4D v13

アップグレード
Windows®/Mac OS®

4D v13 - Upgrade
Windows® and Mac OS® Versions

IMPORTANT LICENSE INFORMATION

Use of this software is subject to its license agreement included with the software. Please read the License Agreement carefully before using the software.

目次

1 ようこそ

4D v13 にようこそ。この新しいバージョンでは既存のデータベースとの互換を最大限保ちつつ、オープンであることにフォーカスしています :

- 完全に書き直された HTTP サーバー、および 64-bit ODBC ドライバーを使用すれば他のデータベースやインターネットからデータベースにアクセスする事ができます。
- データベース内のすべてのメソッドに読み込み / 書き込みアクセスを可能とする新しいソースツールキットを使用すれば、他の開発ツールを使用してバージョン管理などを行うことができます。ソースツールキットを使用して、例えばコードを標準のテキスト形式で書き出し、管理することができます。
- Web エリアでの描画エンジンの選択、リストボックスの新しいオプション、ピクチャーメタデータのインデックス化、Windows では Direct2D 描画エンジンや XPS を使用したプリントプレビューを使用するなど、新世代のインターフェースを構築できます。
- 最後に、4D v13 は 4D アプリケーションの最適化や開発を容易にする数多くの機能を提供します。BLOB やピクチャーデータを外部ファイルとして保存したり、プログラムからフォームオブジェクトにアクセスする為に使用できる一連のコマンドが追加されました。

前バージョンのデータベースの変換

バージョン 6.x、2003.x、2004.x、11.x そして 12.x で作成されたデータベースは (ストラクチャー、データファイルとも) 4D v13 と互換性があります。

- バージョン 6.x、2003.x、2004.x のデータベースファイルはウィザードを使用して変換しなくてはならず、変換後は元のバージョンで開くことができません。

- バージョン 11 または 12 のデータベースは直接バージョン 13 に変換できます。

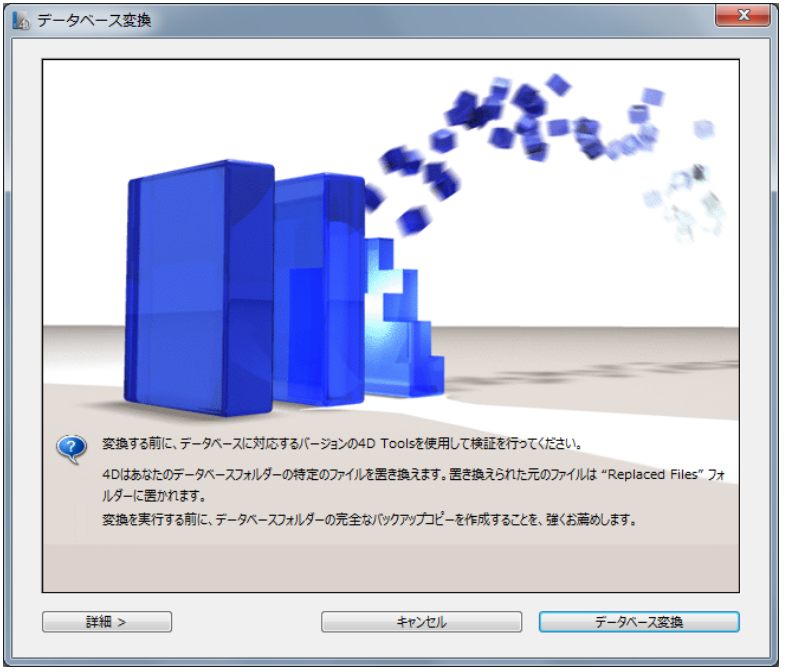
注 インタープリターストラクチャーを変換できます。コンパイルコードが含まれていてもかまいませんが、変換後に再コンパイルが必要です。

バージョン 6.x, 2003.x, 2004.x のデータベース

4D のデータベースエンジンレベルで構造的な変更が行われたため、バージョン 11 より前のデータベースストラクチャーおよびデータファイルはウィザードを使用した完全な変換が必要です。このウィザードはまず元のデータベースのコピーを作成します。コピーを使用すれば前のバージョンに戻ることもできます。

変換前に 4D Tools ユーティリティを使用してデータベースの整合性を検証 (圧縮、検証、および必要に応じて修復) することをお勧めします。元のバージョンに対応した 4D Tools を使用してください。

データベースの変換を行うには、4D v13 のデータベースを開くダイアログで対象のストラクチャーファイルを選択します。データベース変換ウィザードが開かれます :

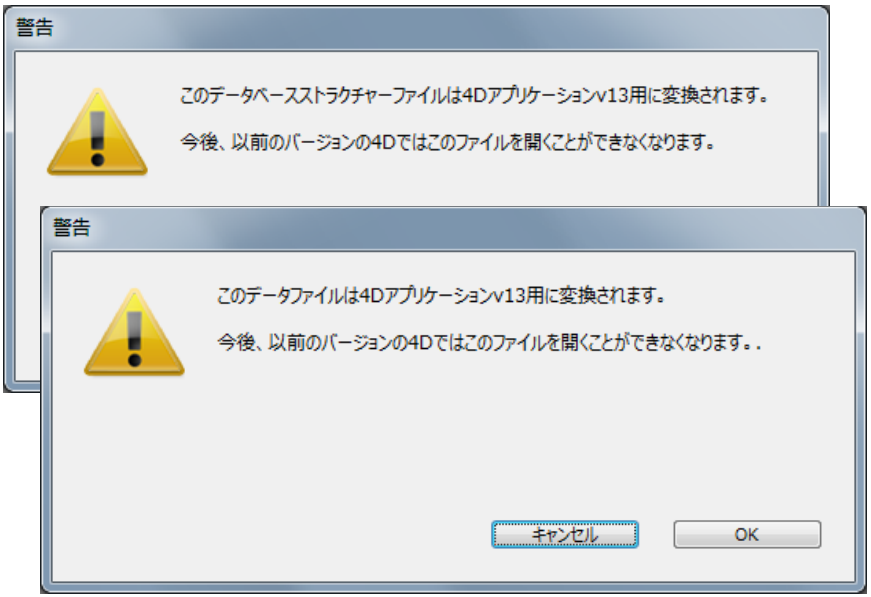


データベース変換ボタンをクリックすると、ストラクチャーとデータファイルを変換する標準の処理が開始されます。

- このダイアログおよび変換処理の詳細については、4D デザインリファレンスマニュアルの " 以前のバージョンからのデータベースの変換 " を参照してください。
- 変換時にはサポートが停止された古いメカニズムが削除されます。削除の動作に関する詳細は 4D v11 SQL アップグレードマニュアルを参照してください。また変更点については 4D v11 SQL 変換の手引きを参照してください。いずれのドキュメントも以下の URL からダウンロードできます : http://doc.4d.com/home.ja.html

バージョン 11, 12 のデータベース

バージョン 11 および 12 のデータベースは、4D v13 で開く際に直接変換されます。ファイルが変換され、以前のバージョンでは開けなくなることを通知するダイアログが 2 つ連続して表示されます :



データファイルを変換すると、インデックスが再構築されます。

注 変換対象の v11 データベースで、v11 よりも古い 4D Pack を使用していた場合、変換前に元のデータベースに最新の 4D Pack v11 プラグインをインストールすることを推奨します。

バージョン 11, 12 のコンポーネント

4D v13 は v12 や v11 のコンポーネントを (コンパイル済みでもインタープリターでも) 直接開くことができます。変換の必要はありませんし、確認ダイアログが表示されることもありません。コンポーネントは常に読み込みのみで開かれる点に留意してください。

コンポーネントを再コンパイルする必要はありませんが、v13 への変換は .4DB ファイルだけが可能であり、.4DC ファイルは変換できません。

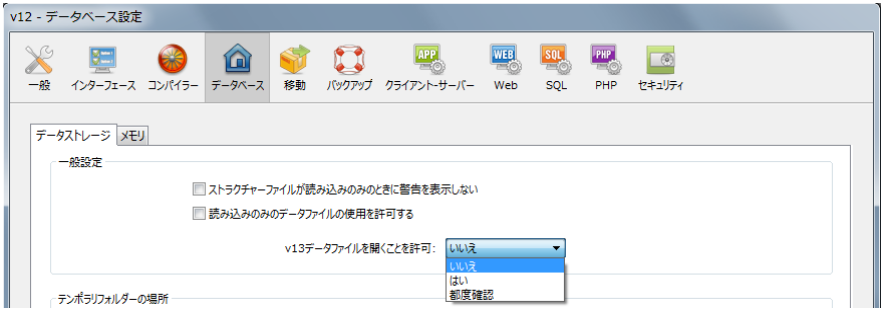
変換したデータベースの前方互換性

v13 に変換された 4D データベースファイルの前方互換性について以下の表にまとめます :

4D バージョン	v13 ストラクチャー	v13 データファイル
4D v11 以前	不可	不可
4D v12	不可	可能 (*)

(*) v12 の明示的に許可していれば、一度 v13 に変換したデータファイルを v12 で開くことが可能です。この設定はデータベース設定の " データベース / データストレージ " ページ内、**v13 データファイルを開くことを許可**設定を使用して行います :

v13 データファイルを開けるようにするための 4D v12 データベース設定



デフォルトでは**いいえ**オプションが選択されています。**はい**を選択すればデータファイルが直接開かれ、**都度確認**を選択するとデータを開く前に確認ダイアログが表示されます。

このオプションは特定の状況下でデータの復旧を行う目的で用意されており、注意を持って使用する必要があります :

- データを再度開く前 : すべてのケースで、v13 インデックスファイル (**.4dindx** ファイル) を削除することを推奨します。
- データを再度開いた後 : バージョン 13 の新機能がデータベーステーブルに適用されている場合、MSC を使用してデータファイルの圧縮および修復を行うことを推奨します。

互換性

ここでは以前のバージョンから変換されたアプリケーションの動作を変更してしまうかもしれない新しい機能について説明します。

4D Chart は外部プラグインに

4D Chart プラグインは 4D v13 アプリケーションから取り除かれ、追加のプラグインとして提供されるようになりました。以前のバージョンの 4D ではこのプラグインがアプリケーションに含まれていました。

あなたの 4D アプリケーションで 4D Chart の機能を使用している場合、4D Chart v13 外部プラグインを所定の場所にインストールすることで動作するようになります。インストールは他のプラグインと同様であり、4D レベルあるいはデータベースレベルの **Plugins** フォルダーに 4D Chart プラグインを配置するだけです。

4D Chart プラグインは以降機能向上および修正が行われない点に留意してください。データをグラフィカルに表現するためには SVG を使用することをお勧めします。

廃止予定の機能

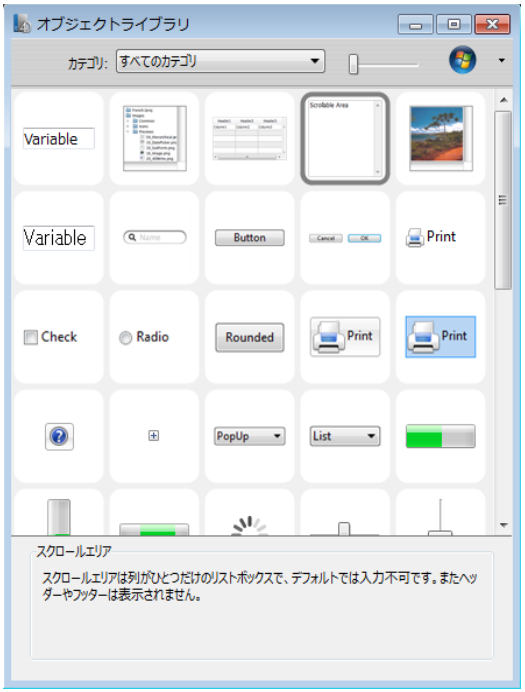
AP Get file MD5 digest (4D Pack)

4D Pack の **AP Get file MD5 digest** 関数は互換性のために保持されています。このコマンドは廃止予定であり、使用を停止する必要があります。この機能は新しい Generate digest コマンドで置き換えることができます。

スクロールエリア

4D v13 のフォームエディターでは " スクロールエリア " を作成することはできなくなりました。新しいインターフェースにおいてはリストボックスオブジェクトを使用してください。

4D v13 ではリストボックスに基づくスクロールエリアを作成するために、定義済みオブジェクトライブラリから " スクロールエリア " を選択できます :



注 データベースが変換されると、既存のスクロールエリアはリストボックスに変換されます。以前のバージョンのスクロールエリアと同じように動作させるため、プロパティリスト中の**互換性**オプションが選択されます (40 ページの " スクロールエリアの互換性 " 参照)。しかしこの互換モードのリストボックスを標準のリストボックスに変更することを推奨します。

Mac OS 記号の表示

4D v13 からはメソッドエディター中で Mac OS 用の記号を表示できなくなります :

- 時間定数に使用されていた † (Option+T)
- 文字位置指定のために使用されていた ≤...≥ (Option+< と Option+>)

注 これらの記号はもともと日本語バージョンで使用されていませんでした。

Mac OS においてキーコンビネーションを使用することでこれらの記号を挿入することが可能ですが、それらはそれぞれ ? と [[および]] に置換されて保存されます。

削除された機能

4D v12 のリリース時にアナウンスされた通り、いくつかの機能が 4D v13 では削除されました。

Web サーバーコンテキストモード

4D v13 の新しい Web サーバーではコンテキストモードがサポートされません。コンテキストモードに関連するすべてのオプション、定数、コマンド (例えば **Web Context**, **SET WEB DISPLAY LIMITS** や **SET WEB TIMEOUT**) は無効となりました。これらのコマンドは名前の前に "_o_" が付加され、実行するとエラー 33 (メソッドまたは関数が実装されていません) が生成されます。
替わりの機能として、新しい Web セッション管理を使用することを推奨します (61 ページの "Web セッション管理 " 参照)。

外部 CGI のサポート

4D v13 HTTP サーバーは (URL で呼ばれる) 自動モードであれ (**SET CGI EXECUTABLE** を使用する) マニュアルモードであれ、外部 CGI の実行をサポートしません。さらに **SET CGI EXECUTABLE** コマンドにも "_o_" が付加され、実行すると エラー 33 (メソッドまたは関数が実装されていません) が生成されます。

一時的なブレークポイント

一時的なブレークポイントは 4D v13 で利用できなくなります。この処理は新しい**デバッガーのコンテキストメニュー**で置き換えることができます。

モノクロパターン

もともとモノクロインターフェースで使用されていた旧来のパターンパレットはフォームエディターから取り除かれました。変換されたデータベースでは既存のパターンが自動でカラーのコンビネーションに置き換えられます。この点に関する詳細は 43 ページの " 以前のパターンの変換 " を参照してください。

最低動作環境

4D v13 製品ラインアプリケーションは動作のために最低以下の環境を必要とします :

	Windows	Mac OS
プロセッサー	Intel® Core Duo	
OS	Windows XP SP3(*), Windows 7	Mac OS X 10.6.8 以降
RAM メモリ	2 GB	
画面解像度	1280 * 1024 ピクセル	

(*) Windows XP では、4D v13 に統合された WebKit 描画エンジンを使用できません。詳細は 44 ページの "Web 描画エンジンの選択 " を参照してください。

2 データベース

4D v13 のデータベースエンジンでは新機能が提供され、最適化が行われています :

- ピクチャーフィールドのキーワードインデックスサポート
- 読み込んだピクチャーファイル名の記憶
- BLOB、テキスト、およびピクチャーフィールドのデータをデータファイルの外に保存可能

ピクチャーフィールド

ここでは 4D v13 におけるピクチャーフィールドの管理に関連する新機能について説明します。

注 ピクチャーフィールドのデータをデータファイルの外に保存することも可能となっています。この点については 21 ページの " データをデータファイル外に保存 " で説明しています。

ピクチャーキーワードのインデックス

4D v12 から SET PICTURE METADATA と GET PICTURE METADATA コマンドを使用して、ピクチャーに関連付けられているメタデータを保存し管理することが可能となりました。

4D v13 ではこのデータをインデックス化することで、アクセスを高速にできます。この機能により、保存したキーワードを使用したピクチャーの検索をほとんど瞬時に行えるようになります。

インデックス可能なメタデータ

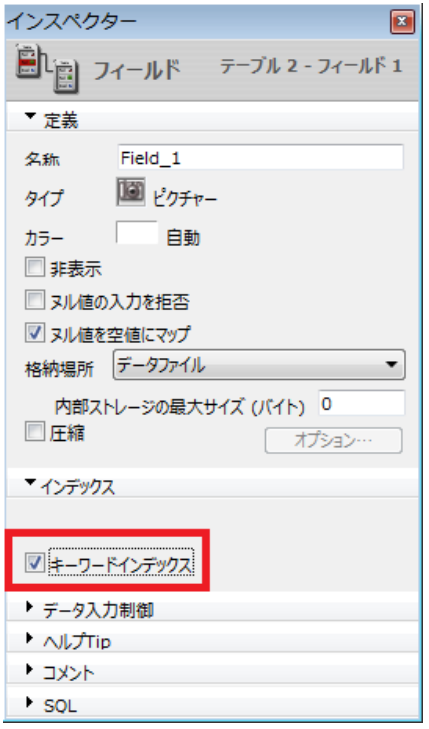
ピクチャーキーワードのインデックスは **IPTC/Keywords** タイプのメタデータに基づきます。これらのタイプのメタデータは特に TIFF や JPEG フォーマットでサポートされています (BMP、PNG、および GIF はこのタイプをサポートしない点に留意してください)。

他のタイプのメタデータのインデックスはサポートされていません。

ピクチャーキーワードインデックスの管理

ピクチャーフィールドに関連付けられたキーワードインデックスの作成や削除の方法は、標準キーワードのそれと同じです。この処理はストラクチャーエディターやプログラムから行うことができます。

- **ストラクチャーエディターのインスペクターを使用する**: ピクチャーフィールドのインスペクター上で、対応するオプションを選択、あるいは選択解除することで、インデックスの作成と削除を行えます .



注 エディターのコンテキストメニューから**インデックス > キーワード**と選択してインデックスを作成することもできます。

- **ストラクチャーエディターのインデックスエクスプローラーを使用する**: (ボタンをクリックするかコンテキストメニューから**新規インデックス ...** コマンドを選択して表示される) インデックスエクスプローラーを使用してピクチャーフィールドを表示し、既存のインデックスに追加することができます。ピクチャーフィールドではキーワードタイプのインデックスのみを利用できます。
- **プログラムを使用する**: 既存の CREATE INDEX や DELETE INDEX コマンドはピクチャーフィールドのキーワードインデックスをサポートします。

注 SET INDEX コマンドを (index 引数 -1 で) 使用してピクチャーフィールドのキーワードインデックスを作成できますが、この場合プログラムを使用した削除はできません。

ピクチャーのキーワードインデックスは、ピクチャーが保存 (レコード作成、更新、読み込みなど) されるたびに 4D が自動で更新します。4D は **IPTC/Keywords** タイプのメタデータをピクチャー内に見つけると、自動でインデックス化します。ピクチャーフィールドのインデックスにキーワードを含める目的で開発者が SET PICTURE METADATA コマンドを呼び出す必要はありません。

ピクチャーキーワードインデックスの利用

ピクチャーキーワードインデックスを利用すれば、アプリケーションの速度を劇的に向上させられます。

テキストフィールドのキーワードインデックスのように、このインデックスは **% 演算子**で使用します : インデックスの値を明示的に使用するには、この演算子をクエリまたは並び替えフォーミュラで使用しなければなりません。例えば :

```
QUERY ([PICTURES];[PICTURES]Photos % "cats")
  // cats キーワードを持つ写真を探す
```

この書式はすべてのクエリおよび並び替えコマンドで同様に動作します : QUERY BY FORMULA, QUERY SELECTION, ORDER BY 等
% 演算子の動作については 4D ランゲージリファレンスマニュアル " 比較演算子 " を参照してください。

キーワードのリスト

フィールドに定義されたすべてのピクチャーのキーワードリストやあるピクチャーに割り当てられたキーワードリストを取得できます。

- テキストフィールドのように DISTINCT VALUES コマンドを使用して、ピクチャーフィールドのキーワードインデックスに含まれるキーワードのリストを取得できます。
キーワードインデックスが割り当てられたピクチャーフィールドにこのコマンドが適用されると、引数として渡された配列にインデックスのキーワードが返されます。

注 ピクチャーフィールドにキーワードインデックスが設定されていない場合、空の配列が返されます。

例えば :

```
ALL RECORDS([PICTURES])
ARRAY TEXT(<>_MyKeywords;10)
```

DISTINCT VALUES([PICTURES]Photos;<>_MyKeywords)

- 特定のピクチャーに割り当てられているキーワードのリストを取り出す場合、新しいコマンド GET PICTURE KEYWORDS (168 ページ) を使用できます。

ピクチャーファイルのデフォルト名

4D v13 ではフィールドに格納された各ピクチャーごとにデフォルト名を記録できます。つまりユーザーによる書き出しや WRITE PICTURE FILE コマンド等を (fileName 引数に空の文字列を渡して) 使用してピクチャーフィールドの内容をディスクに保存する際に使用されるデフォルトのファイル名を設定できます。

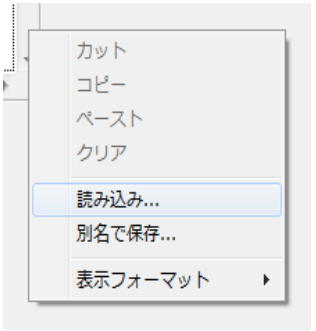
フィールドの内容を変数や他のフィールドにコピーした場合、デフォルト名もコピーされます。

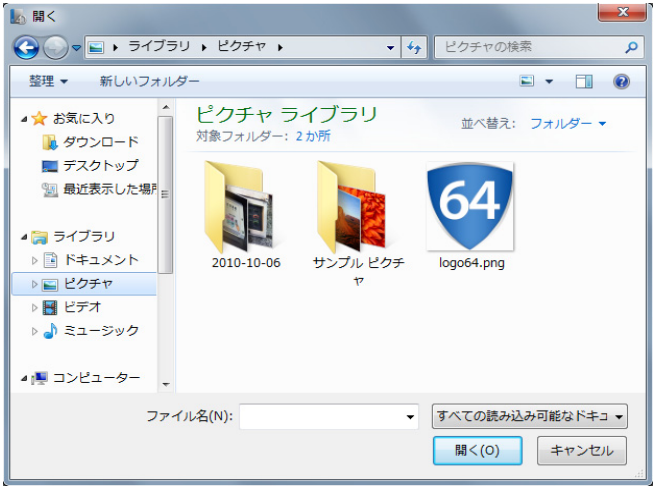
ピクチャーフィールドに格納されるピクチャーのデフォルト名を設定する方法は 2 つあります :

- 新しいコマンド SET PICTURE FILE NAME (168 ページ) を使用してプログラムで行う。このコマンドを使用してピクチャーのデフォルトファイル名を割り当てられます。また新しいコマンド Get picture file name (167 ページ) を使用してピクチャーのデフォルト名を取得することもできます。
- コンテキストメニューや READ PICTURE FILE コマンドを使用してピクチャーフィールドにピクチャーファイルの内容を読み込む際は自動で設定されます。この場合 4D は元のピクチャーファイルの名前を記録します。

