 2

簡単な4D式の参照を含んだ4D Write Proオブジェクトを作成したい場合を考えます:

```
C_TEXT(myText)
myText:="Today is "
ST INSERT EXPRESSION(myText;"string(current date;System date long)";ST End text)
myWPA:=WP New(myText)
```

例題 3

以前作成したテンプレートを使用して4D Write Proエリアを初期化したい場合を考えます:

```
 //既存のエリアからテンプレートを書き出し
C_TEXT(wpTemplate)
WP EXPORT VARIABLE(myWPArea;wpTemplate;wk web page html 4D)

 // 新規エリアに対してテンプレートを使用
C_OBJECT(myNewWPA)
myNewWPA:=WP New(wpTemplate)
```

例題 4

4Dフィールドに保存されている4D Write ドキュメントを新しい4D Write Proエリア内に読み込みたい場合を考えます:

```
C_OBJECT(wpArea)
wpArea=WP New([Templates]Reference_)
```

例題 5

あらかじめフォーマットされ、それぞれがブックマークとして保存された複数のパーツで構成された、テンプレートドキュメントを定義しました。このテンプレートから任意のブックマークを新規ドキュメントとして抽出し、作成中のドキュメントに挿入することができます。

```
ARRAY TEXT($_BookmarkNames;0)
WP GET BOOKMARKS([TEMPLATES]WP;$_BookmarkNames) // テンプレートからブックマークを取得します
$targetRange:=WP New //空のドキュメントを作成（これが最終的なドキュメントになります）

$p:=Find in array($_BookmarkNames;"Main_Header") // 使用したいブックマークを名称で探します
If($p>0)
   $Range:=WP Get bookmark range(WParea;$_BookmarkNames{$p}) // ブックマークからレンジを取得します
   $RangeDoc:=WP New($Range) // レンジから新規ドキュメントを作成します
   WP INSERT DOCUMENT($targetRange;$RangeDoc;wk append+wk freeze expressions) // wk append =
$RangeDocドキュメントは $targetRange の先頭に挿入されます
End if
```

SET ALLOWED METHODS

SET ALLOWED METHODS (methodsArray)

引数	型		説明
methodsArray	文字配列	➡	メソッド名配列

説明

SET ALLOWED METHODS コマンドを使用して、カレントセッションのフォーミュラーエディター上に表示されるメソッドを指定することができます。指定したメソッドはコマンドリストの最後に表示され、フォーミュラーで利用することができます。デフォルトでは (このコマンドを使用しない場合)、フォーミュラーエディター上にメソッドは表示されません。許可されていないメソッド名がフォーミュラーで使用されている場合、シンタックスエラーが生成され、フォーミュラーの確定はできません。

引数*methodsArray*には、フォーミュラーエディターに提示するメソッドリストを格納した配列の名前を渡します。この配列は事前に定義しておかなければなりません。
メソッド名にワイルドカード記号（@）を使用し、許可されるメソッドグループを定義することができます。

デフォルトとして許可されていない4Dコマンドやプラグインコマンドをユーザーが呼び出せるようにしたい場合、これらのコマンドを取り扱う特定のメソッドを使用しなくてはなりません。

注: データベース設定の"セキュリティ"ページのオプションによって、すべてのユーザーまたはDesignerとAdministratorに対し、フォーミュラーエディターでのコマンドやメソッドへのアクセスを制限するメカニズムを無効にすることができるようになりました。"すべてのユーザーを制限する"オプションが チェックされていると、 **SET ALLOWED METHODS** コマンドは効果がありません。

例題

この例は、名前が“formula”で始まるすべてのメソッドと、“Total_general”メソッドをフォーミュラエディタで使用可能にします:

```
ARRAY STRING(15;methodsArray;2)
methodsArray{1}:="formula@"
methodsArray{2}:="Total_general"
SET ALLOWED METHODS(methodsArray)
```