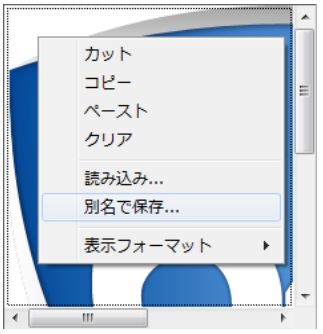
これは以下のように動作します :

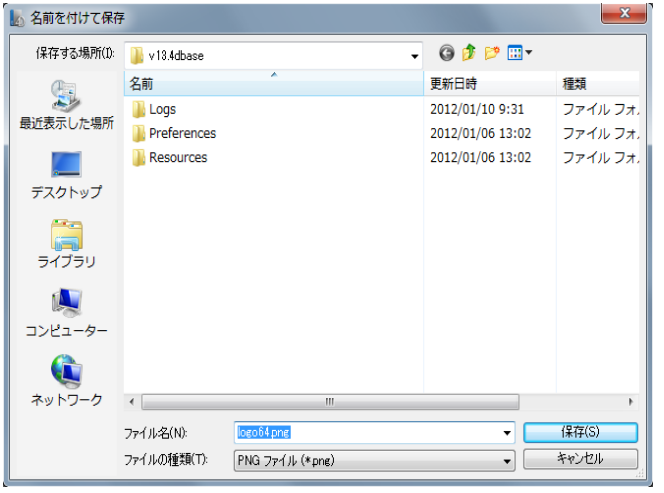
- 1. ユーザーがピクチャーフィールドにファイル名 logo64.png を読み込みます :





- 2. 次にユーザーがレコードからピクチャーを書き出します。保存ダイアログには logo64.png というファイル名がデフォルトで表示されます (これはデフォルトで提案される名前であり、変更が可能です):





データをデータファイル外に保存

4D v13 では各 BLOB、ピクチャー、テキストフィールドごとにデータの保存先を指定できます。既存の保存オプション (レコード内、またはレコード外) に加え、データファイル外への保存を選択できます。この場合データはディスク上の外部ファイルとして保存されます。外部ファイル

はサードパーティアプリケーションで変更することができますが、この場合データ整合性は開発者が管理しなければなりません。

この動作はユーザーに対し透過的です。データの場所に関わらず、データへのアクセスは同じです。

ファイルの外に保存できることで、アプリケーションの動作を最適化することができます。例えば大量のデータを外部に保存できればその分データファイルのサイズを減らすことができます。またデータベースが開かれていないときでも OS からそれらのファイルにアクセスすることができます。

自動モード / カスタムモード

データファイル外部への保存は 2 つのモードで実行できます :

- **自動モード** : このモードでは、4D は特定の構造のフォルダーを自動で作成・管理し、すべてのデータをそこに格納します。この場合データは透過的に管理され、フィールドデータがデータファイル内部に格納された場合と同様に動作します。
  この自動モードを有効にするには以下のいずれかの方法を取ります :
  - ストラクチャーエディターで外部への格納を設定する (設定はデータベースストラクチャーに保存されます)。
  - 新しいSET EXTERNAL DATA PATHコマンドのpath引数に空の文字列を渡す (設定は 4D を終了するまで有効)。
- **カスタムモード :** このモードでは各フィールドおよび各レコードごと、外部ファイルの保存場所を自由に選択できます。この場合 4D はフィールドとデータの間のリレーションのみを管理し、データベースメカニズムの一部は動作しなくなります。
  カスタムモードは新しい SET EXTERNAL DATA PATH コマンドを使用して、デフォルト以外のフォルダーを指定することで有効にされます。

以下の表では自動およびカスタムモードの機能を比較します :

	自動モード	**カスタムモード**
格納先フォルダーの設定	名前と場所は 4D が設定します。データベース全体で 1 つのデフォルトフォルダーを使用します。	名前と場所を自由に設定可能。各フィールドごとに変えることもできます。
外部ファイルの作成、ロード、保存	自動	自動
レコードが削除された際の外部ファイルの削除	はい	いいえ

フィールドに NULL 値が割り当てられたときの外部ファイルの削除	はい	いいえ
データベースが保存された際の自動統合	はい	いいえ
ログファイル統合の際の自動サポート	はい	いいえ
標準インデックスのサポート (テキストフィールド)	いいえ	いいえ
キーワードインデックスのサポート (テキストとピクチャー)	はい	いいえ
トランザクションのサポート	はい	いいえ

外部データの場所

データファイルの外に保存されたデータは以下の原則に基づき整理されます :

- 各レコードごと、データは xxx.txt (テキスト)、xxx.blob (BLOB)、xxx.jpg や xxx.tiff (ピクチャー、拡張子はピクチャータイプに基づく) という名前で保存されます。xxx は 4D が管理する UUID です。
- 自動モードでは、すべての外部データが **<DatabaseName>.ExternalData** という名前のフォルダーに配置されます (DatabaseName はデータベースストラクチャーの名前です)。このフォルダーはデータファイル (.4DD) と同階層に作成されます。
  4D はこのフォルダー内に、外部にデータを格納するフィールドを持つテーブルごとにフォルダーを作成し (名前は "Table"+ テーブル番号)、さらに各外部フィールドごとにサブフォルダーを作成します (名前は "Field"+ フィールド番号)。フィールド中の最初の 100 項目はフォルダーの第一レベルに格納されます。それ以降は "2" から始まるサブフォルダーが順次作成され、その中に 100 要素までが格納されます。例えば "Accounting" データベースのテーブル番号 3 のフィールド番号 5 のデータがデータファイルの外に格納される場合、以下の構成でフォルダーやファイルが 4D により作成されます :

```
Accounting.4DD
[Accounting.ExternalData]
  Table3
    Field 5
      Data_1B7F3A 56F6544B45951EFA60426D5ABC.txt
      Data_1B7F3A 56F6544B45951EFA60426D5CCC.txt
      ...
      2
        Data_2ADBFBA478AAE4409DA9C2D13C90A53B.txt
        Data_32F8A30B87EE7E4BBC802468D553DC43.txt
        ...
```

- カスタムモードでは新しい SET EXTERNAL DATA PATH コマンドを使用して、各フィールドごと個別にフォルダーの名前と場所を自由に設定できます。この設定はデータベースストラクチャーには保存されません。

ファイルの作成と更新

フィールドの外部ファイルへの記録は、レコードがディスクに保存される際に行われます (必要であればトランザクションが有効にされた後):

- 外部ファイルが存在しなければ作成されます。
- 外部ファイルが既に存在する場合、4D はそれを新しいファイルで置き換えます。
  既存のファイルを保持したい場合、(SET EXTERNAL DATA PATH コマンドを使用して) 異なるパスを指定するか、レコード保存前に既存の外部ファイルからその内容をメモリにロードするために新しい RELOAD EXTERNAL DATA コマンドを使用します。この方法はレコードがロードされた後、外部ファイルが他のアプリケーションによって変更された場合にも有効です。

同期と複製

データの格納場所は各データベースごとのローカルパラメーターです。データの同期や複製を行う場合、 これらのパラメーターはローカルデータベースとリモートのそれとでは異なるかもしれません。 この場合データの格納は各データベースのパラメーターに従います。 同期や複製は設定を変更しません。

例えばリモートデータベースにおいてピクチャーフィールドが外部ファイルとして保存され、同じフィールドがローカルデータベースではデータファイル内に保存されている場合、複製が行われると、ローカルデータベースのこのフィールドに追加されたすべての (データファイル内の) データは、リモートデータベースのデータファイル外に保存されます。

ファイルへの外部からのアクセス

外部に格納されたファイルは 4D 以外のアプリケーション (OS、テキストやグラフィックエディター等) から読み書きモードでアクセスできます。しかしこれはアプリケーションの動作を変更してしまうかもしれないので、注意して行わなければなりません。

- 外部ファイルが OS やその他のアプリケーションによって削除、名称変更、あるいは移動されると、4D は対応するフィールドが NULL 値を持つものとして扱い、レコードが保存されるときに (値が NULL でなければ) 新しいファイルが作成されます。
- インデックスを使用している場合、サードパーティアプリケーションが外部ファイルを変更しても、親レコードがディスクに保存されるまでは、インデックスが更新されません。

注 外部テキストファイルは BOM なしの UTF-8 で保存されます。サードパーティアプリケーションでそのファイルを開き、BOM 付きで保存された場合、そのファイルを 4D で開くことはできますが、保存の際に BOM は失われます。

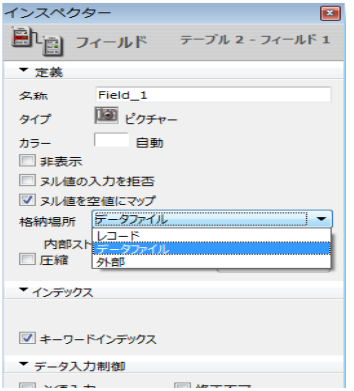
読み込みのみレコードとファイル

" 読み込みのみ " モードでレコードをロードしても、このレコードの対象フィールドの外部ファイルはロックされません。ディスクファイルはその内容が 4D のメモリに読み込まれていますが、4D やその他のアプリケーションを使用して編集が可能です。

ストラクチャーエディターで外部への格納を設定する

4D ストラクチャーエディターを使用して BLOB、ピクチャー、テキスト型フィールドの外部への格納を指定できます。この場合格納場所はストラクチャーファイルに永続的に保存され、データアクセスメカニズムは完全に 4D が管理します (自動モード)。

このオプションはフィールドインスペクターの新しいポップアップを使用して設定します :



以下の選択肢があります :

- **レコード**と**データファイル** : これらのオプションはそれぞれ以前のバージョンの 4D と同じです。
  - **レコード**: 以前のバージョンではテキストフィールドのみで利用できました。
    このオプションは、標準の B ツリーインデックスを利用する場合に必要となるため、バージョン 13 で作成されたテキストフィールドにデフォルトで選択されています。
  - **データファイル**: 以前のバージョンのBLOBとピクチャー (そして設定によってはテキスト) フィールドの標準格納場所。

このオプションは BLOB とピクチャーフィールドでデフォルトで選択されます。

- **外部** : この新しいオプションを使用してデータを .4DD ファイルの外部にディスクファイルとして保存できます。この点については 23 ページの " 外部データの場所 " で説明しています。

互換性に関する注意

このオプションは設定を行った後に新規に作成されたレコードに対してのみ有効です。レコードが既に存在するテーブルで設定を行った場合、既存レコードのデータの格納場所は変更されず、内部 / 外部格納モード混在で動作します。
外部モードを既存のレコードにも適用したい場合、新しい**レコードの強制更新**オプションを使用してデータを圧縮します (54 ページの "Maintenance & security center" 参照)。

内部ストレージの最大サイズ

データファイルまたは外部を選択した場合 BLOB、ピクチャー、テキストすべての型で、**内部ストレージの最大サイズ**オプションを使用できます。これらの型で可変サイズのデータを処理する場合、サイズの条件に応じて格納場所を調整し、パフォーマンスを調整することができます。
このオプションは先のバージョンの 4D と同様に動作します。このエリアに入力される値はバイトを表し、これよりも小さなサイズのデータは設定された格納場所に関わらずレコードに保存されます。
例えばピクチャーフィールドで 30 000 を設定すると、20KB のピクチャーはレコードに保存され、40KB のピクチャーは指定された場所に保存されます (**データファイル**または**外部**)。デフォルト値は 0 で、すべてのデータがレコード外に保存されます。

3 開発環境

この章では 4D v13 の開発環境の、フォームやツールを使用したアプリケーション構築レベルにおける、新機能と変更点について説明します :

- 拡張されたリストボックスの機能
- パターンパレットの近代化
- Web エリアへの WebKit の統合
- Windows における新しい Direct2D グラフィックレンダリングエンジン
- Windows における XPS を使用した印刷プレビュー
- 依存関係検索の変更
- 新しい MSC オプション
- アプリケーションビルダーの新しいログフォーマット
- ユーザー設定の外部ファイル化

リストボックス

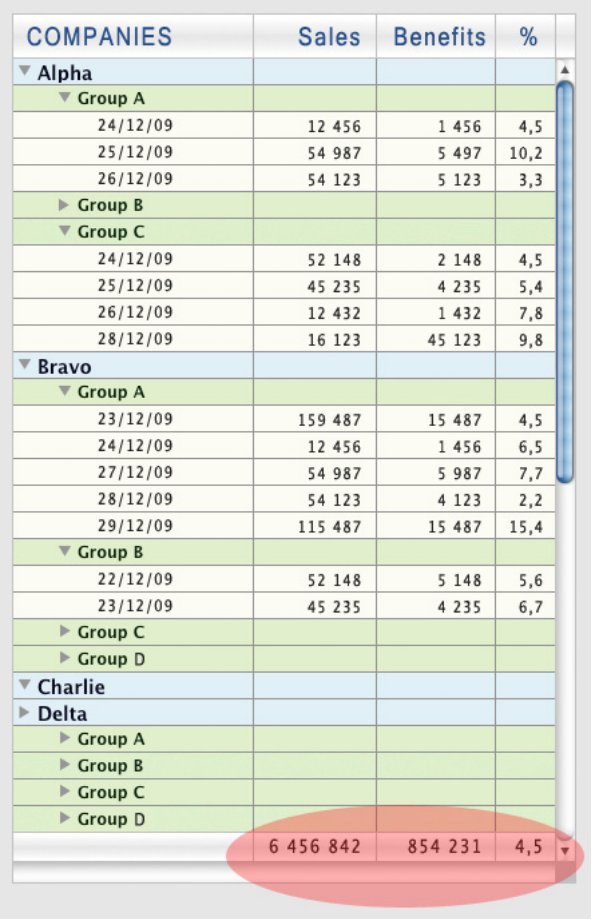
4D v13 でリストボックスはその機能が大幅に拡張されました。

注 多くの新機能にはリストボックスランゲージコマンドからアクセスする事もできます。詳細は 125 ページの " リストボックス " を参照してください。

フッター

4D v13 よりリストボックスにフッターを表示できるようになりました。このエリアはユーザーによる入力ができず、追加の情報を表示するために使用されます。表形式で表示されるデータに対し、通常フッターは合

計や平均などの計算値を表示するために使用されます。以下はフッターが表示されたリストボックスの例です :



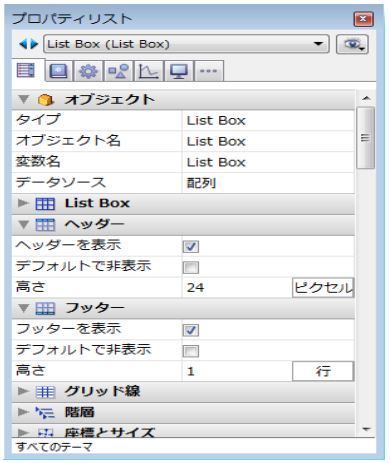
COMPANIES	Sales	Benefits	%
Alpha			
Group A			
24/12/09	12 456	1 456	4,5
25/12/09	54 987	5 497	10,2
26/12/09	54 123	5 123	3,3
Group B			
Group C			
24/12/09	52 148	2 148	4,5
25/12/09	45 235	4 235	5,4
26/12/09	12 432	1 432	7,8
28/12/09	16 123	45 123	9,8
Bravo			
Group A			
23/12/09	159 487	15 487	4,5
24/12/09	12 456	1 456	6,5
27/12/09	54 987	5 987	7,7
28/12/09	54 123	4 123	2,2
29/12/09	115 487	15 487	15,4
Group B			
22/12/09	52 148	5 148	5,6
23/12/09	45 235	4 235	6,7
Group C			
Group D			
Charlie			
Delta			
Group A			
Group B			
Group C			
Group D			
	6 456 842	854 231	4,5

フッターエリアへのアクセス

フッターはセレクションおよび配列、また階層 / 非階層いずれのタイプのリストボックスでも利用できます。

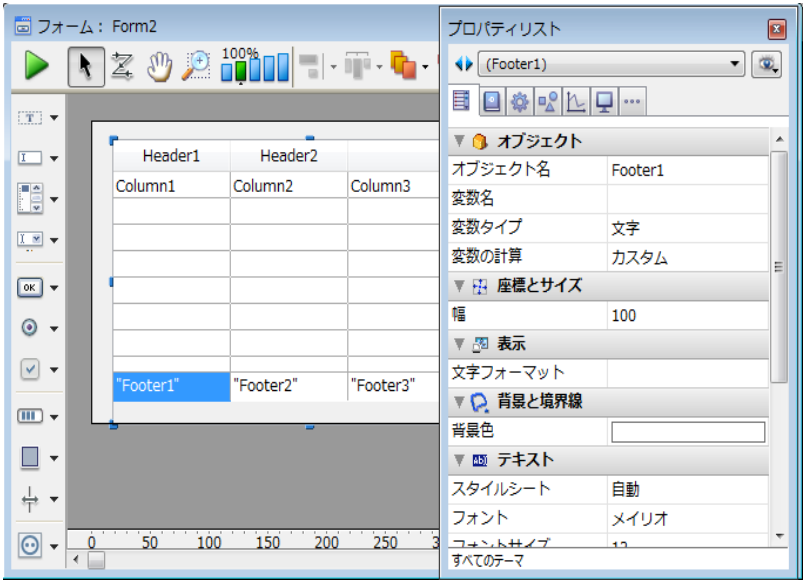
警告 : ヘッダーエリアと同様、フォーム実行中はフッターエリアにユーザーが入力を行うことはできません。表示させる値は 4D に自動計算させるか、プログラムで設定します。

フッターはリストボックスのために追加された新しいエリアです。フッターを利用するためにはリストボックスのプロパティリストで**フッターを表示**が選択されていなければなりません :



一列につきひとつのフッターがあり、各フッター毎に設定が可能です。

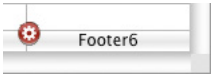
リストボックスへのフッター表示設定を行ったら、フッターセルをクリックして選択し、プロパティリストからさらに詳細な設定を行うことができます :



フッターエリアのプロパティ

プロパティリストおよびランゲージコマンドを使用してフッターエリアを定義できます。(コマンド LISTBOX SET FOOTER CALCULATION (132 ページ) 参照).

フッターエリアには以下のプロパティがあります :

- **オブジェクト名**と**変数名** : フッターエリアはフォーム内でユニークなオブジェクト名、および変数を割り当てることができます。デフォルトで変数名は空であり、4D はダイナミック変数を使用します。
- **変数タイプ** : このメニューで変数の型を設定し、またプロパティリストの他のオプション表示を更新します。ダイナミック変数を使用しない場合、変数の型定義はランゲージを使用して行わなければなりません。
- **変数の計算** : このオプションを使用してフッターエリアに適用される計算タイプを設定します。いくつかの計算オプションの他、**カスタム**も選択できます。
  - **最小値**, **最大値**, **合計値**, **カウント**, **平均値**, **標準偏差**, **分散値**および**平方和**。これらの計算は 30 ページの " 自動計算 " で説明されています。計算が選択されると、リストボックス列中のすべての値が自動で計算の対象となります。リストボックス行の表示 / 非表示状態は考慮されません。表示行だけを計算対象にしたい場合、カスタムを選択してプログラムコードで計算します。
    フッターエリアに自動計算が設定されるとフォームエディター上では " 標準アクション " バッジが表示されます :
    

  - **カスタム** : このオプションを選択した場合、4D は自動計算を行いません。プログラムを使用して表示する値をエリアに代入しなければなりません。
- **座標とサイズ**, **表示**, **背景と境界線**, **テキスト** : これらのプロパティは (奇数行背景色がないことを除き) 列のそれと同じです。
- **ヘルプ** : 各フッターには個別にヘルプメッセージを割り当てられます。

自動計算

フッターエリアには様々な自動計算を割り当てることができます。以下の表は列のタイプに基づき使用することのできる計算を示しています :

	数値	テキスト	日付	時間	ブール	ピクチャー
最小値	○		○	○	○	
最大値	○		○	○	○	
合計値	○			○	○	

	数値	テキスト	日付	時間	ブール	ピクチャー
カウント	○	○	○	○	○	○
平均値	○			○		
標準偏差 (*)	○			○		
分散値 (*)	○			○		
平方和 (*)	○			○		

(*) 配列表示モードのリストボックスのみ

新しい On Footer Click フォームイベント

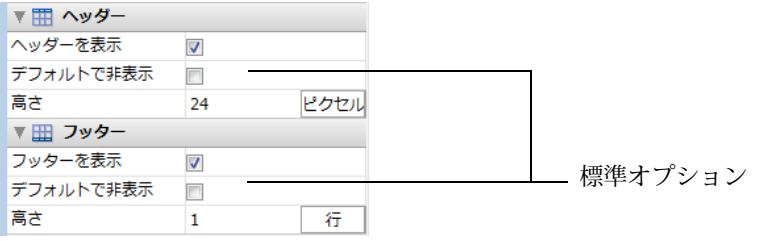
リストボックスオブジェクトおよびリストボックス列に新しいフォームイベント **On Footer Click** が追加されました。

このイベントはフッターエリアがクリックされたときに、リストボックスおよびリストボックス列で生成されます。このイベントを使用して例えばコンテキストメニューを表示したり、カスタムアクションを実行させたりできます。

注 コマンド Form event (110 ページ) の説明も参照してください。

デフォルトで非表示

ヘッダーおよび新しいフッターに新しいオプション**デフォルトで非表示**が加わりました :



すべてのフォームオブジェクトと同様、このオプションは OBJECT SET VISIBLE コマンドを使用したオブジェクトの表示切り替えを容易にします。

" デフォルトで非表示 " オプションはそれぞれ " ヘッダーを表示 " や " フッターを表示 " オプションが選択されているときにのみ利用可能である点に留意してください。ヘッダーやフッターを表示するオプションが選択されていない場合、OBJECT SET VISIBLE コマンドは効果がありません。

行、ヘッダー、フッターの高さ

4D v13 ではリストボックスの行の高さ設定に関連する 2 つの主要な新機能が提供されます :

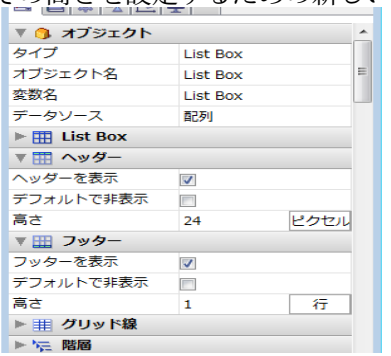
- ヘッダーやフッターの高さをそれぞれ個別に設定できるようになりました。
- 今までと同様ピクセル単位の他に、行単位で高さを設定できるようになりました。

ヘッダーとフッターの高さ

リストボックスのヘッダーとフッターの高さをそれぞれ設定できます。これを行うには 2 つの方法があります :

- プロパティリスト中の新しいプロパティを使用する (後述)
- 新しい LISTBOX SET HEADERS HEIGHT や LISTBOX SET FOOTERS HEIGHT コマンドを使用する (" ランゲージ " 参照)

リストボックスのプロパティリストにヘッダーやフッターの表示およびその高さを設定するための新しいテーマが 2 つ追加されました :



注 高さ設定の単位として行またはピクセルを選択できます。行の高さの単位については以下の節を参照してください。

デフォルトでヘッダーおよびフッターの高さは **1 行**です。この値は以前のバージョンから変換されたデータベースにも適用されます。

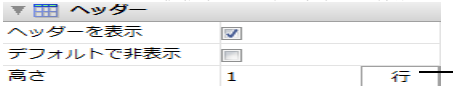
ピクセル単位を使用したヘッダーの最小の高さはシステムにより異なります。最小値よりも小さい値を渡した場合、システムに設定された最小値で上書きされます。フッターと行に最小値はありません。

互換性に関する注意 Windows 7 と Windows Vista では、ヘッダーの高さ最小値が 24 ピクセルです。変換されたデータベースでヘッダーの高さがこれよりも低く設定されていた場合、自動でサイズが変更されます。この場合、それに伴ってフォームを変更する必要があるかもしれません。

行単位で設定された高さ

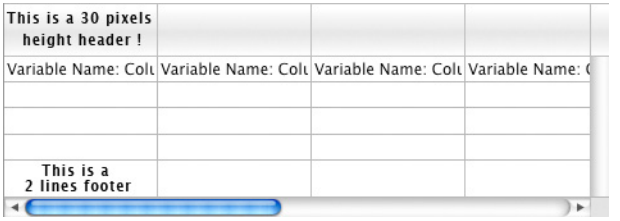
以前のバージョンの 4D では行の高さをピクセル単位でのみ設定することができました。4D v13 ではヘッダー、フッター、および行に対し、高さを**行**単位で設定できるようになりました。

新しい LISTBOX SET COLUMN FORMULA, LISTBOX SET FOOTERS HEIGHT コマンドおよび LISTBOX SET ROWS HEIGHT コマンドで特定の引数を使用して、あるいはプロパティリストの " 高さ " フィールドの右にあるボタンを使用して、単位を指定できます



注 または高さを入力するエリアに直接単位を入力することもできます。行単位の場合は "L"、ピクセル単位の場合は "P" を入力します (例えば "17 P")。ボタンラベルは入力に基づき更新されます。

同じリストボックス内でヘッダーとフッター異なる単位を使用することもできます :



- " ピクセル " を使用した場合、指定された値は関連する行に対し直接適用されます。列に使用されるフォントサイズ等は考慮されません。フォントが行の高さに対して大きい場合、テキストは切り取られます。ピクチャーはフォーマットに基づき切り取られるかリサイズされます。
- " 行 " を使用した場合、高さは関連する行のフォントサイズに合わせて計算されます。複数の異なるサイズが設定されている場合、4D はもっとも大きなものを使用します。
例えば行に "Verdana 18", "Geneva 12" そして "Arial 9" が設定されている場合、4D は行の高さの決定に "Verdana 18" を使用します。複数行の場合はこの高さの倍数が使用されます。

注 この計算にはピクチャーのサイズやフォントに適用されるスタイルは考慮されません。

- **単位の変換** : 単位を変更した場合、4D は自動で値を再計算し、結果をプロパティリストに表示します。例えば使用されるフォントが "Lucida grande 24" で高さが "1 行 " に設定されていれば "30 ピクセル " に、"2 行 " なら "60 ピクセル " になります。

単位の変更を繰り返すと、4D が自動で計算を行うため、最初の値とは結果が異なってしまうこともあります。例えば以下のようになります :
(Arial 18): 52 ピクセル -> 2 行 -> 40 ピクセル
(Arial 12): 3 ピクセル -> 0.4 行が 1 行に変換される -> 19 ピクセル

縦方向の整列

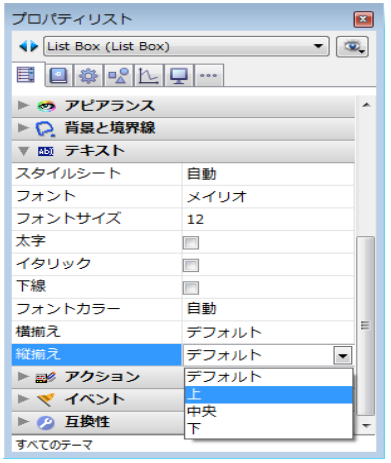
リストボックスで縦方向の整列を設定できるようになりました。

Header1	Header2	Header3
Vertical alignment: Top	Vertical alignment: Center	Vertical alignment: Bottom

この新しい動作を使用するには 2 つの方法があります :

- デザインモードでプロパティリストを使用する (後述)
- 新しいコマンド OBJECT SET VERTICAL ALIGNMENT (162 ページ) を使用する

リストボックスやその要素に縦方向の整列を設定するには、テキストテーマの**縦揃え**メニューから設定値を選択します :*



この設定はリストボックス全体に適用でき、さらに各列やヘッダー、フッターごとに設定を上書きできます。

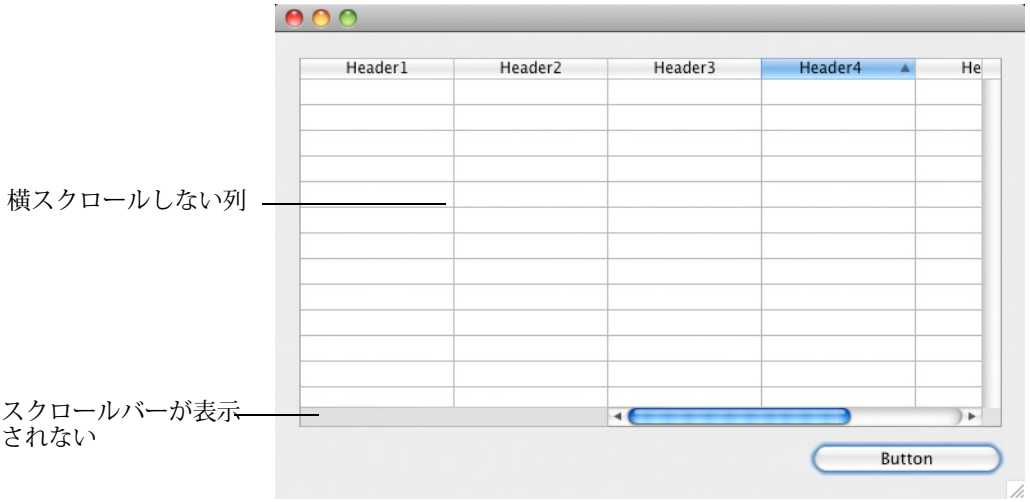
デフォルトオプションを選択した場合 :

- ピクチャーを除くすべての型は**下**が、
- ピクチャーは**上**が使用されます。

横スクロールしない列

4D v13 からは横スクロールしない列を設定できるようになりました。

この設定がされた列はユーザーが横スクロールを行ってもリストボックスの左側に常に表示されます。またそれらの列の下には横スクロールバーが表示されません :



通常の列と同様、横スクロールしない列の幅を変更したり、入力したりすることはできます。スクロールしない点のみが異なります。

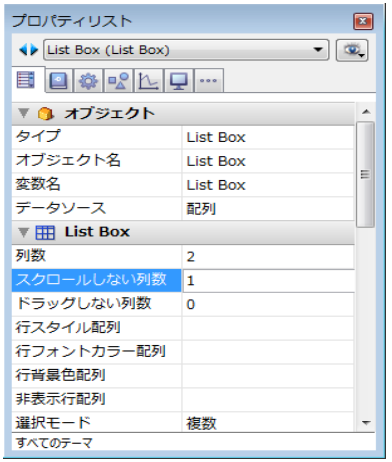
これは大きな表で行のタイトルを表示するような場合に便利です。

横スクロールしない列を設定

横スクロールしない列を設定する方法は 2 つあります :

- デザインモードでプロパティリストを使用する。
- 新しいコマンド LISTBOX SET LOCKED COLUMNS (135 ページ) を使用する

フォームエディターで横スクロールしない列を設定するには、リストボックスを選択して**スクロールしない列数**オプションに設定値を入力します :



注 このオプションを設定すると " 横スクロールしない " エリアを設定できます。この設定が行われた列をプログラムで削除すると、横スクロールしない列数はその分減らされます。同様にこのエリアに列が挿入されると、その列は自動で横スクロールしない列となります。

横スクロールしない列とドラッグしない列

リストボックスの横スクロールしない列とドラッグしない列はそれぞれ独立して動作します :

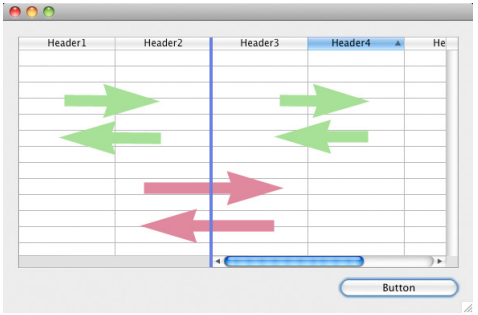
- 横スクロールしない列は常にリストボックスの左側に表示されます。横スクロールされません。
- ドラッグしない列はリストボックス中でドラッグ & ドロップによる列の移動ができません。

注 4D v13 ではドラッグしない列を新しいコマンド LISTBOX SET STATIC COLUMNS (136 ページ) で設定できます。

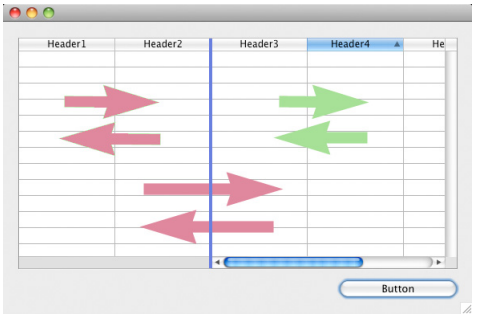
これらのプロパティは以下のように相互作用します :

- 列をドラッグしないのみに設定した場合、(以前のバージョンの 4D と同様) その列は移動することができません。

- 列を横スクロールしないのみに設定した場合、横スクロールしないエリア内ではドラッグで列を移動することができます。しかしエリアを越えて移動することはできません。



- 横スクロールしない列とドラッグしない列を同じ数に設定した場合、スクロールしないエリア内ではドラッグで移動することもできません。



- 必要に応じて横スクロールしない列数とドラッグしない列数をそれぞれ設定できます。例えば横スクロールしない列を 3、ドラッグしない列を 1 に設定した場合、ユーザーは横スクロールしないエリア内で右側 2 つの列を入れ替えることができます。

階層リストボックスで使用する新しいイベント

階層リストボックスの表示と動作を最適化する 2 つの新しいフォームイベント **On Expand** と **On Collapse** が追加されました。

階層リストボックスは配列の内容を使用して構築され、その配列の内容はすべてメモリにロードされなければなりません。このため SELECTION TO ARRAY コマンドを使用するなどしてデータから配列を生成し、大きな階層リストボックスを使用することが困難になることがありました。これは配列の生成に時間がかかったり、使用するメモリ量が大きくなったりするためです。

On Expand と **On Collapse** フォームイベントを使用すればこの制限を回避することができます。例えばユーザーアクションに基づき、表示され

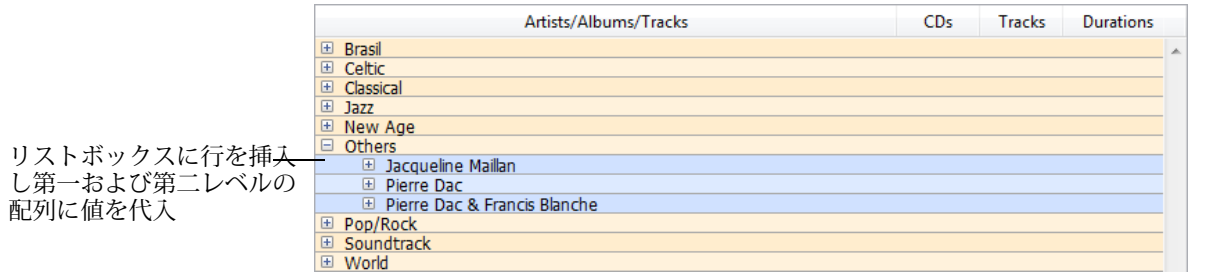
る階層部分だけをメモリにロードし、閉じられた階層部分はメモリからアンロードすることができます。

これらのイベントのコンテキストで、LISTBOX GET CELL POSITION コマンドはユーザーが階層を展開 / 折りたたむためにクリックしたセルを返します。

この場合開発者がプログラムコードを使用して配列要素を挿入したり削除したりしなければなりません。実装の指針は以下の通りです :

- リストボックスが最初に表示されるとき、第一階層の配列のみ値が埋められます。しかし展開ボタンを表示するために、第二階層以降の配列も同じ要素数を空の値で作成しなければなりません :

- ユーザーが展開ボタンをクリックしたら **On Expand** イベントが生成されます。LISTBOX GET CELL POSITION コマンドを使用すればクリックされたセルを取得できるので、適切な階層を作成します。まず第一レベルはクリックされた列と同じ値を繰り返します。そして第二レベルは SELECTION TO ARRAY コマンド等で作成した値を使用します。LISTBOX INSERT ROWS コマンドを使用して必要な行数を挿入して、リストボックスの列配列に値を設定します。



- ユーザーが折りたたみボタンをクリックすると、**On Collapse** イベントが生成されます。LISTBOX GET CELL POSITION コマンドを使用すればクリックされたセルを取得できるので、LISTBOX DELETE ROWS コマンドを使用して必要なだけ行を削除します。

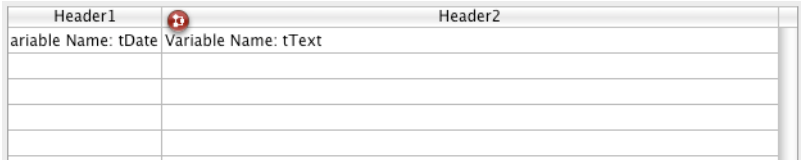
セルへの入力制御

セレクション呼び配列モード両方のリストボックスで、セルへの入力可否の制御が変更されました。

リストボックスセルへの入力を許可するためには以下の条件を満たす必要があります :

- そのセルが属する列を**入力可**に設定する (この設定がされていない場合、そのセルに入力を行うことはできません)。
- **On Before Data Entry** イベントで $0 に -1 が**渡されない**。
4D v13 ではカーソルがセルに設定されると、列メソッドで **On Before Data Entry** イベントが生成されます。このイベントのコンテキストで $0 に -1 が設定されると、そのセルは入力不可として扱われます。**Tab** や **Shift+Tab** が押された後にイベントが生成され、入力が拒否された場合、フォーカスはその次あるいは前のセルに送られます。$0 が -1 でない場合 (デフォルトで $0 は 0)、セルは入力可として扱われ、入力モードに移行します。

♦ 2 つの配列 (日付とテキスト) を表示するリストボックスで考えてみましょう。
日付配列には入力を許可せず、テキスト配列には該当行の日付を過ぎていなければ入力できるものとします。



arrText 列のメソッドは以下のとおりとなります :

```
C_LONGINT($0)
Case of
  :(Form event = On Before Data Entry) // セルがフォーカスを得た
     LISTBOX GET CELL POSITION(*;"lb";$col;$row)
       // クリックされた列を取得
     If(arrDate{$row} < Current date) // 日付が過ぎていれば
       $0:=-1 // セルは入力不可
     Else
       // そうでなければ入力可
     End if
End case
```

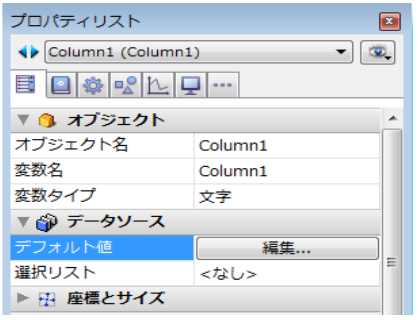
互換性に関する注意

4D v13 より、**On Before Data Entry** イベントは **On Getting Focus** よりも前に生成されるようになりました。この変更により、以前のバージョンで **On Before Data Entry** 中、Get edited text コマンドを使用して入力され

たテキストを読み取っていた場合、入力可能なセルのプログラムを変更する必要があるかもしれません。4D v13 でこのコマンドはこのコンテキストで空の文字列を返します。データ入力を処理するためには **On Getting Focus** イベントを使用してください。

デフォルト値

リストボックス列にデフォルト値を割り当てることができるようになりました。この設定を行うには列のプロパティリスト中**デフォルト値**欄にある**編集 ...** ボタンをクリックします :



標準の編集ダイアログが表示されるので、値のリストを改行で区切って入力できます。設定された値はフォームが実行された際に自動で配列の要素値として使用されます。

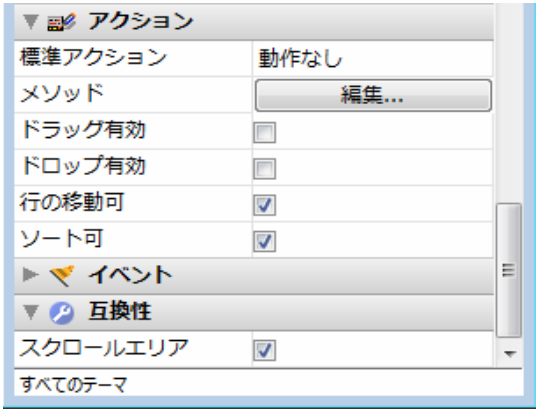
スクロールエリアの互換性

v13 に変換されたデータベースでは以前の " スクロールエリア " が自動でリストボックスに変換されます (13 ページの " スクロールエリア " 参照)。元のインターフェースを再現するため、このリストボックスはフォームエディター中 2 つのオプションを使用し、特別なモードで動作します :

- " 互換性 " テーマの**スクロールエリア**オプション
- **グループ化したリストボックス**を使用していた場合、リストボックスの**接続**

"スクロールエリア" オプション

スクロールエリアから変換されたリストボックスのプロパティリストでは、自動で**スクロールエリア**オプションが選択されます :



リストボックスでこのオプションが選択されていると以下の特別な動作を行うようになります :

- リストボックス列配列に非表示プロパティが設定されていると、リストボックス全体が非表示になります。
- リストボックス列の配列変数に数値を代入すると、当該行が選択されます (例 : MyArray:=5 とするとリストボックス中 5 番目の要素が選択されます)。
- 逆に行をクリックすると列配列の現在値が更新されます。
- リストボックス行から他のオブジェクトにドロップが行われたとき、このオブジェクト内で実行される DRAG AND DROP PROPERTIES コマンドは (リストボックス自身ではなく) リストボックス列への配列を返します。

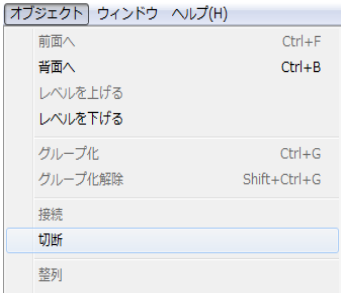
接続されたリストボックス

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 :

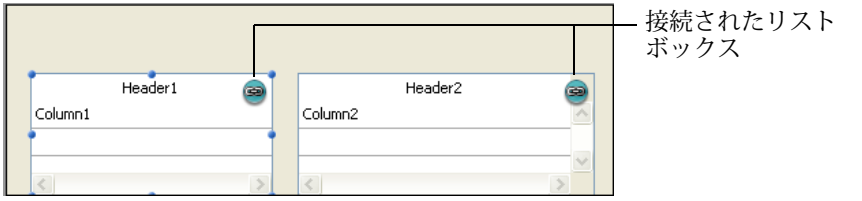
- ひとつのリストボックス上で行を選択すると、接続されたグループに属するすべてのリストボックスの同じ行が選択されます。
- リストボックスをスクロールすると、接続されたグループに属するすべてのリストボックスがスクロールされます。

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、新しい**接続**および**切断**メニューを使用して、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た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 :



互換性に関する注意

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に置き換えることを推奨します。

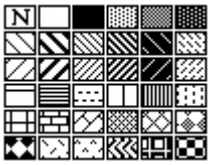
フォームエディターの更新

4D v13 では以下の点においてフォームエディターが更新されました :

- 塗りつぶしと境界線パターンが削除され、既存のパターンは自動で変換されます。
- 廃止されたコマンドが**オブジェクト**メニューから削除されました。

以前のパターンの変換

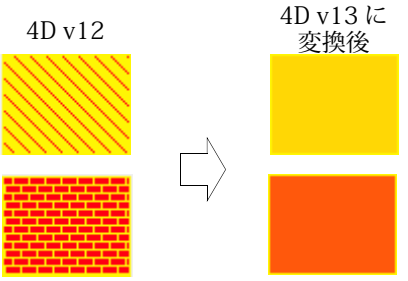
塗りつぶしや境界線のパタンサポートは 4D v13 で停止されました。この機能はもともと白黒インターフェース用に存在するものです :



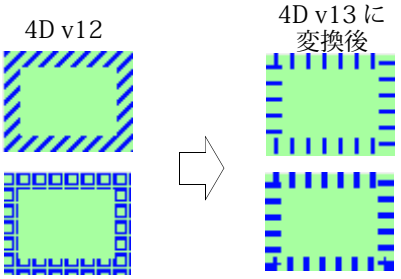
4D フォームエディターにて、パターンを設定するためのメニューコマンドとプロパティリスト項目は削除されました (**境界線** , **塗りつぶし**)

変換されたデータベースでは、塗りつぶしや線パターンを使用したオブジェクトは可能な限り当初の描画を保持するよう自動で変更されます。この変換は以下のように行われます :

- テキストの塗りつぶしパターン (フィールド、変数、スタティックテキスト) は取り除かれます。
使用されたパターンが " 黒 " (パレット中の三番目) だった場合、パターンは取り除かれ、背景色とフォント色の組み合わせは保持されます。
- 描画オブジェクト (四角、楕円等) の塗りつぶしパターンは背景色で置き換えられます。このカラーはそのオブジェクトのパターンで使用された以前の " 行の色 " と " 塗りつぶし色 " プロパティの組み合わせです。色を決定する際、4D は元のパターンに配慮した特別な係数を使用します。
例えば四角を変換した場合、以下の結果が得られます :



- グラフィックオブジェクトの線パターンは線スタイルで置き換えられます。自動で適用されるスタイルはパターンに基づきます。例えば以下のような結果が得られます :



オブジェクトメニューの更新

廃止あるいは冗長なコマンドが**オブジェクト**メニューから削除されました。以下のメニューが置換あるいは削除されました :

オブジェクトメニューから取り除かれたコマンド	**同等の機能**
線幅	プロパティリスト " 線幅 " フィールド
塗りつぶし	削除 (前述)
境界線	削除 (前述)
カラー	- プロパティリスト " カラー " フィールド - **カラー** コンテキストメニュー
フォント	プロパティリスト " フォント " フィールド
スタイル	プロパティリスト " テキスト " テーマ中のフィールド

Web 描画エンジンの選択

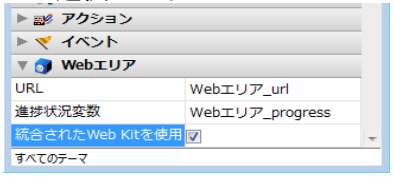
4D v13 では Web エリアの描画エンジンを 2 つのオプション、システム組み込みのエンジンと 4D に統合された WebKit エンジンから選択できるようになりました。

- **システム組み込みのエンジンを使用する (デフォルトオプション)**
この選択により以前のバージョンの 4D と同じ動作となります。ただし Windows において、4D v13 はマシンにインストールされている最新バージョンの Internet Explorer を自動で使用するようになりました。以前のバージョンでは IE7 のエンジンが常に使用されていました。Mac OS ではインストールされているバージョンの WebKit を使用します。
この場合 HTML5 や JavaScript を使用した Web 描画について、最新のエンジンを使用することができます。しかし Internet Explorer と WebKit との実装上の違いを考慮にいれる必要があります。

- **統合された WebKit エンジンを使用する**
  4D v13 にはオープンソースでマルチプラットフォームな WebKit と呼ばれる Web 描画エンジンが統合されました。このエンジンは Safari や Google Chrome 等でも使用されています。
  この統合 WebKit を使用すれば、Web エリアの描画とその動作が (ピクセル単位での若干の相違やネットワーク実装に関連する違いを除き) プラットフォームに関わらず同じになります。
  このオプションが選択されると、OS により行われる自動更新などの利点を得ることができなくなります。使用される WebKit の新バージョンは 4D のリリースを通して提供されます。

注 Windows XP では統合 WebKit エンジンがサポートされません。この場合標準モードが常に使用されます。

4D v13 ではアプリケーションの特性に基づき、Web エリアごとに上記いずれかのモードを選択できます。統合 WebKit エンジンを使用するには Web エリアのプロパティリストから**統合された Web Kit を使用**オプションを選択します :



デフォルトでこのオプションは選択されておらず、Web エリアはシステムの描画エンジンを使用します。

注 統合 Web Kit 描画エンジンを使用する場合、以下の点に留意してください :
- HTTPS プロトコルを使用したページを表示する場合でも、認証ルートは検証されません。
- Windows で PDF ファイルは外部ウィンドウにのみ表示が可能です。

Windows での新しい Direct2D グラフィック描画エンジン

Windows 上で 4D v13 は、デザインモードとアプリケーションモード両方で、4D アプリケーションのほぼすべてのインターフェース (コントロール、図形、テキスト等) 描画のために新しい Direct2D グラフィック描画エンジンを使用します。

概要

Direct2D はベクターグラフィック API で、図形 (弧、四角、ベジェによるパス等) や画像、テキストを表示するために使用されます。

Direct2D は Direct3D に基づくため、図形や画像、テキストを描画する際には直接グラフィックカードのハードウェアアクセラレーターの恩恵を得ることができます。しかし Direct2D はハードウェアサポートがない場合、" ソフトウェア " による実装にスイッチする場合があります。いずれの場合も、以前のバージョンの 4D が Windows で使用していた描画エンジン (GDI/GDIPlus) の性能よりも勝っています。

4D での利用と制御

Direct2D は Windows Vista 以降で利用できます。

4D v13 は Windows のバージョンと利用可能なマシンリソースに基づき、Direct2D の有効化を決定します :

- Windows 7 以降では Direct2D がストラクチャーエディター、コードエディター、SVG 描画用にデフォルトで有効となり、利用可能であればハードウェアアクセラレーションが使用されます。利用できない場合、また他のエディター用には " ソフトウェア " エミュレーション (D2D WARP) が使用されます。
- Windows Vista では、ハードウェアアクセラレーションが利用可能であれば、デフォルトで Direct2D がストラクチャーエディター、コードエディター、SVG 描画用に使用されます。他のコンテキストまたはハードウェアアクセラレーターが利用できない場合、4D は GDI/GDIPlus を使用します。この動作はユーザーにとって透過的です。

注 特定の古いコントロールは Direct2D と互換性がなく、以前の描画エンジンを使用して描画されます。

Get database parameter, SET DATABASE PARAMETER コマンドで新しいセレクターを使用して、アプリケーションでの Direct2D の利用を制御できます (74 ページ参照)。

Windows での XPS 印刷プレビュー

Windows において 4D v13 は XPS ドキュメントとして印刷プレビューを生成するようになりました。XPS (XML Paper Specification) フォーマットはページ記述言語に使用されるオープンな仕様で、Microsoft により開発され Ecma International により標準化されました。現在では Windows アプリケーションで広く使用され、印刷ドキュメントを正確にスクリーンに表示できます。ハードウェア設定の影響は受けません。

注 Mac OS ではプレビューを生成するためにシステムにネイティブで実装されている PDF フォーマットを使用します。

OS ごとのサポート

XPS プレビュードキュメントを生成するために 4D は XPS ドライバーを呼び出します。このドライバーは Windows Vista 以降に統合されていますが、Windows XP でも .NET framework をインストールすることで 利用できます。

Windows のバージョンに基づき、プレビューはネイティブで利用可能な XPS ドライバーあるいは Internet Explorer を使用して開かれます。

以下の表は Windows のバージョンごとに利用可能な XPS ドライバーとビューアーのリストです :

Windows バージョン	**XPS ドライバー**	**XPS ビューアー**
Windows XP	なし	なし
Windows XP + .NET 3.0 framework	あり	IE を利用
Windows Vista	あり	IE を利用 (*)
Windows 7 以降	あり	あり (reader.exe)

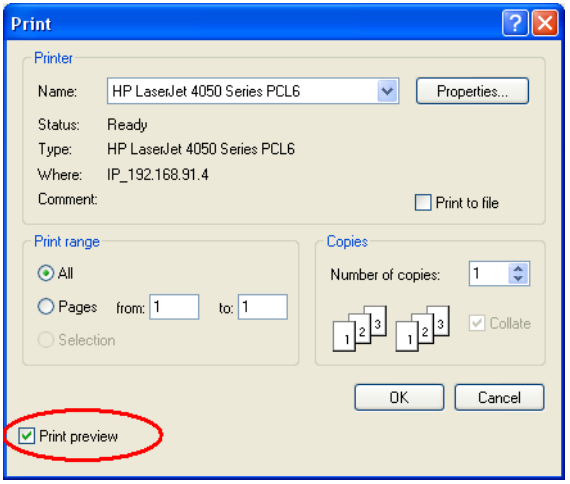
(*) 独立したビューアーもあります。

4D での動作

4D v13 で XPS プレビューは以下のために使用されます :

- デザインモードとアプリケーションモード両方で、すべての 4D フォームとオブジェクト

- ラベルおよびクイックレポートエディター



印刷プレビュー生成が要求されると、4D は印刷を一時的な XPS ファイルに転送し、.XPS ファイルに関連付けられたアプリケーションを起動します。

注 関連付けられたアプリケーションが XPS ビューアーでない場合 (例えばデフォルトブラウザーが IE でない場合)、4D は強制的に Internet Explorer を実行します。

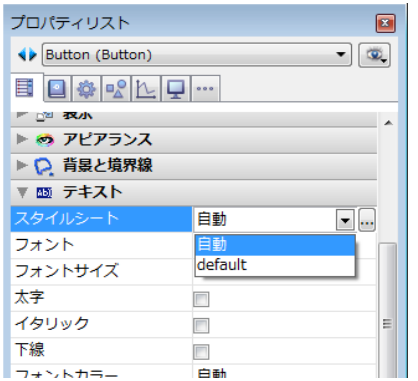
一時的な XPS ファイルはローカルのユーザーフォルダーに保存されます :
...:\Users\{UserName}\AppData\Local\Temp\
4D\PrintPreview\FolderNumber\NumUUID\

このフォルダーがいっぱいになることを避けるために、最新 100 個のドキュメントのみが保持されます。フォルダーへの適切なアクセス権がない場合、-9799 エラーが生成されます。Windows は "Temp" フォルダーを空にすることがある点に留意してください。
ビューアーが自動でドキュメントを開きますが、まだ名前付きで保存されていません。そのために、XPS ドキュメントを格納するための UUID サブフォルダーが連番のフォルダーの下に作成されます。印刷が終了するとドキュメントに名前が付けられます。デフォルトでは印刷が起動されたときのウィンドウ名が使用されます。この名前は印刷前に SET WINDOW TITLE や SET PRINT OPTION (Spooler document name option; "xx") コマンドを使用することで変更できます。

注 4D v13 では 2 つの新しいコマンドを使用してプレビューオプションをコントロールできます (コマンド Get print preview (169 ページ) とコマンド Is in print preview (169 ページ) 参照)。

自動スタイルシート

プロパティリストに新しいスタイルシート、**自動**が追加されました。



注 このスタイルシートはプロパティリストからのみ利用可能です。ツールボックスのスタイルシートページには表示されません。

他のスタイルシートと異なり、" 自動 " スタイルシートには属性が事前には設定されていません。オブジェクトに使用される**フォント**や**フォントサイズ**は以下のシステムパラメーターに基づいて動的に決定されます :

- プラットフォーム
- システム言語
- フォームオブジェクトのタイプ

この自動機能はデザインモードとアプリケーションモードでフォームが実行される都度適用されます。このスタイルシートを利用すればフォームオブジェクトのタイトルが現在のシステムパラメーターに準拠して表示されることを期待できます。そのサイズは条件に応じて変更されます。

" 自動 " スタイルシートはフォントとフォントサイズを管理します。フォームエディターで 2 つのいずれかを変更すると、そのオブジェクトの自動スタイルシートは動的に適用されなくなります。他方自動設定を変更することなくカスタムスタイルプロパティ (**太字** , **イタリック** または **下線**) を設定することはできます。

4D v13 より、フォームエディターに配置される新規オブジェクトにはすべてのこのスタイルシートがデフォルトで適用されます。
変換されたデータベースに既に設定されているスタイルシートは変更されません。既存のスタイルシートを自動で置換したい場合、標準の検索 / 置換機能 (スタイルシート参照をサポートするようになりました) を使用することができます。

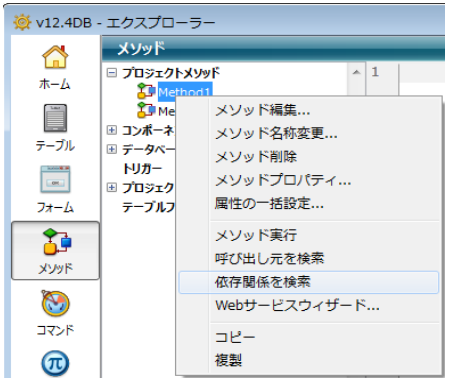
依存関係の検索

プロジェクトメソッドの依存関係を自動で検索できるようになりました。依存関係とはメソッドから直接あるいは間接に参照される他のプロジェクトメソッドやフォームのことです。

結果は検索結果ウィンドウに表示され、それらを選択し、ドラッグ & ドロップを使用して他のデータベースに転送することもできます。これによりデータベース間で特定の機能をコピーすることが簡単にできるようになります。

検索の起動

依存関係を検索するには、エクスプローラーでメソッドを選択し、コンテキストメニューから**依存関係を検索**を選択します :



結果ウィンドウにこのメソッドのすべての依存関係が表示されます。各対応するエディターを使用して項目を開いたり、ドラッグ & ドロップでコピーしたりできます。

複数のプロジェクトメソッドを選択して**依存関係を検索**を選択することもできます。この場合これらすべてのメソッドの依存関係が結果ウィンドウに表示されます。

依存関係の分析

依存するプロジェクトメソッドを検索するために、4D はメソッドコードを解析します :

- コードからの直接呼び出し
- 以下のコマンドに引数として渡されるメソッドの名前 :
  Open window (controlMenuBox 引数)
  ON ERR CALL
  ON EVENT CALL
  New process
  EXECUTE ON CLIENT
  SET MENU ITEM METHOD
  EXECUTE METHOD

ただし検索されるのはこれらのコマンドにテキストリテラル定数として渡されたメソッド名で、かつ実行可能で有効なメソッド名でなければなりません。

コードから依存関係を検索するために、4D は以下のコマンドに引数として渡されたフォーム名を検索します :

- Print form
- DIALOG
- FORM SET OUTPUT
- FORM SET INPUT
- PRINT SETTINGS
- FORM GET PROPERTIES
- Open form window

フォームが属するテーブル引数が認識されますが、明示的に指定されていなければなりません。省略されていた場合、正しく検索できません。

検索できないケース

以下のような場合 4D は依存関係を検索できません :

```
  // メソッド名がリテラルとして渡されていない場合
v:="myhandler"
ON ERR CALL( v )

  // テーブルポインターが使用されている場合
DIALOG( Table(1)-> ; $formname)

  // DEFAULT TABLE が使用されている場合
DEFAULT TABLE([myTable])
DIALOG( "myForm")
```

```
  // "myForm" はプロジェクトフォームとして評価されます
  // SQL コードから FN を使用した関数呼び出し
Begin SQL
  SELECT Film_Title, {FN How_Many_Actors(ID) AS NUMERIC}
  ...
End SQL
```

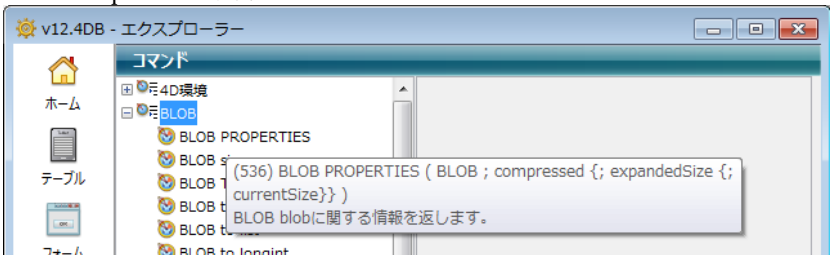
結果ウィンドウからのドラッグ＆ドロップ

デザインモードで、結果ウィンドウ中の項目をドラッグ＆ドロップにより他の 4D アプリケーションにコピーできます。オブジェクトの移動に関する原則 (特に分離できないオブジェクト) は、エクスプローラーからコピーする場合と同じです。詳細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。

4D v13 ではメソッド内で参照されているテーブルは、デフォルトでメソッドとともにコピーされます。結果ウィンドウやエクスプローラーからメソッドをコピーする際に **Shift** キーを押すことでこの動作を無効にできます。

エクスプローラーのヘルプ Tip

エクスプローラーのリストエリアにコマンドや定数ページで名前の上にマウスを重ねると、そのコマンド番号とシンタックスあるいは低数値がヘルプ Tip として表示されるようになりました :



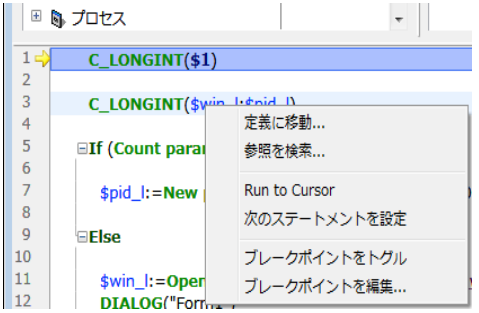
表示されるヘルプ Tip はメソッドエディターの編集エリアに表示されるものと同じです。4D v13 ではリスト上にも表示されるようになりました。

注 以前の 4D ではこの情報がエクスプローラーのプレビューエリアにも表示されていました。4D v13 よりこのプレユーエリアにはコマンド、定数、フィールド、グループに関する情報は表示されなくなりました。

デバッガーのコンテキストメニュー

4D v13 ではデバッガーのインターフェースがメソッドエディタのそれと調和させられました。特に行番号が表示されるようになり、コードブロックの折りたたみや展開が可能になりました。

またトレースモードでのメソッド実行時に利用できる機能にアクセスするための新しいコンテキストメニューが追加されました :



- **定義に移動 ...**: 選択したオブジェクトの定義に移動します。このコマンドは以下のオブジェクトで利用できます :
  - プロジェクトメソッド : 新しいメソッドエディターを開き、選択したメソッドを表示します。
  - フィールド : ストラクチャーウィンドウのインスペクターで、選択したフィールドを表示します。
  - テーブル : ストラクチャーウィンドウのインスペクターで、選択したテーブルの属性を表示します。
  - フォーム : フォームエディターでフォームを表示します。
  - 変数 (ローカル、プロセス、インタープロセス、または $n 引数): カレントメソッドまたはコンパイラーメソッド中で変数が宣言されている箇所を表示します。
- **参照を検索 ...** (メソッドエディターでも利用可能): メソッドのカレント要素を参照しているすべてのデータベースオブジェクト (メソッドとフォーム) を検索します。カレント要素とは選択されているものあるいはカーソルが上にあるもので、フィールド名、変数、コマンド、文字列等です。検索結果は標準の検索結果ウィンドウに表示されます。
- **カーソルまで実行** : プログラムカウンター (黄色の矢印) からメソッド中の選択された行までのコードを実行します。

互換性に関する注意

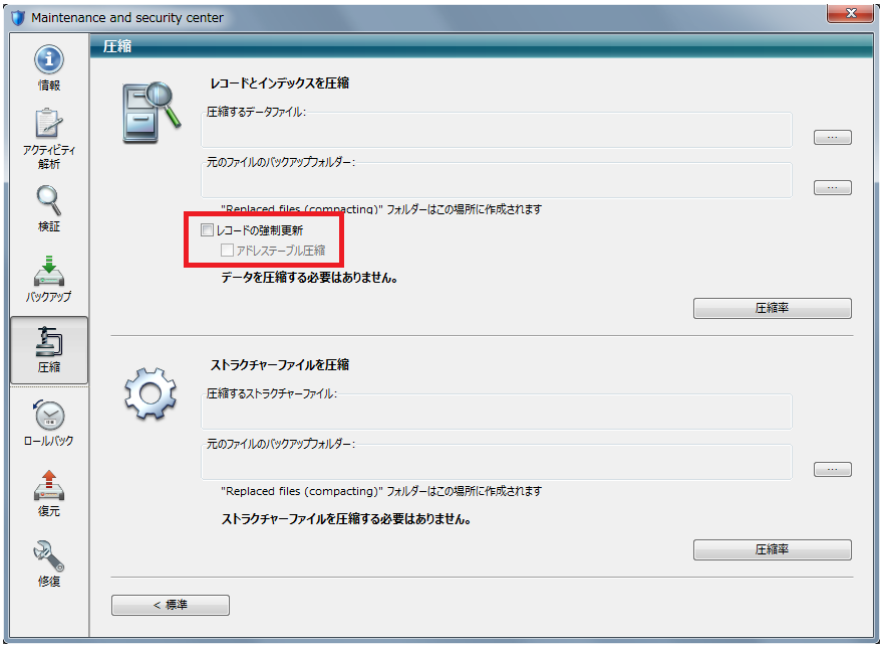
この機能は 4D v13 で取り除かれた、一時的なブレークポイントの機能を置き換えるものです。

- **次のステートメントを設定**: プログラムカウンター (黄色の矢印) を選択された行に移動しますが、途中の行は実行しません。指定された行はユーザーがコードを実行するボタンをクリックした際に実行されます。
- **ブレークポイントをトグル** (メソッドエディターでも利用可能): 選択した行のブレークポイントの入 / 切を切り替えます。この設定によりブレークポイント設定はメソッドに保存されます。つまり、例えばデバッガーでブレークポイントを削除すると、メソッドエディタからも削除されます。
- **ブレークポイントを編集 ...** (メソッドエディターでも利用可能): ブレークポイントプロパティダイアログボックスを表示します。更新された内容はメソッドに保存されます。

Maintenance & security center

新しい圧縮オプション

レコードやインデックスなどのデータ圧縮に関連する 2 つの新しいオプションが**圧縮 / 上級**ページに追加されました。:



これらのオプションはコマンド Compact data file (73 ページ) でも利用できます。

- **レコードを強制更新**: このオプションが選択されている場合、圧縮実行時に 4D は、その時点でのストラクチャー定義を使用して各テーブルのすべ

てのレコードを再保存します。このオプションが選択されていない場合、4D はディスク上のデータの再配置のみを行います。

このオプションは以下のケースで利用します :

- データが保存された後にストラクチャー定義を変更した場合。例えば倍長整数型から実数型に変更したような場合です。4D では (データ欠損の可能性があるものの)、実数とテキスト間などさらにまったく別な型への変換も可能です。
この場合 4D は既存のデータを書き直すことはしません。データはレコードがロードされ (データが再入力されて) 保存されるときに変換されます。このオプションを使用すればすべてのデータを圧縮時に変換することができます。
- データが入力された後にテキストや BLOB、ピクチャーのデータファイル外保存オプションが変更された場合。これは特にデータベースを v13 に変換した後にオプションを変更することによって発生します (21 ページの " データをデータファイル外に保存 " 参照)。先のケースと同様、設定を変更しても 4D は既存のデータをその設定に基づいて自動で書き換えることはしません。既存のデータに対し新しいデータ格納方法設定を適用するために、データの強制更新を使用することができます。
- テーブルやフィールドが削除された場合。この場合レコードを強制的に更新して圧縮することで、削除されたデータ分のスペースを取り除くことができ、ファイルサイズが小さくなります。

注 このオプションが選択されると、すべてのインデックスが更新されます。

- **アドレステーブル圧縮** (先の設定が有効にされたときのみ選択可能): このオプションを選択すると、圧縮時にレコードのアドレステーブルが完全に再構築されます。これによりアドレステーブルのサイズが最適化されます。そしてこれは大量のデータが作成され、削除されたデータベースで使用されます。
このオプションを使用すると圧縮の時間が大幅に増えること、そして SAVE SET コマンドを使用して保存されたセットが無効になる点に留意してください。この場合、保存したセットは削除してしまうことを強く推奨します。無効になったセットを使用すると誤ったセレクションを使用してしまうことになります。

注 圧縮の際にはゴミ箱に入れられたテーブルのレコードも削除されます。対象のレコードが大量にある場合、これも処理を遅くする要因となります。

外部データのサイズ

MSC の " 情報 / データ " ページにはデータファイルの外に保存されたデータのサイズが含まれません (21 ページの " データをデータファイル外に保存 " 参照)。

アプリケーションビルダーの HTML ログ

アプリケーションビルダーを使用するとデータベースの **Logs** フォルダーに、いままでの XML 形式に加え、HTML 形式のログファイルも作成するようになりました。このファイルは BuildApp.log.html という名称です。アプリケーションをビルドするたび、両方のファイルが更新されます。

この HTML ファイルではエラーや警告がグラフィカルに表示されます :

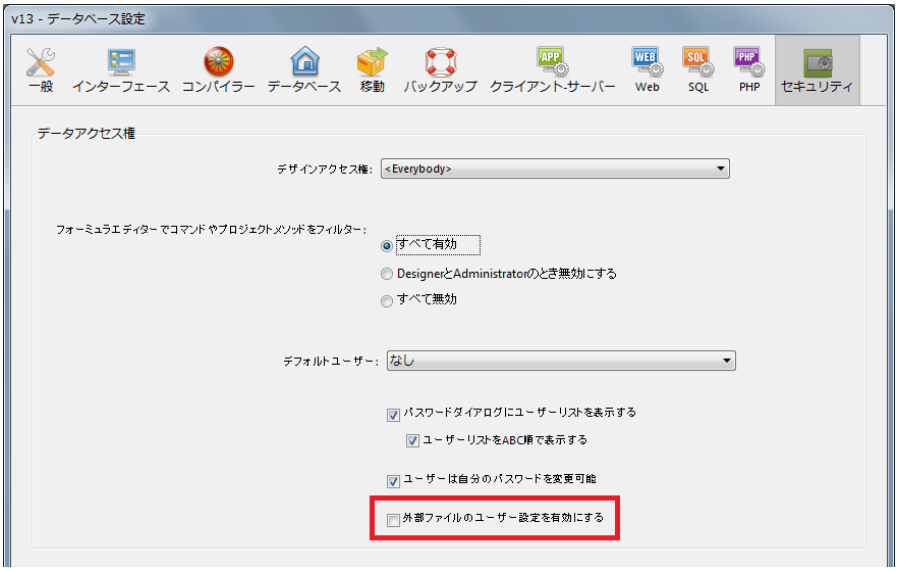
ユーザー設定の外部ファイル化

4D v13 ではデータベースのカスタム設定を外部ファイルとして生成できるようになりました。この機能が有効にされると、ユーザー設定を設定したときに、内容が外部ファイルに書き出されるようになります。そしてデータベースストラクチャーファイルに書き込まれたストラクチャー設定よりもこの設定が優先して使われます。

この結果 4D アプリケーションを更新しても、カスタム設定を保持できるようになります。あるいは異なる場所に展開する同じアプリケーションに対し、異なる設定を適用することが可能になります。またファイルの内容は XML で記述されるため、プログラムで設定を管理することができるようにもなります。

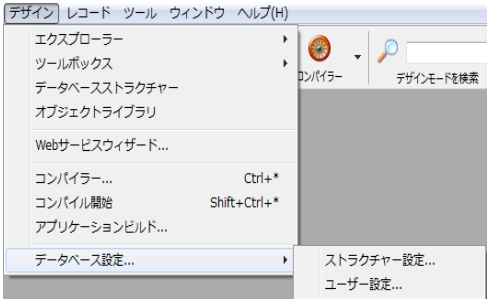
" ユーザー設定 " モードの有効化

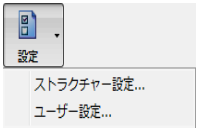
ユーザー設定の外部ファイル化を有効にするには、データベース設定の " セキュリティ " ページ内で**外部ファイルのユーザー設定を有効にする**オプションを選択します :



注 このオプションを有効化するにはデータベースを再起動する必要があります。

このオプションをチェックすると、データベース設定が 2 つのダイアログ、**ストラクチャー設定**と**ユーザー設定**に分けられます。これらのダイアログボックスには**デザイン / データベース設定**メニューあるいはツールバーの**設定**ボタンからアクセスできます :



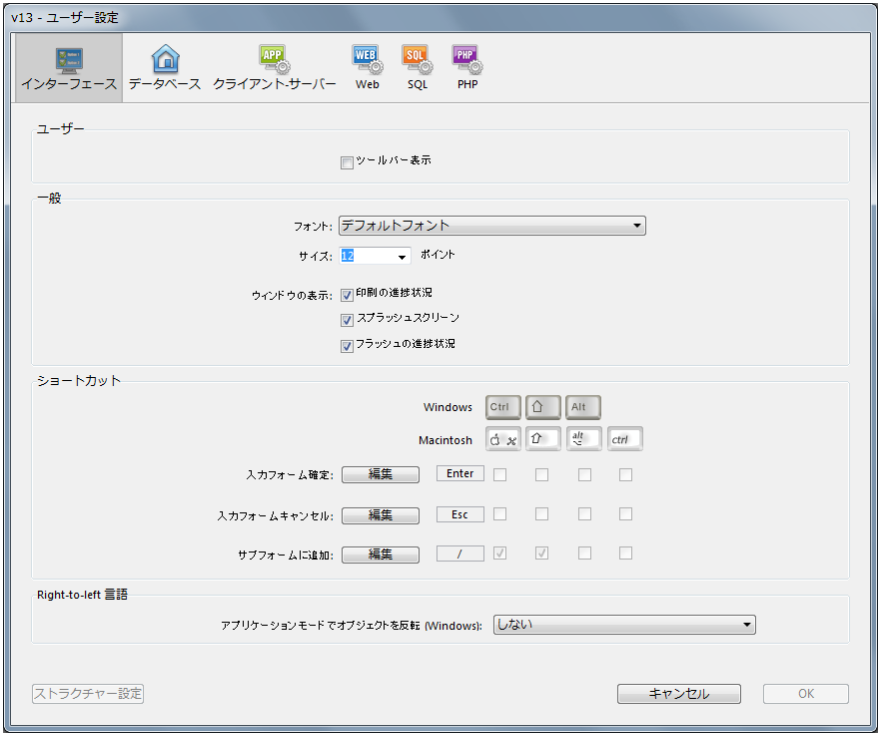


設定を管理するダイアログボックス

外部保存モードが有効にされると、データベース設定は " ユーザー設定 " または " ストラクチャー設定 "、2 つのダイアログボックスから行えるようになります。。

" ストラクチャー設定 " ダイアログボックスは標準のデータベース設定と同じであり、すべてのプロパティにアクセスできます。" ユーザー設定 " ダイアログボックスには外部ファイル化が可能なプロパティに関連するものが含まれます。

ユーザー設定



ユーザー設定が優先される原則に沿うために、このダイアログボックスを使用してオプションが変更されると、ストラクチャー設定に格納された値に代わりそちらが使用されます。

" ユーザー設定 " ダイアログボックスに含まれるページのリストと、各ページでの標準設定との主な違いを以下の表にまとめます :

ページ	ユーザー設定
一般	利用できません
インターフェース	標準設定と同じです
コンパイラー	利用できません
データベース	" メモリ " タブのみ
移動	利用できません
バックアップ	利用できません
クライアント - サーバー	標準設定と同じです
Web	"Web サービス " タブ : メソッドプリフィクスオプションが表示されません
SQL	標準設定と同じです
PHP	標準設定と同じです
セキュリティ	利用できません
互換性	利用できません

注 コマンド OPEN SETTINGS WINDOW (78 ページ) も変更されました。

ユーザー設定ファイル

データベース設定で**外部ファイルのユーザー設定を有効にする** オプションを選択し設定を行うと、ユーザー設定ファイルは以下の場所に作成されます :

[DatabaseFolder]/Preferences/**settings.4DSettings**

ここで [DatabaseFolder] はデータベースストラクチャーファイルが配置されたフォルダーの名前です。

このファイルの内容は XML であり、4D の XML コマンドや XML エディターを使用して読み込んだり変更したりできます。つまり、特に 4D Volume Desktop をマージしたエンジン組み込みアプリケーションにおいて、設定内容をプログラムで管理することが可能です。このファイルをプログラムで更新した場合、設定内容はデータベース再起動後に有効となります。

4 Web サーバー

4D v13 では Web サーバーが完全に書き直されました。最新テクノロジーを統合し、Web の世界での最近の進化が適合されました。

この新しいエンジンは既存の 4D Web アプリケーションと完全に互換性があります。HTML コードとタグ、特別な URL、そして 4D アプリケーションがスタティックおよびダイナミックな Web ページを配信するためのすべてのメカニズムは v13 で同じに動作します。若干の調整が必要になる個所がありますが、その点についてはこの章で説明されています。さらには Web アプリケーションの能力を拡張させる新機能が組み込まれました。

ひとつだけ例外があります。4D v12 で廃止予定とアナウンスされていたコンテキストモードは 4D v13 でサポートが停止されました。アプリケーションでこの機能を使用している場合、例えば新しい Web セッション管理機能を使用した、標準のメカニズムに移行する必要があります。

注 より明示的にするために、"Web サーバー " テーマのコマンドは "WEB" から始まるよう名称が変更されました (197 ページの "Web サーバーテーマコマンドの名称変更 " 参照)。

Web セッション管理

4D v13 の Web サーバーはユーザーセッションを管理するための簡潔で完全なメカニズムを提供します。

説明

自動セッション管理メカニズムを使用すると、ある Web クライアントからの一連のリクエストが同じコンテキスト (セレクションや変数インスタンス) を再利用するようになります。

これは 4D 自身が設定する "4DSID" という名前のプライベート cookie を使用することで行われます。Web クライアントからのリクエストを受信

するたび、4D は 4DSID cookie がリクエストヘッダーに含まれているかどうかをチェックします :

- cookie が含まれている場合、4D は既存のコンテキストの中からこのリクエストに対応するコンテキストを見つけるよう試みます。
  - コンテキストが見つかった場合、それがリクエスト処理のために再利用されます。**Compiler_Web** メソッドは実行されません。
  - コンテキストが見つからない場合、4D は新しいコンテキストを作成します。
- cookie がリクエストヘッダーに含まれていない場合、4D は新しいコンテキストを作成します。

cookie の有効期限とコンテキストの保存

cookie の有効期限はデフォルトで 480 分 (8 時間) です。この有効期限は新しい WEB SET OPTION コマンドを使用して変更できます。cookie の有効期限は Web Inactive session timeout オプション、Web サーバープロセスのタイムアウトは Web Inactive process timeout オプションで個別に設定できます。例えばブラウザー側でショッピングカート用の cookie 有効期限を 24 時間有効にするとしても、最適化の目的で Web プロセスをそんなに長く保持したくないと思うでしょう。この場合、Web プロセスのタイムアウトをもっと短く、例えば 4 時間に設定できます。Web プロセスがタイムアウトするときは On Web Session Suspend データベースメソッドが呼び出され、プロセスが破棄される前に、セレクションや変数の値などを後で利用するために退避させることができます。次回 Web クライアントが (24 時間以内に) 接続してきたときは、同じ cookie がサーバーに送信されるので、その cookie 値をキーとして前回のセッションのコンテキスト (変数値やセレクションの状態など) を新しく作成された Web プロセス内に読み込むことができます。

非動作コンテキストの破棄

4D はセッションを管理する Web プロセス数が上限値に達した場合、もっとも古いプロセスを破棄します。この上限値はデフォルトで 100 であり、新しい WEB SET OPTION コマンドで変更できます。
プロセスが破棄される直前には新しい On Web Session Suspend データベースメソッドが呼び出されるので、変数やセレクションなどを次回のリクエストで再利用できるよう保存することができます。

例題

この例題では On Web Connection データベースメソッドや On Web Session Suspend データベースメソッドを使用してセッションを管理する方法を示します。

- On Web Connection のコード :

```
// On Web Connection ( または On Web Authentication)
```

```
C_TEXT (www_SessionID)
If (www_SessionID = WEB Get current session id)
    // Compiler_Web は呼び出されない
    // すべての変数やセレクションは既に存在している
  ...
Else
    // Compiler_Web が実行された
    // これは新しいセッションで、変数やセレクションは存在しない
    // 4D が作成したこのセッションの ID をプロセス変数に格納する
  www_SessionID:= WEB Get current session id

    // Web セッションの初期化を行う
    // セレクションを作成
    // ユーザーを検索
    // www_Login には認証済みログインユーザー名が入っている
  QUERY ([User];[User]Login = www_Login)
  QUERY ([prefs];[prefs]Login = www_Login)

    // 従業員情報を検索
  QUERY ([employees];[employees]Name=[user]name)
  QUERY ([company];[company]Name=[user]company)

    // 変数のセットアップ
    // このユーザーの環境設定をロード
  SELECTION TO ARRAY([prefs]name;prefNames;[prefs]values;prefValues)
  www_UserName:=[User]Name
  www_UserMail:=[User]mail

// セッションが初期化された
End if
```

- On Web Session Suspend データベースメソッドのコード :

```
  // On Web Session Suspend
  // タイムアウトその他の理由で 4D はセッションを閉じることがある
C_LONGINT(www_SessionID)
www_SessionID:=""
  // セッション情報を永続化する
  // このプロセスを使用して接続していたユーザーの環境設定を保存
QUERY ([prefs];[prefs]Login = www_Login) // 変数値はまだ使用できる
ARRAY TO SELECTION(prefNames;[prefs]name;prefValues;[prefs]values)

  // 重要 : 4D はプロセスを終了します
  // 変数やセレクションなどは破棄されます
```

メカニズムを有効 / 無効にする

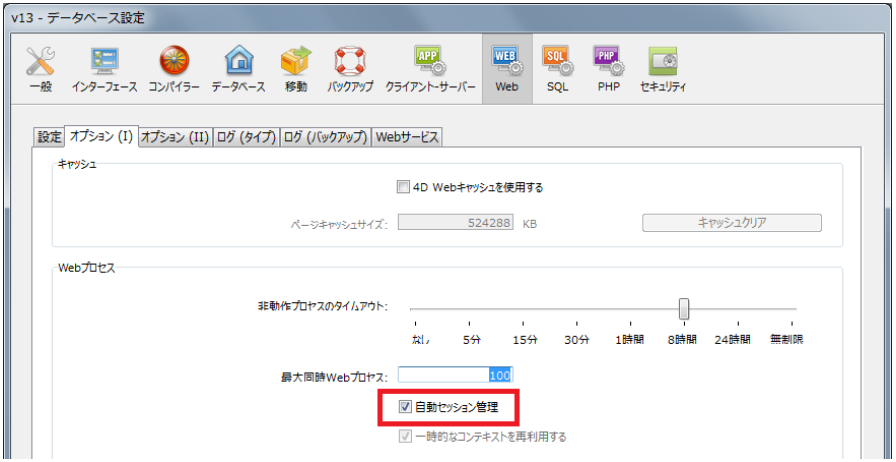
アプリケーションで 4D の自動 Web セッション管理を使用するには、その機能が有効化されていなければなりません。

4D v13 以降で作成されたデータベースでは、このメカニズムがデフォルトで有効になっています。

4D v12 以前から変換されたデータベースの場合、互換性のためこの機能は無効になっています。この新機能を使用するためには明示的に有効化しなければなりません。

自動セッション管理を有効化する方法は 2 つあります :

- データベース設定の Web ページ、" オプション (I)" の**自動セッション管理**を使用する :



これを使用した場合、設定はディスクに保存され永続化されます。

- コマンド WEB SET OPTION (195 ページ) で Web Keep session オプションを使用する。この場合、設定は 4D を終了するまで有効で、データベース設定の指定を上書きします。

いずれの場合も設定はローカルマシンに有効です。つまり 4D Server とリモートモードの 4D では異なる設定を行うことができます。

" 一時的なコンテキストを再利用する (リモートモード)" オプション

新しい**自動セッション管理**オプションを選択すると、既存の**一時的なコンテキストを再利用する (リモートモード)** オプションも自動で選択され、ロックされます。セッション管理メカニズムは Web プロセスをリサイクルする原理に基づいています。各セッションは、セッションの寿命が切れるまで保持される、同じ 4D Web プロセスを使用します。あるセッションプロセスが他のセッションで使用されることはありません。

セッションが終了するとプロセスは自動で終了され、再利用されません
ます。

以前のバージョンと同様、このオプションはリモートモードの 4D Web サーバーで有効です。ローカルモードの 4D では (セッションプロセスを除く) すべての Web プロセスが、使用後に終了されます。

COOKIE が拒否されている場合

4D の自動セッション管理メカニズムは cookie を利用しています。Web クライアントが cookie をサポートしない、あるいは拒否している場合、4D HTTP サーバーはセッションを保守できません。この場合各リクエストは新規接続として扱われ、各接続毎に Compiler_Web メソッドが実行されます。

セッションと IP アドレス

4D HTTP サーバーはセッションを開始した IP アドレスを記憶します。異なる IP アドレスから既存のセッションにアクセスされた場合、HTTP 400 エラーがクライアントに返されます。

新しいタグ

4D v13 Web サーバーでは 2 つの新しい 4D HTML タグ、**4DELSEIF** と **4DBASE** を使用できるようになりました。

4DELSEIF

4DELSEIF タグを使用して Case of タイプの条件判定構造を記述できます。

シンタックス : <!--#4DIF (条件式 1)--> <!--#4DELSEIF (条件式 2)-->...<!--#4DELSEIF (条件式 N)--> {<!--#4DELSE-->} <!--#4DENDIF -->

このタグを使用して複数の条件式をテストできます。最初に **True** と判定されたブロックの中に記述されたコードのみが実行されます。真となる条件式が見つからない場合、4DELSE があればそのブロックが実行され、なければ何も実行されません。4DELSE タグは最後の 4DELSEIF の後に記述できます。

4D v12 以前で以下のように記述していたコードは :

```
<!--#4DIF Condition1-->
 /* Condition1 が真 */
<!--#4DELSE-->
<!--#4DIF Condition2-->
    /* Condition2 が真 */
  <!--#4DELSE-->
     <!--#4DIF Condition3-->
        /* Condition3 が真 */
```

```
      <!--#4DELSE-->
         /* 真となる条件がない */
      <!--#4DENDIF-->
   <!--#4DENDIF-->
<!--#4DENDIF-->
```

4DELSEIF タグを使用して以下のように記述できます :

```
<!--#4DIF Condition1-->
   /* Condition1 が真 */
<!--#4DELSEIF Condition2-->
   /* Condition2 が真 */
<!--#4DELSEIF Condition3-->
  /* Condition3 が真 */
<!--#4DELSE-->
  /* 真となる条件がない */
<!--#4DENDIF-->
```

- 接続ユーザーに基づき異なるページコンテンツを返す例 :

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

4DBASE

新しい **4DBASE** タグを使用して 4DINCLUDE タグを評価する際に使用されるワーキングディレクトリを指定できます。

シンタックス : <!--#4DBASE folderPath-->

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。
WEBFOLDER キーワードを渡すと、デフォルトパスに戻されます (つまりそのページに対して相対)。

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。

4D v12 では各 4DINCLUDE 呼び出しごとに相対パスを指定し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-->
```

4D v13 では **4DBASE** タグを使用して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-->
```

- 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 ファイル中ではさらに 4DBASE を使用してカレントフォルダーが変更されます。ただし index.html のコンテキストでは変更されません :

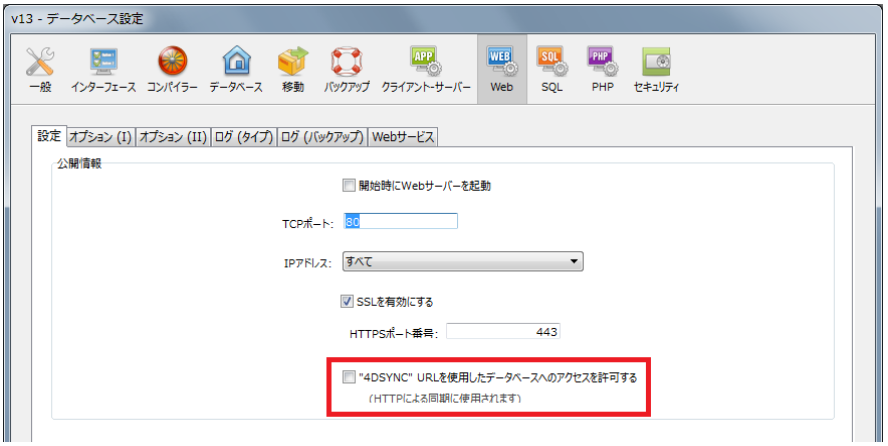
```
/* head.html */
/* ここでのワーキングディレクトリは組み込みファイルからの相対 (FR/ または US/) */
<!--#4DBASE Styles/-->
<!--#4DINCLUDE main.css-->
<!--#4DINCLUDE product.css-->
<!--#4DBASE Scripts/-->
<!--#4DINCLUDE main.js-->
<!--#4DINCLUDE product.js-->
```

特別な URL の管理

/4DSYNC リクエストサポートオプション

4D v13 では新しいオプションを使用して /4DSYNC URL リクエストをサポートするかどうか制御できるようになりました。この URL は HTTP を使用してデータを同期するために使用されます (この URL に関する詳細は 4D ランゲージリファレンスの http://doc.4d.com/4Dv13/help/Title/ja/page1204.html を参照してください)。

"4DSYNC" URL を使用したデータベースへのアクセスを許可するという名称の新しいオプションがデータベース設定の "Web/ 設定 " ページに追加されました :



この新しいオプションを使用して /4DSYNC を含むリクエストの処理を有効 / 無効にできます :

- 選択されていない場合、/4DSYNC リクエストは標準のリクエストとみなされ、特別な処理は行われません。同期リクエストを行うと "404 - リソースを利用できません " タイプのレスポンスが送信されます。
- 選択されている場合、同期メカニズムが有効となります。/4DSYNC リクエストは特別なリクエストとして処理され、4D HTTP サーバーは特別な解析を行います。

4D v13 で作成されたデータベースはデフォルトでこのオプションが無効になっています。

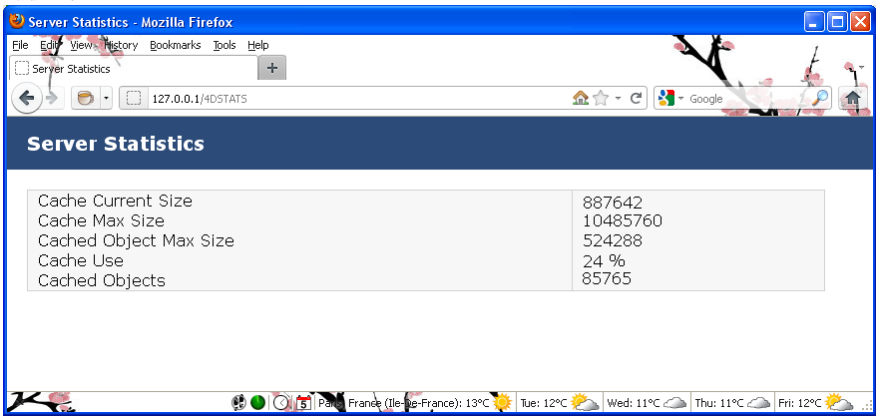
しかし互換性保持のため、以前のバージョンから変換されたデータベースではこのオプションが選択されていることがあります。アプリケーションで HTTP による同期のメカニズムを使用しない場合、このオプションの選択を解除することを推奨します。

注 以前のバージョンでは /4DSYNC URL を **On Web Authentication** データベースメソッドでフィルターすることが推奨されていました。データベース設定でこの機能を無効にしている場合、このリクエストを拒絶する目的で URL をフィルターする必要はなくなりました。

このオプションのスコープはアプリケーションに対しローカルであり、設定を有効にするには Web サーバーを再起動しなければなりません。

/4DSTATS と /4DHTMLSTATS

4D v13 では /4DSTATS と /4DHTMLSTATS URL のレスポンスが HTML の表形式で返されるようになりました :



/4DSTATS

/4DSTATS URL から返される情報は以下の通りです :

- **Cache Current Size:** 現在のキャッシュのサイズ (バイト)
- **Cache Max Size**: キャッシュの最大サイズ (バイト)
- **Cached Object Max Size**: キャッシュに含まれる一番大きなオブジェクトのサイズ (バイト)
- **Cache Use**: キャッシュの使用率
- **Cached Objects**: キャッシュ中のオブジェクト数 (ページ、ピクチャー等)

/4DHTMLSTATS

以前のバージョンの 4D と同様 /4DHTMLSTATS URL は /4DSTATS と同じ情報を返しますが、**Cached Objects** フィールドには (ピクチャーファイル等を含まない) HTML ページの数のみが返されます。さらにこの URL を使用した場合、追加で **Filtered Objects** フィールドが返されるようになりました。

- **Filtered Objects**: ピクチャーなど、この URL 呼び出しで **Cached Objects** にカウントされなかったオブジェクトの数。

最適化

4D v13 で行われた複数の最適化の恩恵を受け、4D Web アプリケーションの実行も速度が速くなりました。通常この最適化のためにコードを変更しなければならないということはありません。しかし動作が異なる点があります。

4D Server 上でのクライアントプロセス作成のタイミング

以前のバージョンの 4D では、リモートモードの 4D で (名前が "$" から始まらない) グローバルプロセスが作成されると、4D Server 上でも即座に 2 つのプロセスが作成されていました :

- ランゲージアクセスを処理するコオペラティブプロセス
- データベースエンジンアクセスを処理するプリエンプティブプロセス

4D v13 より、これら 2 つのプロセスはクライアント側のコードを実行する際、**必要になったときに**作成されるようになりました。これによりリモートモードの 4D Web サーバーへの HTTP 接続が最適化されます。

新しいプロセス処理では、クライアントコードの実行に伴いサーバープロセスは以下のように作成されます :

```
  // グローバルプロセスが開始されるが、
  // サーバーにはプロセスが作成されない
CREATE RECORD([Table_1])
[Table_1])field1_1:="Hello world"
SAVE RECORD([Table_1])
  // ここでサーバー上にプリエンプティブプロセスが作成される
$serverTime:=Current time(*)
  // ここでサーバー上にコオペラティブプロセスが作成される
```

以下の動作に留意してください :

- On Server Open Connection データベースメソッドは、サーバー上にコオペラティブプロセスが作成されたときに実行されます。このことから、4D v13 以降このデータベースメソッドは、クライアント側のコード実行途中に呼び出されるかもしれません。コオペラティブプロセスが作成されない場合は呼び出されません。またこのデータベースメソッドの呼び出しはプリエンプティブメソッドの作成と関係なく行われます。
- リモートの 4D が接続する際に作成される最初のプロセスなど、メインのプロセスは影響を受けません。対応する 2 つのプロセスがサーバー側に即座に作成されます。

Gzip 圧縮

4D v13 Web サーバーは Web サーバーとクライアント間で " ネゴシエーション " が行われた後の gzip 圧縮をサポートするようになりました。

圧縮が使用される為には以下の条件を満たす必要があります :

- Web クライアントが個の圧縮をサポートする ("Accept-Encoding: gzip" ヘッダーを送信してきている)。
- コンテンツがテキストタイプ。ピクチャーやバイナリデータは圧縮されません。しかし COMPRESS BLOB コマンドで GZIP 圧縮が可能になった点に留意してください。これにより WEB SEND RAW DATA コマンドを使用して圧縮データを送信できます。
- データが 1024 バイトを超えること。このサイズ以下の場合、圧縮の効果はありません。

新しいコマンド WEB SET OPTION (195 ページ) を使用して Web 圧縮の敷居値やレベルを設定できます。

5 ランゲージ

この章では 4D v13 における新しいランゲージコマンドや変更点について説明します。

4D 環境

CREATE DATA FILE

CREATE DATA FILE (accessPath)

4D Server: CREATE DATA FILE コマンドを 4D Server で実行できるようになりました。この場合、4D Server は指定されたファイルを作成する前に、内部的に QUIT 4D を実行します。結果各リモートマシンにはサーバーが終了しようとしている旨のダイアログボックスが表示されます。

参照 : OPEN DATA FILE

Compact data file

Compact data file (structurePath ; dataPath {; archiveFolder{; options{; method}}}) → テキスト

省略可能な option 引数に、"Data file maintenance" テーマの新しい 2 つの引数を渡せるようになりました :

定数	型	値	説明
Compact address table	倍長整数	131072	強制的にレコードのアドレステーブルを更新します (圧縮時間は長くなります)。このオプションのみを指定すると、4D は自動でレコードの再保存を行います。
Update records	倍長整数	65536	現在のストラクチャー定義に基づき、すべてのレコードを強制的に再保存します。

これらのオプションに関する詳細は 54 ページの "Maintenance & security center" を参照してください。

Get 4D folder

Get 4D folder ({folder}{;}{*}) → 文字

Mac OS において、このコマンドに Active 4D Folder 定数を渡した場合、あるいは引数を省略した場合に返されるフォルダーの場所が、4D v13 で変更されました。

新しい場所は以下の通りです :
{Disk}:Users:User:Library:Application Support:4D

互換性に関する注意

以前のバージョンの 4D では、Mac OS X で 4D フォルダーは以下の場所に作成されていました : [...]Library:Preferences:4D
以前のバージョンからデータベースを変換し、このデータベースが 4D フォルダーの中に配置したカスタムデータにアクセスしていた場合、その内容を手作業あるいはプログラムを使用して新しい場所に移動する必要があります。

Get database parameter, SET DATABASE PARAMETER

SET DATABASE PARAMETER ({aTable; }selector; value)

Get database parameter ({aTable; }selector{; stringValue}) → 実数

Get database parameter, SET DATABASE PARAMETER コマンドは Windows 版の 4D で Direct2D グラフィック描画エンジンの実装を設定するための新しい定数を受け入れます。
Debug Log Recording セレクター用の新しい設定値と 2 つの新しいセレクター (Diagnostic Log Recording と Log Command list) はアプリケーションを開発するためのツールとしての完成度をさらに高めるものです。
なお Web サーバーを管理するためのセレクターは廃止予定となりました。

Direct2D の管理

この Direct2D グラフィック描画エンジンは Windows Vista 以降でサポートされています。

注 Direct2D に関する詳細は 45 ページの "Windows での新しい Direct2D グラフィック描画エンジン " を参照してください。

- **Selector= 69** (Direct2D Status)
  - 説明 : Windows における Direct2D 実装のアクティベーションモード
  - 値: "Database Parameters" テーマの以下の定数のいずれか (デフォルトでモード 2):

• Direct2D Disabled (0): Direct2D モードは無効であり、データベースは過去のモード (GDI/GDIPlus) で動作する。
• Direct2D Hardware (1): 4D アプリケーション全体で Direct2D グラフィックハードウェアコンテキストを使用する。このコンテキストが利用できない場合、Direct2D グラフィックソフトウェアコンテキストを使用 (Vista を除く。Vista ではパフォーマンスのために GDI/GDIPlus モードが使用されます)。
• Direct2D Hardware SVG and Editors (2) (デフォルトモード): SVG、コード、およびストラクチャーエディターで Direct2D グラフィックハードウェアコンテキストを使用する。このコンテキストが利用できない場合、Direct2D グラフィックソフトウェアコンテキストを使用 (Vista を除く。Vista ではパフォーマンスのために GDI/GDIPlus モードが使用されます)。
• Direct2D Software (3): Windows 7 以降、4D アプリケーション全体で Direct2D グラフィックソフトウェアコンテキストを利用。Vista ではパフォーマンスのために GDI/GDIPlus モードが使用されます。
• Direct2D Software SVG and Editors (4): Windows 7 以降、SVG、コード、およびストラクチャーエディターの Direct2D グラフィックソフトウェアコンテキストを使用する。Vista ではパフォーマンスのために GDI/GDI/GDIPlus モードが使用されます。
• Direct2D Mixed (5): Windows 7 以降、SVG、コード、およびストラクチャーエディターで Direct2D グラフィックハードウェアコンテキストを使用する。4D アプリケーションの他の部分では Direct2D グラフィックソフトウェアコンテキストを使用する。Vista ではパフォーマンスのため GDI/GDIPlus モードを使用します。

- **Selector = 74** (Direct2D Get active status)
  注 : このセレクターは Get database parameter コマンドでのみ使用することができ、設定することはできません。
  - 説明 : Windows において Direct2D のアクティブな実装を返します。
  - 値 : 0, 1, 2, 3, 4 または 5 (セレクター 69 の値参照)。返される値は Direct2D が利用可能かどうか、およびハードウェア、OS によってサポートされる Direc2D の品質に基づきます。
    例えば :
    SET DATABASE PARAMETER(Direct2D Status ; Direct2D Hardware)
    $mode:=**Get database parameter**(Direct2D Get active status)
    - Windows 7 以降、システムが Direct2D 互換のハードウェアを検知すると、$mode に 1 が設定されます。そうでなければ $mode は 3 に設定されます (ソフトウェアコンテキスト)。
    - Windows Vista では、システムが Direct2D 互換のハードウェアを検知すると、$mode に 1 が設定されます。そうでなければ $mode は 0

に設定されます (Direct2D 無効)。
- Windows XP では、$mode は常に 0 です (Direct2D 非互換)。

デバッグ

- **Selector = 34** (Debug Log Recording)
  このセレクターは実行時間を記録する詳細モードを指定する新しい値 4 を受け入れます。
  このモードが有効にされると、デバッグログファイルに追加の情報項目、ミリ秒単位の処理実行時間が記録されます。例えば :
  4072 p=5 puid=16 (2) cmd: DELAY PROCESS(5;60). 1046 ms
  5335 p=5 puid=17 end_meth: Method2 5335 ms

  "< ms" はコマンドが 1 ミリ秒以下で実行されたことを示します。
  ファイルにあまりにも大量の情報が書き込まれることを避けるために、記録する 4D コマンドを Log Command list (selector = 80) で制限できます (後述)。
- **Selector = 79** (Diagnostic Log Recording)
  - 説明 : 4D 診断ファイルの記録を開始または停止する。デフォルトの値は 0 で記録を行いません。
    4D は診断ファイルの中に内部的なアプリケーション処理に関連するイベントを継続的に記録することができます。このファイルに含まれる情報は 4D アプリケーション開発のために使用され、4D 社の技術サポート担当により解析されます。このセレクターに 1 を渡すと、DatabaseName.txt という名称のファイルが自動でデータベースの **Logs** フォルダーに作成されるか、既に存在する場合は開かれます。このファイルのサイズが 10MB に達するとそのファイルは閉じられ、DatabaseNameN.txt が生成されます (N は連番)。
    LOG EVENT コマンドを使用してカスタム情報をこのファイルに書き込むこともできます。
  - 値 : 0 または 1 (0 = 記録しない , 1 = 記録する)
- **Selector = 80** (Log Command list)
  - 説明 : デバッグファイルに記録する 4D コマンドのリスト (セレクター 34, Debug Log Recording 参照)。デフォルトではすべての 4D コマンドが記録されます。
    このセレクターを使用すれば、記録に残したい 4D コマンドを指定することで、デバッグファイルに保存される情報の量を制限することができます。
  - 値 : 記録する 4D コマンドの番号リスト。型は文字列で各コマンド番号をセミコロンで区切ります。"all" を渡すとすべてのコマンドが記録され、"" (空の文字列) を渡すとなにも記録されません。

- 例 :

SET DATABASE PARAMETER(Log Command list;"277;341")
// QUERY と QUERY SELECTION コマンドのみを記録

SET DATABASE PARAMETER (Log Command list;"")
// どの 4D コマンドも記録しない

Web 関連セレクターの移動

Web サーバー管理に関連するセレクターは廃止予定となり、現在は互換性の目的で保持されています。4D v13 以降は Web サーバーに関係するオプションはすべて新しい WEB SET OPTION と WEB GET OPTION コマンドを使用して管理しなければなりません。影響を受けるセレクターは以下の通りです :

セレクター	値	WEB SET OPTION / WEB GET OPTION コマンドの新しいセレクター
Character set	17	Web Character set
HTTP Compression Level	50	Web HTTP Compression Level
HTTP Compression Threshold	51	Web HTTP Compression Threshold
HTTPS Port ID	39	Web Port ID
IP Address to listen	16	Web IP Address to listen
Max Concurrent Web Processes	18	Web Max Concurrent Processes
Maximum Web requests size	27	Web Maximum requests size
Web Log Recording	29	Web Log Recording

これらのセレクターの動作は変更されていません。WEB SET OPTION で設定された新しいオプションは常にローカルの Web サーバーに適用され、4D を終了するまで有効であり、保存はされない点に留意してください。

GET MEMORY STATISTICS

GET MEMORY STATISTICS は 4D v13 における GET CACHE STATISTICS の新しい名称です。このコマンドが返す情報に一致するよう、コマンド名称が変更されました。動作に変更はありません。

OPEN DATA FILE

OPEN DATA FILE (accessPath)

4D Server: OPEN DATA FILE コマンドを 4D Server で実行できるようになりました。この場合、4D Server は指定されたファイルを開く前に、内部的に QUIT 4D を実行します。結果各リモートマシンにはサーバーが終了しようとしている旨のダイアログボックスが表示されます。

参照 : CREATE DATA FILE

OPEN SETTINGS WINDOW

OPEN SETTINGS WINDOW (selector {; access} {; settingsType})

引数	型		説明
selector	文字	→	環境設定やデータベース設定ダイアログボックス中のテーマや ページ、またはパラメーターグループを 指定するキー
access	ブール	→	True= ダイアログボックスの他のページをロックする、False または省略 = ダイアログの他のページもアクティブにする
settingsType	倍長整数	→	0 または省略時 = ストラクチャー設定、1 = ユーザー設定

注 このコマンドの名称は前一のバージョンで OPEN 4D PREFERENCES でした。

OPEN SETTINGS WINDOW コマンドは省略可能な追加の引数、settingsType を受け入れるようになりました。

この引数は " ユーザー設定 " モードに設定されたデータベースでのみ使用されます (56 ページの " ユーザー設定の外部ファイル化 " 参照)。このモードでこの引数を使用すると、" ストラクチャー設定 " あるいは " ユーザー設定 " どちらのダイアログボックスにアクセスするのかを指定できます。"4D Environment" テーマの新しい以下の定数を利用できます :

定数 (値)	説明
Structure Settings (0)	ストラクチャー設定を使用 (引数が省略された際のデフォルト)。このモードでは使用される selector の値は標準モードと同じです。
User Settings (1)	ユーザー設定を使用。このモードでは特定のキーのみを selector 引数で使用できます。

" ユーザー設定 " モードでは selector に以下のキーのいずれかのみを渡さなくてはなりません :

/Database

/Database/Interface
/Database/Database/Memory and cpu
/Database/Client-Server
/Database/Client-Server/Network
/Database/Client-Server/IP configuration
/Database/Web
/Database/Web/Config
/Database/Web/Options 1
/Database/Web/Options 2
/Database/Web/Log format
/Database/Web/Log scheduler
/Database/Web/Webservices
/Database/SQL
/Database/php

配列

ARRAY TO SELECTION

ARRAY TO SELECTION{(array ;aField{; arrayN; aFieldN}
{; *})}

引数	型		説明
array	配列	→	コピー元の配列
aField	フィールド	←	配列データを受け取るフィールド
*	演算子	→	実行をスタックする

ARRAY TO SELECTION コマンドは最後の引数として * を受け入れるようになりました。* 引数を渡すと、4D はその式を即座には実行せず、代わりにメモリに格納します。結果 * で終わる行はスタックされます。実行待ちのすべての行は * 引数を渡さない最後の ARRAY TO SELECTION 式により実行されます。この目的のため、このコマンドを引数なしで呼び出すことができるようになりました。

QUERY コマンドと同様、この書式により複雑で長い行をシンプルに記述できるようになり、可読性と保守性が高まります。また途中に他の式を記述することもできます。

- 以下の例ではレコードセレクションを複製します :

```
ARRAY TEXT(a1;0)
ARRAY TEXT(a2;0)
ARRAY TEXT(a3;0)
```

```
ARRAY TEXT(a4;0)

ALL RECORDS([Table_1])
For ($i;1;Get last field number(1))
  $p:=Get pointer("a"+String($i))
  SELECTION TO ARRAY(Field(1;$i)->;$p->;*)
    // まだ配列は作成されない
End for
SELECTION TO ARRAY // スタックした配列作成がここで実行される

REDUCE SELECTION([Table_1];0)
For ($i;1;Get last field number(1))
  $p:=Get pointer("a"+String($i))
   ARRAY TO SELECTION($p->;Field(1;$i)->;*)
     // セレクション構築
End for
ARRAY TO SELECTION // 式の実行
```

参照 : SELECTION TO ARRAY

SELECTION RANGE TO ARRAY

SELECTION RANGE TO ARRAY (start; end{; aField | aTable ; array} {; aField2 ; array2 ; ... ; aFieldN ; arrayN})

SELECTION RANGE TO ARRAY コマンドを start と end 引数だけで呼び出せるようになりました。この特別なシンタックスにより、新しい * 引数を使用してスタックした SELECTION TO ARRAY コマンドの実行を、指定したセレクションに対してのみ適用できます。

- [Invoices] テーブルの最初の 50 レコードのみを配列にロードします :

```
  // 式のスタック
SELECTION TO ARRAY([Invoices]InvoiceRef;arrLInvRef;*)
SELECTION TO ARRAY([Invoices]Date;arrDInvDate;*)
SELECTION TO ARRAY([Clients]ClientRef;arrLClientRef;*)
  // スタックした式を実行
SELECTION RANGE TO ARRAY(1;50)
```

参照 : SELECTION TO ARRAY

SELECTION TO ARRAY

SELECTION TO ARRAY {(aField | aTable ; array {; aField2 ; array2 ; ... ; aFieldN ; arrayN} {; *})}

引数	型	説明

aField | aTable	フィールド , テーブル	→	データを取得するフィールドまたは レコード番号を取得するテーブル
array	配列	←	フィールド値またはレコード番号を受け取る配列
aField	フィールド	→	配列に値を取得するフィールド
array	配列	←	フィールドデータを受け取る配列
*	演算子	→	実行をスタックする

SELECTION TO ARRAY コマンドは最後の引数として * を受け入れるようになりました。* 引数を渡すと、4D はその式を即座には実行せず、代わりにメモリに格納します。結果 * で終わる行はスタックされます。実行待ちのすべての行は * 引数を渡さない最後の SELECTION TO ARRAY 式により実行されます。この目的のため、このコマンドを引数なしで呼び出すことができるようになりました。

配列の型は * 引数が渡されない最後の行が実行されるときに検証されます。

QUERY コマンドと同様、この書式により複雑で長い行をシンプルに記述できるようになり、可読性と保守性が高まります。また途中に他の式を記述したり、ループの中で配列を構築したりすることもできます。

- 4D v12 までのコード :

SELECTION TO ARRAY
([Clients]Name;arrClients;[Clients]ID;arrID;[Company]Name;arrComp)

4D v13 では以下のように記述できます :

SELECTION TO ARRAY ([Clients]Name;arrClients;*)
SELECTION TO ARRAY ([[Clients]ID;arrID;*)
SELECTION TO ARRAY ([Company]Name;arrComp)

参照 : ARRAY TO SELECTION, SELECTION RANGE TO ARRAY

TEXT TO ARRAY

TEXT TO ARRAY(varText; arrText; width; fontName; fontSize{; fontStyle{; *}})

引数	型		説明
varText	テキスト	→	分割する元のテキスト
arrText	テキスト配列	←	単語または行に分割されたテキストを受け取る配列

width	倍長整数	→	文字列の最大幅 (ピクセル)
fontName	テキスト	→	フォント名
fontSize	倍長整数	→	フォントサイズ
fontStyle	倍長整数	→	フォントスタイル
*	演算子	→	指定時、テキストをマルチスタイルとして解釈する

新しい TEXT TO ARRAY コマンドはテキスト変数をテキスト配列変換します。元の varText テキストは (スタイル付きであってもなくても) 分割され、arrText 配列の要素となり、コマンドから返されます。このコマンドを使用して例えば適切な位置でテキストが分割されたメール本文を作成することができます。

テキストの分割はコマンドに渡される行幅やテキスト属性に基づいて計算されます。

varText 引数には配列要素に分解する元のテキストを渡します。このテキストはマルチスタイルであってもなくてもかまいません。マルチスタイルの場合、他のいくつかの引数は無視されます。

arrText 引数にはテキスト配列を渡します。この配列には分割されたテキストを要素とするテキスト配列が返されます。

width 引数にはテキストを分割する際のサイズ計算に用いる行の幅をピクセル単位で返します。テキスト全体に対しテキストの描画属性 (フォント名やサイズ、スタイル) を考慮に入れたうえで、コマンドはこのサイズに収まる単語の切れ目を計算します。

- マルチスタイルテキストの場合、元のテキストのスタイルが計算に使用され、以降の引数はコマンドに渡されても無視されます。この場合結果配列に格納されるテキストは元のスタイルを保持します。
- スタイルなしのテキストの場合、コマンドが行の長さを決定できるようにするため、すべての引数を渡さなければなりません。

各配列要素には最低一つの単語が含まれます。width に渡された値が分割ルールに対し小さすぎる場合、配列は渡された引数に基づいて可能な限り計算されて作成され、OK 変数に 0 が返されます。例えば 3 ピクセル幅を渡した場合、ほとんどの単語はこの長さより大きくなるでしょう。このようなケースでは OK 変数が 0 となります (4D が単語の切れ目を認識できないような場合には、最大幅を超えた要素が作成されることもある点に留意してください)。

またこれは配列の最大際要素数は論理的に varText の単語数になることを意味します。

fontName と fontSize 引数にはそれぞれ、varText の分割計算の際に使用するフォント名とサイズを渡します。これらの引数はスタイルなしテキストの場合に必須です。

fontStyle 引数には Font Styles テーマの定数を 1 つ以上渡せます。この引数は省略可能です。省略された場合、Normal スタイルが使用されます。

オプションの * 引数を指定すると、マルチスタイルテキストの場合で、元のテキストにテキスト属性が指定されていない場合、fontName, fontSize そして fontStyle 引数に渡した値を使用するよう指示することができます。しかしこれらの属性が元のテキスト内で定義されていた場合は無視されます。

- スタイル付きテキストを 200 ピクセル幅で分割します :

```
TEXT TO ARRAY(theText;TextArray;200;"Arial";20;Normal;*)
  // Arial, 20, そして Normal 属性はマルチスタイルテキスト中で
  // これらの属性が指定されていない場合にのみ使用されます
```

- スタイルなしテキストを 350 ピクセル幅、Bodoni Bold フォント、サイズ 14 で分割します。フォントが利用できない場合コマンドは正しく動作しないので、フォントの存在をチェックしなければなりません :

```
ARRAY TEXT($FontList;0)
FONT LIST ($FontList)
$Font:="Bodoni"
$p:=Find in array($FontList; $Font)
If($p>0)
  TEXT TO ARRAY(theText;TextArray;350;"Bodoni";14;Bold)
Else
  // 他のフォントを使用する
End if
```

- スタイル付きテキストをスタイルなしの Arial Normal フォント、サイズ 12、最大幅 600 ピクセルで印刷したいとします :

```
  // マルチスタイルテキストをスタイルなしテキストに変換します
$RawText:=OBJECT Get plain text (vText)
  // 配列を作成します
TEXT TO ARRAY($RawText;TextArray;600;"Arial";12)
```

- 400 ピクセル幅のエリアを印刷します。テキストは最大 80 行で可能な限り大きなフォント (最大 24 ポイント) を使用します :

```
ARRAY TEXT(TextArray;0)
```

```
$Size:=24
Repeat
   TEXT TO ARRAY($RawText;TextArray;400;"Arial";$Size)
   $Size:=$Size-1
   $n:=Size of array(TextArray)
Until($n<=80)
```

BLOB

BLOB PROPERTIES

BLOB PROPERTIES (blob ; compressed {; expandedSize {;currentSize}})

BLOB PROPERTIES コマンドは compressed 引数に、GZIP 圧縮に関連する 2 つの新しい値を返すようになりました (COMPRESS BLOB 参照)。

COMPRESS BLOB

COMPRESS BLOB(blob {; compression})

COMPRESS BLOB は compression メソッドに 2 つの新しい値を受け入れます。以下の定数を使用できます :

定数 (値)	説明
GZIP Best compression mode (-1)	GZIP 圧縮で圧縮率を優先します。
GZIP Fast compression mode (-2)	GZIP 圧縮で速度を優先します。

注 同時に EXPAND BLOB コマンドは GZIP の解凍をサポートします。

- GZIP で圧縮された raw HTTP データを送信します :

```
COMPRESS BLOB($blob; GZIP Best compression mode)
C_TEXT($vEncoding)
$vEncoding:="content-encoding";"gzip"
WEB SET HTTP HEADER($vEncoding)
WEB SEND RAW DATA($blob ; *)
```

データベースメソッド

On System Event

新しい On System Event データベースメソッドはシステムイベントが発生するたびに呼び出されます。

すべてのモードの 4D、4D Server、4D Volume Desktop が統合されたコンパイル済みアプリケーションなど、すべての 4D 環境で有効です。

イベントを処理するために、メソッドの中で $1 引数をテストし、"Database events" テーマの以下の定数と比較しなければなりません :

定数 (値)	コメント
On application background move (1)	4D アプリケーションがバックグランドに移動した
On application foreground move (2)	4D アプリケーションが最前面に移動した

注 これらのイベントはユーザーアクションにより 4D アプリケーションのレベルが変わったときに生成されます。例えば :
- 4D あるいは他のアプリケーションのウィンドウがクリックされた。
- **Alt+Tab** (Windows) または **Command+Tab** (Mac OS) キーボードショートカットでアプリケーションを選択した。
- Dock の隠すコマンドが選択された (Mac OS)。
- Dock やタスクバーでアプリケーションアイコンがクリックされた。
- メインウィンドウで最小化ボタンがクリックされた (Windows)。

データベースメソッド内では $1 を倍長引数型で宣言しなければなりません。データベースメソッドの構造は以下のようになります :

```
// On System Event データベースメソッド

C_LONGINT($1)

Case of
  :($1=On application background move)
      // 必要な処理を行う
  :($1=On application foreground move)
      // 必要な処理を行う
End case
```

On Web Session Suspend

On Web Session Suspend データベースメソッドは Web セッションが閉じられる直前に、4D Web サーバーから呼び出されます。4D は以下のような場合に Web セッション (セッションを管理する Web プロセス) を閉じます :

- セッションを管理する Web プロセス数の最大値 (デフォルトで 100、WEB SET OPTION コマンドで変更可能) に達している状態で、さらに新

しい Web セッションを作成する必要があるとき (4D は一番古い Web セッションプロセスを自動で破棄します)。

- セッションプロセスのタイムアウトに達したとき (デフォルトで 480 分 = 8 時間、WEB SET OPTION コマンドで変更可能)。
- WEB CLOSE SESSION コマンドが呼び出された場合

このデータベースメソッドが呼び出された時点で、セッションのコンテキスト (プロセス変数の値やカレントセレクション) は有効です。そのセッションに関連するデータ (変数の値やセレクション) を退避し、後で Web リクエストを処理するプロセスが同じ cookie 値でリクエストを受信したときにそれらを再利用することができます。

On Web Session Suspend データベースメソッドの利用例は 61 ページの "Web セッション管理 " を参照してください。

デザインオブジェクトアクセス

4D v13 ではインタープリターモードにおいて、プログラムを使用してアプリケーションのメソッドの内容にアクセスできます。この機能を使用すれば、アプリケーションのコードコントロールツール、特にバージョンコントロールシステム (VCS) への統合が容易になります。またコードのドキュメント化やカスタマイズされたエクスプローラー、ディスクファイルへの定期的なコードバックアップを実装できるようになります。

以下の原則が適用されます :

- 4D アプリケーション内の各メソッドやフォームはパス名形式で固有のアドレスを持ちます。例えば table_1 のトリガーメソッドは "[trigger]/table_1" で見つけることができます。各オブジェクトパス名はアプリケーション内でユニークです。

注 パス名がユニークであることを保証するために、異なるフォームページ上であったとしても同じ名前のオブジェクトを作ることはできません。4D v13 より前から変換されたデータベースでこの状況が発生している場合、MSC はこのような重複した名前を検出します。

- METHOD GET NAMES や METHOD GET PATHS のような、このテーマのコマンドを使用して 4D アプリケーション内のオブジェクトにアクセスできます。
- このテーマのコマンドはすべて 4D ローカルモードおよびリモートモードで使用できます。
しかしこれらのコマンドをコンパイルモードで使用することはできませ

ん。これらのコマンドの目的はカスタマイズされた開発サポートツールを作成することにあります。実行中のデータベースの機能を動的に変更するために使用してはなりません。例えばカレントユーザーのアクセス権に基づき METHOD SET ATTRIBUTE コマンドでメソッド属性を変更するというようなことはできません。

- このテーマのコマンドがコンポーネントから呼ばれた場合、デフォルトでその呼び出しはコンポーネントオブジェクトにアクセスします。コンポーネントからホストデータベースオブジェクトにアクセスする場合は最後に引数に * を渡します。なお (METHOD SET ATTRIBUTE のような) メソッド属性を変更するコマンドを使用する場合、必ずこのシンタックスを使用しなければならない点に留意してください。コンポーネントメソッドは常に読み込みのみモードです。

パス名の作成

デフォルトで 4D はディスクファイルを作成することはしません。しかしオブジェクトのために作成されたパス名は OS のファイル管理システムと互換性があります。読み込みや書き出しメソッドを作成し、パス名を使用してディスクに直接ファイルを生成できます。

":" のような使用不可文字はエンコードされます。生成されるファイルを自動でバージョン管理システムに統合することができます。

以下はエンコードが行われる文字のリストです :

文字	エンコード結果
"	%22
*	%2A
/	%2F
:	%3A
<	%3C
>	%3E
?	%3F
|	%7C
\	%5C
%	%25

例 :

- Form?1 は Form%3F1 にエンコードされます。
- Button/1 は Button%2F1 にエンコードされます。

新しいマクロタグ

4D のマクロコマンドに新しいタグ **<method_path>** が追加されました。このタグはコード中で実行中のプロジェクトメソッドのフルパス名に置換されます。

Current method path

Current method path → テキスト

引数	型		説明
このコマンドは引数を必要としません			
戻り値	テキスト	←	実行中のメソッドの内部的な完全パス名

Current method path コマンドは実行中の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、またはオブジェクトメソッドの内部的なパス名を返します。

Current method name と異なり、返される値はデータベース中でユニークである点に留意してください。

注 4D マクロコマンドのコンテキストでは、<method_path> タグが実行中のメソッドのフルパス名で置き換えられます。

FORM GET NAMES

FORM GET NAMES({aTable;} arrNames{; filter} {; *})

引数	型		説明
aTable	テーブル	→	テーブル参照
arrNames	テキスト配列	←	フォーム名の配列
filter	テキスト	→	名前のフィルター
*		→	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)

FORM GET NAMES コマンドはアプリケーション中のフォーム名を arrNames 配列に返します。

aTable 引数を渡すと、コマンドはそのテーブルに属するテーブルフォームの名前を返します。この引数を省略するとデータベースプロジェクトフォーム名が返されます。

filter 引数に比較文字列を渡すことでフォームのリストを制限できます。この場合、フィルターにマッチする名前を持つフォームだけが返されます。"@" をワイルドカードとして使用することができます。空の文字列を渡した場合、filter 引数は無視されます。

コマンドがコンポーネント内で実行された場合、デフォルトではコンポーネントのプロジェクトフォーム名が返されます。* 引数を渡すとホストデータベースのフォーム名を受け取ることができます。

注 ゴミ箱内のフォームは返されません。

♦ 典型的な使用例 :

```
  // データベース中すべてのプロジェクトフォームを取得
FORM GET NAMES(arr_Names)

  // [Employees] テーブルのフォームを取得
FORM GET NAMES([Employees] ; arr_Names)

  // [Employees] テーブル中 "input_" で始まるフォームを取得
FORM GET NAMES([Employees] ; arr_Names ;"input_@")
  // データベース中で特定のプロジェクトフォームを取得
FORM GET NAMES(arr_Names ; "dialogue_@")

  // コンポーネントからホストのテーブルフォームを取得
  // テーブル名が不明なためポインターを使用
FORM GET NAMES(tablePtr-> ; arr_Names; *)
```

METHOD Get attribute

METHOD Get attribute(path ; attribType {; *}) → ブール

引数	型		説明
path	テキスト	→	プロジェクトメソッドのパス
attribType	倍長整数	→	取得する属性タイプ
*		→	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)

戻り値	ブール	←	True: 属性が選択されている、False: 選択されていない

METHOD Get attribute コマンドは path 引数で指定されたプロジェクトメソッドの attribType 属性値を返します。このコマンドはプロジェクトメソッドに対してのみ動作します。無効なパスを渡すとエラーが生成されます。

attribType 引数には取得する属性のタイプを指定する値を渡します。"Design Object Access" テーマの以下の定数を使用できます :

定数 (値)	説明
Attribute Executed on server (8)	" サーバー上で実行 " オプションに対応
Attribute Invisible (1)	" 隠す " オプションに対応
Attribute Published SOAP (3)	"Web サービスとして提供 " オプションに対応
Attribute Published SQL (7)	"SQL 利用可 " オプションに対応
Attribute Published Web (2)	"4D タグおよび URL (4DACTION...) で利用可 " オプションに対応
Attribute Published WSDL (4)	"WSDL で公開する " オプションに対応
Attribute Shared (5)	" コンポーネントとホストデータベースで共有する " オプションに対応

注 これらの各属性に関する詳細は 4D デザインリファレンスマニュアルを参照してください。

コマンドがコンポーネントから実行された場合、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のメソッドに適用されます。

コマンドは属性が選択されている場合 True を、選択されていない場合 False を返します。

参照 : METHOD SET ATTRIBUTE

METHOD GET CODE

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

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
*			→	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)

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) // ひと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) // 複数のメソッドのコード
```

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を除き英語で記述されます。
- 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 :
  // %metadata= {invisible:true,lang:"fr"}
  読み込み時にはこの行は取り除かれ、指定された属性のみが考慮されます。"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。

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。

- メソッドの書き出しと読み込みに関する例題はコマンド METHOD SET CODE (107 ページ) を参照してください。

参照 : METHOD SET CODE

METHOD GET COMMENTS

METHOD GET COMMENTS (path ; comments {; *})

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
comments	テキスト	テキスト配列	←	指定したメソッドのコメント
*			→	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)

METHOD GET COMMENTS コマンドは path 引数で指定したメソッドのコメントを comments 引数に返します。

このコマンドを使用して取得することのできるコメントは、4D エクスプローラーのコメント欄で定義されたものです (METHOD GET CODE コマンドを使用して取得できるコード内のコメントではありません):

このコメントはトリガー、プロジェクトメソッド、そしてフォームメソッドに対して作成できます。またスタイルを付けることができます。

注 フォームとフォームメソッドは同じコメントを共有します。

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(tVcomments)
```

```
METHOD GET COMMENTS(tVpath;tVcomments) // ひとつのメソッドのコメント

ARRAY TEXT(arrPaths;0) // テキスト配列
ARRAY TEXT(arrComments;0)
METHOD GET COMMENTS(arrPaths;arrComments) // 複数のメソッドのコメント
```

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。

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のメソッドにアクセスします。

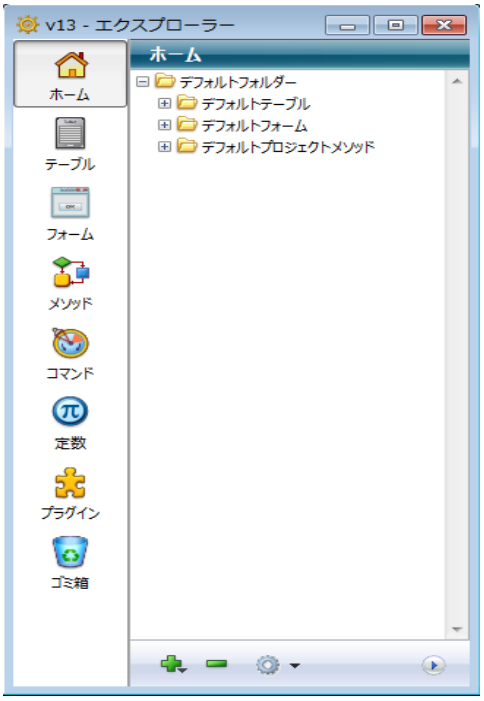
参照 : METHOD SET COMMENTS

METHOD GET FOLDERS

METHOD GET FOLDERS(arrNames{; filter}{; *})

引数	型		説明
arrNames	テキスト配列	←	ホームページのフォルダー名配列
filter	テキスト	→	名前フィルター
*		→	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)

METHOD GET FOLDERS コマンドは 4D エクスプローラーのホームページに作成されたフォルダー名を arrNames 配列に返します :



フォルダー名はユニークでなければならないため、この配列に階層は返されません。

filter 引数に比較文字列を渡して、このフォルダーリストを絞り込むことができます。この場合コマンドは名前がフィルターにマッチするフォルダーのみを返します。"@" をワイルドカードとして使用することができます。空の文字列を渡すと filter 引数は無視されます。

このコマンドがコンポーネントから呼ばれると、デフォルトでコンポーネントのフォルダー名が返されます。* 引数を渡すと配列にはホストデータベースのフォルダーが返されます。

METHOD GET MODIFICATION DATE

METHOD GET MODIFICATION DATE(path ; modDate; modTime{; *})

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
modDate	日付	日付配列	←	メソッド更新日

modTime	時間 | 倍長整数配列	←	メソッド更新時刻
*		→	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)

METHOD GET MODIFICATION DATE コマンドは path 引数で指定されたメソッドの更新日と時刻をそれぞれ modDate と modTime 引数に返します。

配列または変数に基づく 2 タイプのシンタックスを使用できます :

```
C_TEXT(tVpath) // 変数
C_DATE(vDate)
C_TIME(vTime)
METHOD GET MODIFICATION DATE(tVpath;vDate;vTime)
  // 1 つのメソッドの日付と時刻
```

```
ARRAY TEXT(arrPaths;0) // 配列
ARRAY DATE(arrDates;0)
ARRAY LONGINT(arrTimes;0)
METHOD GET MODIFICATION DATE(arrPaths;arrDates;arrTimes)
  // 複数メソッドの日付と時刻
```

この 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。

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のメソッドにアクセスします。

METHOD GET NAMES

METHOD GET NAMES(arrNames{; filter} {; *})

引数	型		説明
arrNames	テキスト配列	←	プロジェクトメソッド名配列
filter	テキスト	→	名前フィルター
*		→	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)

METHOD GET NAMES コマンドはアプリケーション中のプロジェクトメソッドの名前を arrNames 配列に返します。

デフォルトではすべてのメソッドがリストされます。filter 引数に比較文字列を渡して、このリストを絞り込むことができます。この場合コマンドは名前がフィルターにマッチするメソッドのみを返します。"@" をワイルドカードとして使用することができます。空の文字列を渡すと filter 引数は無視されます。

このコマンドがコンポーネントから呼ばれると、デフォルトでコンポーネントプロジェクトメソッドの名前が返されます。* 引数を渡すと配列にはホストデータベースのプロジェクトメソッドが返されます。

注 ゴミ箱内のメソッドは含まれません。

♦ 典型的な利用例 :

```
  // データベースのすべてのプロジェクト名をリストする
METHOD GET NAMES(t_Names)

  // 特定の文字から始まるメソッドのみをリストする
METHOD GET NAMES(t_Names; "web_@")

  // ホストデータベース内で、特定の文字から始まるメソッドのみを
  // リストする
METHOD GET NAMES(t_Names; "web_@";*)
```

METHOD Get path

METHOD Get path (methodType {; aTable}{; objectName{; formObjectName}} {; *}) → テキスト

引数	型		説明
methodType	倍長整数	→	オブジェクトタイプセレクター
aTable	テーブル	→	テーブル参照
objectName	テキスト	→	フォームまたはデータベースメソッド名
formObjectName	テキスト	←	フォームオブジェクト名
*		→	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)

戻り値　　Text　　← オブジェクトのフルパス

METHOD Get path コマンドはメソッドの完全な内部パス名を返します。

methodType 引数にはパスを取得したいメソッドのタイプを渡します。Design Object Access テーマの以下の定数を使用できます :

定数 (値)	説明
Path Database method (2)	指定したデータベースメソッド名。以下のメソッドのリスト : [databaseMethod]/onStartup [databaseMethod]/onExit [databaseMethod]/onDrop [databaseMethod]/onBackupStartup [databaseMethod]/onBackupShutdown [databaseMethod]/onWebConnection [databaseMethod]/onWebAuthentication [databaseMethod]/onWebSessionSuspend [databaseMethod]/onServerStartup [databaseMethod]/onServerShutdown [databaseMethod]/onServerOpenConnexion [databaseMethod]/onServerCloseConnection [databaseMethod]/onSystemEvent [databaseMethod]/onSqlAuthentication
Path Project form (4)	プロジェクトフォームメソッドとすべてのフォームオブジェクトメソッドのパス。例 : [projectForm]/myForm/{formMethod} [projectForm]/myForm/button1 [projectForm]/myForm/my%2list [projectForm]/myForm/button1
Path Project method (1)	メソッド名。 例 : MyProjectMethod
Path Table form (16)	テーブルフォームメソッドとすべてのフォームオブジェクトメソッド。例 : [tableForm]/table_1/Form1/{formMethod} [tableForm]/table_1/Form1/button1 [tableForm]/table_1/Form1/my%2list [tableForm]/table_2/Form1/my%2list
Path Trigger (8)	データベーストリガーのパス。例 : [trigger]/table_1 [trigger]/table_2

aTable, objectName および formObjectName 引数にはメソッドパス名を取得したいオブジェクトのタイプに応じて値を渡します :

オブジェクトタイプ	aTable	objectName	formObjectName
Path Database method		○	
Path Project form		○	○ (省略可)
Path Project method		○	
Path Table form	○	○	○ (省略可)
Path Trigger	○		

オブジェクトが見つからない場合 (メソッドタイプが未知か無効、テーブルが見つからない等)、エラーが生成されます。

コマンドがコンポーネントから実行された場合、デフォルトでコンポーネントメソッドのパスが返されます。* 引数を渡すと配列にはホストデータベースのメソッドパス名が返されます。

- "On Startup" データベースメソッドのパス名を取得 :

  $path:=**METHOD Get path**(Path Database method;"onStartup")

- [Employees] テーブルのトリガーのパス名を取得 :

  $path:=**METHOD Get path**(Path Trigger;[Employees])

- [Employees] テーブルの "input" フォームの "OK" オブジェクトメソッドのパス名を取得 :

  $path:=**METHOD Get path**(Path Table form;[Employees];"input"; "OK")

参照 : METHOD RESOLVE PATH

METHOD GET PATHS

METHOD GET PATHS({folderName;} methodType; arrPaths{; stamp} {; *})

引数	型		説明
folderName	テキスト	→	ホームページのフォルダー名
methodType	倍長整数	→	取得するメソッドタイプセレクター
arrPaths	テキスト配列	←	メソッドパスおよび名前の配列
stamp	倍長整数変数	→	スタンプの最小値
		←	新しい現在値

*

→ 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)

METHOD GET PATHS コマンドはアプリケーションのうち、methodType 引数で指定したタイプであるメソッドの内部的なパス名と名前を arrPaths 配列に返します。

メソッドが 4D エクスプローラーのホームページでフォルダーを使用して階層化されていれば、オプションの folderName 引数にフォルダー名を渡すことができます。この場合 arrPaths 配列にはこのフォルダーに含まれるメソッドのみのパスが返されます。

---

folderName に "@" 文字は使用できません。

---

methodType 引数には arrPaths 配列にパスを取得したいメソッドのタイプを渡します。Design Object Access テーマの以下の定数を個別にあるいは加算して使用できます :

定数 (値)	説明
Path All objects (31)	データベースのすべてのメソッドのパス
Path Database method (2)	指定したデータベースメソッド名。以下のメソッドのリスト : [databaseMethod]/onStartup [databaseMethod]/onExit [databaseMethod]/onDrop [databaseMethod]/onBackupStartup [databaseMethod]/onBackupShutdown [databaseMethod]/onWebConnection [databaseMethod]/onWebAuthentication [databaseMethod]/onWebSessionSuspend [databaseMethod]/onServerStartup [databaseMethod]/onServerShutdown [databaseMethod]/onServerOpenConnexion [databaseMethod]/onServerCloseConnection [databaseMethod]/onSystemEvent [databaseMethod]/onSqlAuthentication

Path Project form (4)	プロジェクトフォームメソッドとすべてのフォームオブジェクトメソッドのパス。例 : [projectForm]/myForm/{formMethod} [projectForm]/myForm/button1 [projectForm]/myForm/my%2list [projectForm]/myForm/button1
Path Project method (1)	メソッド名。 例 : MyProjectMethod
Path Table form (16)	テーブルフォームメソッドとすべてのフォームオブジェクトメソッド。例 : [tableForm]/table_1/Form1/{formMethod} [tableForm]/table_1/Form1/button1 [tableForm]/table_1/Form1/my%2list [tableForm]/table_2/Form1/my%2list
Path Trigger (8)	データベーストリガーのパス。例 : [trigger]/table_1 [trigger]/table_2

stamp 引数を使用すれば特定の時点以降に更新されたメソッドのパスだけを取得できます。バージョンコントロールシステムの一部として、最新のバックアップ以降に更新されたメソッドだけをアップデートするようにできます。
動作の概要 : 4D はメソッド更新カウンターを保持します。メソッドが作成され再び保存されるたびに、このカウンターは増分され、その現在値がメソッドの内部的なスタンプに格納されます。
stamp 引数を渡すと、コマンドはこの引数に渡された値と同じかそれより大きなスタンプ値を持つメソッドのみを返します。さらにコマンドは stamp 引数に更新カウンターの新しい値を返します。この値を保持すれば、次回このコマンドを呼び出す際にこの値を渡すことができ、前回のコマンド実行以降に作成あるいは更新されたメソッドのみを取得できます。

メソッドがコンポーネントから実行された場合、デフォルトでコンポーネントメソッドのパスが返されます。* 引数を渡すと配列にはホストデータベースのメソッドパスが返されます。

- "web" フォルダー内のプロジェクトメソッドを取得する :

  METHOD GET PATHS("web", Path Project method; arrPaths)

- データベースメソッドとトリガーを取得する :

  METHOD GET PATHS(Path Trigger+Path Database method; arrPaths)

- 最新のバックアップ以降に更新されたプロジェクトメソッドを取得する :

```
// 前回のスタンプ値を参照
$stamp := Max([Backups]cur_stamp)
METHOD GET PATHS(Path Project method; arrPaths;$stamp)
// 最新のスタンプ値を保存
CREATE RECORD([Backups])
[Backups]cur_stamp := $stamp
SAVE RECORD([Backups])
```

- 完全な例題についてはコマンド METHOD SET CODE (107 ページ) の説明を参照してください。

参照 : METHOD GET FOLDERS

METHOD GET PATHS FORM

METHOD GET PATHS FORM({aTable;} arrPaths {; filter}{; stamp}{; *})

引数	型		説明
aTable	Table	→	テーブル参照
arrPaths	テキスト配列	←	メソッドパスと名前の配列
filter	Text	→	名前フィルター
stamp	倍長整数 variable	→	スタンプの最小値
		←	新しい現在値
*		→	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)

METHOD GET PATHS FORM コマンドはすべてのフォームオブジェクトとフォームメソッドの内部的なパス名と名前を arrPaths 配列に返します。フォームメソッドには {formMethod} とラベルが付けられます。

コードを含むオブジェクトのみがリストされます。例えば標準アクションのみが割り当てられたボタンは返されません。

aTable 引数を渡すと、コマンドはこのテーブルのテーブルフォームのオブジェクトを返します。この引数を省略するとコマンドはデータベースプロジェクトフォームのオブジェクトを返します。

filter 引数に比較文字列を渡すことでリストを制限できます。この場合フィルターにマッチするフォームだけが返されます。"@" をワイルドカー

ド文字として使用できます。空の文字列を渡すと filter 引数は無視されます。

stamp 引数を使用すれば特定の時点以降に更新されたメソッドのパスだけを取得できます。バージョンコントロールシステムの一部として、最新のバックアップ以降に更新されたメソッドだけをアップデートするようにできます。
動作の概要 : 4D はメソッド更新カウンターを保持します。メソッドが作成され再び保存されるたびに、このカウンターは増分され、その現在値がメソッドの内部的なスタンプに格納されます。
stamp 引数を渡すと、コマンドはこの引数に渡された値と同じかそれより大きなスタンプ値を持つメソッドのみを返します。さらにコマンドは stamp 引数に更新カウンターの新しい値を返します。この値を保持すれば、次回このコマンドを呼び出す際にこの値を渡すことができ、前回のコマンド実行以降に作成あるいは更新されたメソッドのみを取得できます。

メソッドがコンポーネントから実行された場合、デフォルトでコンポーネントメソッドのパスが返されます。* 引数を渡すと配列にはホストデータベースのメソッドパスが返されます。

注 コマンドは継承フォームやサブフォームのオブジェクトを返しません。

- [Employees] テーブルの "input" フォームのすべてのオブジェクトをリストします。テーブルフォームメソッド (そしてプロジェクトフォームメソッド) はフォームに属するオブジェクトとして処理される点に留意してください :

```
METHOD GET PATHS FORM([Employees];arrPaths;"input")
  // arrPaths 配列の内容例
  // [tableForm]/input/{formMethod} -> フォームメソッド
  // [tableForm]/input/bOK -> オブジェクトメソッド
  // [tableForm]/input/bCancel -> オブジェクトメソッド
```

- "dial" プロジェクトフォームのオブジェクトをリスト :

```
METHOD GET PATHS FORM(arrPaths;"dial")
```

- コンポーネントから [Employees] テーブル中 "input" で始まるフォームのすべてのオブジェクトをリスト :

```
METHOD GET PATHS FORM(([Employees];arrPaths;"input@";*)
```

METHOD OPEN PATH

METHOD OPEN PATH(path{; *})

引数	型	説明

path	テキスト	→	開くメソッドのパス
*		→	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)

METHOD OPEN PATH コマンドは内部パス名が path 引数であるメソッドを 4D メソッドエディターで開きます。

このコマンドですべてのタイプ (オブジェクト、フォーム、トリガー、プロジェクト、データベース) のメソッドを開くことができますが、メソッドは既に存在していなければなりません。path 引数が存在しないメソッドを指している場合、エラー -9801 " メソッドを開けません " が生成されます。

コマンドをコンポーネントから実行することもできますが、この場合 (コンポーネントコードには書き込みアクセスができないため) * 引数を渡さなければなりません。この状況で * 引数を省略するとエラー -9763 が生成されます。

参照 : METHOD Get path

METHOD RESOLVE PATH

METHOD RESOLVE PATH(path; methodType ; ptrTable; objectName; formObjectName{; *})

引数	型		説明
path	テキスト	→	解決するパス
methodType	倍長整数	←	オブジェクトタイプセレクター
ptrTable	ポインター	←	テーブル参照
objectName	テキスト	←	フォームまたはデータベースメソッド名
formObjectName	テキスト	←	フォームオブジェクト名
*		→	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)

METHOD RESOLVE PATH コマンドは path 引数に渡された内部パス名を解決し、methodType、ptrTable、objectName、そして formObjectName 引数にそれぞれ情報を返します。

methodType 引数はメソッドのタイプを示す値を受け取ります。この値を Design Object Access テーマの定数と比較することができます :

定数 (値)	説明
Path Database method (2)	データベースメソッドのパス
Path Project form (4)	プロジェクトフォームメソッドとすべてのフォームオブジェクトメソッドのパス
Path Project method (1)	プロジェクトメソッドのパス
Path Table form (16)	テーブルフォームメソッドとすべてのフォームオブジェクトメソッドのパス
Path Trigger (8)	トリガーのパス

ptrTable 引数は、パスがテーブルフォームメソッドやトリガーを参照する場合、データベーステーブルへのポインターを受け取ります。

objectName 引数は以下のいずれかを受け取ります :

- パスがテーブルフォームまたはプロジェクトフォームを参照する場合、フォーム名
- パスがデータベースメソッドを参照する場合、データベースメソッド名

formObjectName 引数は、パスがオブジェクトメソッドを参照する場合、フォームオブジェクト名を受け取ります。

コマンドがコンポーネントから実行された場合、デフォルトではコンポーネントメソッドに関する情報を返します。* 引数を渡すと、配列にはホストデータベースのメソッドパスが返されます。

♦ データベースメソッドパスの解決 :

```
C_LONGINT($methodType)
C_POINTER ($tablePtr)
C_TEXT($objectName)
C_TEXT($formObjectName)

METHOD RESOLVE PATH("[databaseMethod]/onStartup";

$methodType;$tablePtr;$objectName;$formObjectName)
```

```
// $methodType: 2
// $tablePtr: Nil ポインター
// $objectName: "onStartup"
// $formObjectName: ""
```

- テーブルフォームのオブジェクトメソッドのパス解決 :

```
C_LONGINT($methodType)
C_POINTER ($tablePtr)
C_TEXT($objectName)
C_TEXT($formObjectName)

METHOD RESOLVE PATH("[tabletForm]/Table1/output%2A1/

myVar%2A1";$methodType;$tablePtr;$objectName;
                                        $formObjectName)
// $methodType: 16
// $tablePtr: -> [Table1]
// $objectName: "output*1"
// $formObjectName: "Btn*1"
```

参照 : METHOD Get path

METHOD SET ACCESS MODE

METHOD SET ACCESS MODE (mode)

引数	型		説明
mode	倍長整数	→	ロックされたオブジェクトのアクセスモード

METHOD SET ACCESS MODE コマンドはすでに他のユーザーやプロセスにより更新のためにロードされたオブジェクトに書き込みアクセスを行おうとした時の振る舞いを設定します。このコマンドのスコープはカレントセッションです。

mode 引数には Design Object Access テーマの以下の定数を渡します :

定数 (値)	説明
On object locked abort (0)	オブジェクトのロードが中断された (デフォルト動作)
On object locked retry (1)	オブジェクトがリリースされるまでオブジェクトのロードを試行します。

On object locked confirm (2)	再試行するか中断するか選択できるようにするためダイアログを表示します。 リモートモードではこのオプションはサポートされません (ロードは中断されます)。

METHOD SET ATTRIBUTE

METHOD SET ATTRIBUTE(path ; attribType; attribValue {; *})

引数	型		説明
path	テキスト	→	プロジェクトメソッドのパス
attribType	倍長整数	→	属性タイプ
attribValue	ブール	→	True = 属性を選択 False = 属性の選択解除
*		→	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)

METHOD SET ATTRIBUTE コマンドは path 引数で指定されたプロジェクトメソッドの attribType 属性を設定します。このコマンドはプロジェクトメソッドに対してのみ動作します。無効なパスを渡すとエラーが生成されます。

attribType 引数には設定を行う属性のタイプを示す値を渡します。Design Object Access テーマの以下の定数を使用できます :

定数 (値)	説明
Attribute Executed on server (8)	" サーバー上で実行 " オプションに対応
Attribute Invisible (1)	" 隠す " オプションに対応
Attribute Published SOAP (3)	"Web サービスとして提供 " オプションに対応
Attribute Published SQL (7)	"SQL 利用可 " オプションに対応
Attribute Published Web (2)	"4D タグおよび URL (4DACTION...) で利用可 " オプションに対応

Attribute Published WSDL (4)	"WSDL で公開する " オプションに対応
Attribute Shared (5)	" コンポーネントとホストデータベースで共有する " オプションに対応

注 - これらの属性に関する詳細は 4D Design Reference マニュアルを参照してください。
- "WSDL で公開する " オプションは "Web サービスとして提供 " オプションの設定に影響されます。Attribute Published SOAP を選択せずに Attribute Published WSDL 属性を選択しても、設定は有効になりません。

対応するオプションを選択するには attribValue に True を渡します。選択を解除するには False を渡します。

コマンドをコンポーネントから実行することもできますが、この場合 (コンポーネントコードには書き込みアクセスができないため) * 引数を渡さなければなりません。この状況で * 引数を省略するとエラー -9763 が生成されます。

- "Choose dialog" プロジェクトメソッドの " コンポーネントとホストデータベースで共有 " プロパティを選択します :

METHOD SET ATTRIBUTE("Choose dialog";Attribute Shared;**True**)

参照 : METHOD Get attribute

METHOD SET CODE

METHOD SET CODE (path ; code {; *})

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
*			→ 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)

METHOD SET CODE コマンドは path 引数で指定したメソッドのコードを code 引数に渡した内容で置き換えます。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、オブ

ジェクトメソッドなどすべてのタイプのメソッドのコードにアクセスできます。

プロジェクトメソッド : データベースにメソッドが既に存在する場合、内容が置換されます。存在しなければ新しく作成されます。

テキスト配列あるいはテキスト変数を使用する 2 通り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 SET CODE(tVpath;tVcode) // ひと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 SET CODE(arrPaths;arrCodes) // 複数のメソッドのコード
```

この 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。

無効なパス名を渡した場合、コマンドはなにも行いません。

メソッドコードの先頭行に有効なメタデータが含まれる場合、それらはメソッド属性を指定するために使用され、先頭行はコードには含まれません。メタデータの例は以下の通りです :

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

注 T これらのメタデータは METHOD GET CODE コマンドで自動的に生成されます。

コマンドをコンポーネントから実行することもできますが、この場合 (コンポーネントコードには書き込みアクセスができないため) * 引数を渡さなければなりません。この状況で * 引数を省略するとエラー -9763 が生成されます。

- この例題ではアプリケーションのすべてのメソッドを書き出し / 読み込みします :

```
$root_t:=Get 4D folder(Database Folder)+"methods"+Folder separator
ARRAY TEXT($fileNames_at;0)
CONFIRM(" メソッドを読み込みますか、書き出しますか ?"; "Import";
"Export")

If (OK=1)
   DOCUMENT LIST ($root_t;$fileNames_at)
   For ($loop_l;1;Size of array($fileNames_at))
      $filename_t:=$fileNames_at{$loop_l}
```

```
      DOCUMENT TO BLOB($root_t+$filename_t;$blob_x)
      METHOD SET CODE($filename_t;BLOB to text($blob_x;
                                          UTF8 Text without length))
   End for
Else
   If (Test path name($root_t)#Is a folder)
      CREATE FOLDER($root_t;*)
   End if
   METHOD GET PATHS(Path Project method;$fileNames_at)
   METHOD GET CODE($fileNames_at;$code_at)
   For ($loop_l;1;Size of array($fileNames_at))
      $filename_t:=$fileNames_at{$loop_l}
      SET BLOB SIZE($blob_x;0)
      TEXT TO BLOB($code_at{$loop_l};$blob_x;
                                          UTF8 Text without length)
      BLOB TO DOCUMENT($root_t+$filename_t;$blob_x;*)
   End for
End if
SHOW ON DISK($root_t)
```

参照 : METHOD GET CODE

METHOD SET COMMENTS

METHOD SET COMMENTS (path ; comments {; *})

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
comments	テキスト	テキスト配列	→	指定したメソッドに設定するコメント
*			→	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)

METHOD SET COMMENTS コマンドは path 引数で指定したメソッドのコメントを comments 引数の内容で置き換えます。

このコマンドを使用して更新することのできるコメントは、4D エクスプローラーのコメント欄で定義されたものです (METHOD GET CODE コマンドを使用して取得できるコード内のコメントではありません)。このコメントはトリガー、プロジェクトメソッド、そしてフォームメソッドに対して作成できます。またスタイルを付けることができます。

注 フォームとフォームメソッドは同じコメントを共有します。

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(tVcomments)
METHOD SET COMMENTS(tVpath;tVcomments) // 1 つのメソッドのコメント

ARRAY TEXT(arrPaths;0) // テキスト配列
ARRAY TEXT(arrComments;0)
METHOD SET COMMENTS(arrPaths;arrComments) // 複数メソッドのコメント
```

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。

無効なパス名を指定するとエラーが生成されます。

コマンドをコンポーネントから実行することもできますが、この場合 (コンポーネントコードには書き込みアクセスができないため) * 引数を渡さなければなりません。この状況で * 引数を省略するとエラー -9763 が生成されます。

- 既存のトリガーコメントに更新日を追加する :

```
METHOD GET COMMENTS("[trigger]/Table1";$comments)
$comments:="Modif:"+String(Current date)+"\r"+$comments
METHOD SET COMMENTS("[trigger]/Table1";$comments)
```

参照 : METHOD GET COMMENTS

フォームイベント

Form event

Form event→ 倍長整数

4D v13 に新しいフォームイベントが追加され、さらに **On Expand** と **On Collapse** が階層リストボックスで利用できるようになりました。

新しいイベント

4D v13 では新しいフォームイベントが複数追加されました :

定数	型	値	コメント
On Page Change	倍長整数	56	フォーム中のカレントページが変更された

On Footer Click	倍長整数	57	リストボックスあるいはリストボックス列でフッターがクリックされた
On Delete Action	倍長整数	58	(階層リストとリストボックスのみ) ユーザーが項目の削除を試みた

- **On Page Change**: このイベントはフォームレベルでのみ利用でき (フォームメソッド内でのみ使用します)、フォームのカレントページが変更されるたびに生成されます (FORM GOTO PAGE コマンドの呼び出しや標準ナビゲーションアクションに伴い)。
ページが完全にロードされた後に呼び出されることに留意してください。例えば (Web エリアを含む) すべてのオブジェクトが初期化された後です。このイベントはすべてのオブジェクトが初期化済みの状態で実行する必要のあるコードがあるときに有用です。またフォームがロードされたときにすべてのコードを実行するのではなく、特定のページが開かれたときにのみコードを実行するようにして、アプリケーションを最適化できます。ユーザーがそのページを開かなければ、コードは実行されません。

- **On Footer Click**: このイベントはリストボックスやリストボックス列で利用でき、リストボックスやリストボックス列のフッターエリアがクリックされたときに生成されます。
この場合 Object Get pointer コマンドはクリックされたフッター変数へのポインターを返します。イベントは左および右クリックどちらでも生成されます。
リストボックスフッターに関する詳細は 27 ページの " フッター " を参照してください。

- **On Delete Action:** このイベントはユーザーが削除キー (**Delete** や **Backspace** キー) を押して、または編集メニューから削除を選択して、選択された行の削除を指示したときに生成されます。
リストボックスの場合、このイベントはリストボックスオブジェクトレベルでのみ利用でき、ユーザーが行の削除を要求したときに生成されます。
4D はイベントの生成を行うだけであることに留意してください。4D は項目を消去しません。実際の削除処理や事前警告の表示などは開発者の責任です (例題参照)。

♦ 階層リストで削除アクションを管理するコード例 :

```
  // 階層リストのメソッド
: ($Event=On Delete Action)
  ARRAY LONGINT($itemsArray;0)
  $Ref:=Selected list items(<>HL;$itemsArray;*)
  $n:=Size of array($itemsArray)
```

```
Case of
   : ($n=0)
      ALERT(" 項目が選択されていません。")
      OK:=0
   : ($n>0)
      CONFIRM(" 選択された項目を削除しますか ?")
End case

If (OK=1)
   For ($i;1;$n)
      DELETE FROM LIST (<>HL;$itemsArray{$i};*)
   End for
End if
```

既存のイベントの拡張

On Expand と **On Collapse** フォームイベントを階層リストボックスで使用できるようになりました。これらのフォームイベントを使用すると、フィールドに格納された値を必要に応じて配列にロードできるようになり、メモリおよびネットワークトラフィックを最適化できます。この点に関する詳細は 37 ページの " 階層リストボックスで使用する新しいイベント " を参照してください。

フォーム

FORM Get current page

FORM Get current page {(*)} → 倍長整数

引数	型		説明
*		→	カレントサブフォームページ番号を返す
戻り値	倍長整数	←	現在表示されているページ番号

既存の FORM Get current page コマンドがオプションの * 引数を受け入れるようになりました。

この引数は複数のページを含むサブフォームタイプのページのコンテキストでコマンドが呼び出される場合に使用します。この場合、この引数を渡すと、コマンドは (コマンドを呼び出した) カレントサブフォームのページを変更します。
* が省略された場合、デフォルトでコマンドは常に親フォームに適用されます。

FORM GOTO PAGE

FORM GOTO PAGE (pageNumber {; *})

引数	型		説明
pageNumber	倍長整数	→	表示するフォームページ
*		→	カレントサブフォームのページを変更

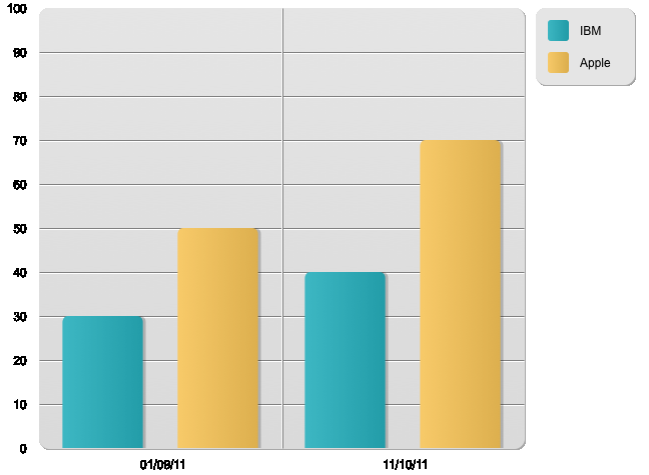
既存の FORM Get current page コマンドがオプションの * 引数を受け入れるようになりました。

この引数は複数のページを含むサブフォームタイプのページのコンテキストでコマンドが呼び出される場合に使用します。この場合、この引数を渡すと、コマンドは (コマンドを呼び出した) カレントサブフォームのページを変更します。
* が省略された場合、デフォルトでコマンドは常に親フォームに適用されます。

グラフ

GRAPH

4D の SVG 描画エンジンを使用して描画されるグラフのアピアランスが更新されました (ピクチャー変数を使用する場合)。以下のようなグラフを生成できるようになりました :

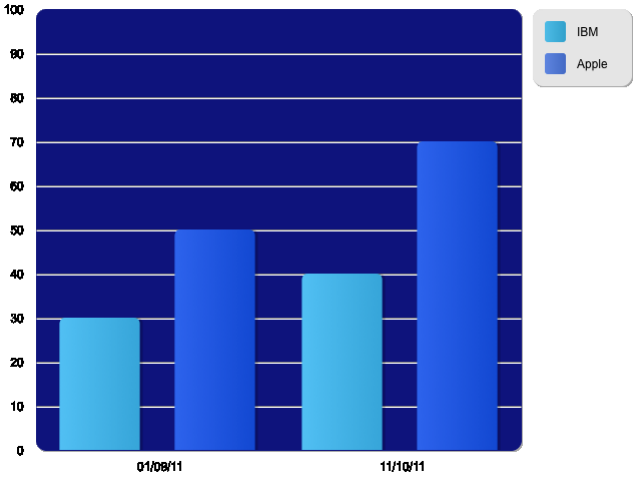


さらに SVG グラフ内の要素に自動で特定の ID が割り当てられるようになりました :

ID	説明

ID_graph_1 ～ ID_graph_8	棒、折れ線、、、
ID_graph_shadow_1 ～ ID_graph_shadow_8	グラフの影
ID_bullet_1 ～ ID_bullet_8	点 (折れ線および散布図のみ)
ID_pie_label_1 ～ ID_pie_label_8	円グラフのラベル (円グラフのみ)
ID_legend_1 ～ ID_legend_8	凡例のタイトル
ID_legend_border	凡例の境界線
ID_legend_border_shadow	凡例の境界線の影
ID_x_values	X 軸の値
ID_y_values	Y 軸の値
ID_y0_axis	Z 軸の値
ID_background	背景
ID_background_shadow	背景の影
ID_x_grid	X 軸のグリッド
ID_x_grid_shadow	X 軸のグリッドの影
ID_y_grid	Y 軸のグリッド
ID_y_grid_shadow	Y 軸のグリッドの影

4D の統合 SVG コマンドを使用して、上記のグラフを例えば以下のように変換できます :



階層リスト

GET LIST ITEM PARAMETER ARRAYS

GET LIST ITEM PARAMETER ARRAYS ({*; }list; itemRef | *; arrSelectors{; arrValues})

引数	型		説明
*		→	指定時 list はオブジェクト名 (文字列) 省略時 list はリスト参照番号
list	ListRef, 文字列	→	リストのオブジェクト名 (* 指定時) またはリスト参照番号 (* 省略時)
itemRef | *	倍長整数 , *	→	項目参照番号、または 0 = リストに最後に追加された項目、または * = カレントリスト項目
arrSelectors	テキスト配列	←	パラメーター名配列
arrValues	テキスト配列	←	パラメーター値配列

GET LIST ITEM PARAMETER ARRAYS コマンドは list 引数で指定した参照またはオブジェクト名を持つ階層リスト中で、itemRef で指定した項目に

割り当てられたパラメーター (およびその値) を一回の呼び出しで取り出すことができます。

項目に関連付けられたパラメーターは各項目の追加の情報を格納しています。これらは SET LIST ITEM PARAMETER コマンドで設定できます。

一番目のオプションの引数 * を渡すと、list はフォーム中でリストを表示するリストオブジェクトに対応するオブジェクト名 (文字列) です。この引数を渡さない場合、list は階層リスト参照 (ListRef) です。ひとつのリストオブジェクトだけ、あるいは (二番目の * を使用しないで) 項目を構造的に選択する場合、どちらのシンタックスでも使用できます。しかし同じリストを複数のリストオブジェクトに表示し、かつ (二番目の * を使用して) 現在選択されている項目に対してコマンドを適用する場合、各リストオブジェクトは異なるカレント項目を持つことができるため、オブジェクト名に基づくシンタックスを使用しなければなりません。

GET LIST ITEM PARAMETER ARRAYS は itemRef 項目に設定されたパラメーターを arrSelectors テキスト配列に返します。arrValues テキスト配列が渡された場合、これらのパラメーターに設定された値も返します。

arrValues はテキスト配列でなければなりません。関連付けた値がテキストでない (数値やブール) の場合、文字に変換されます (True="1", False="0")。

- 以下のコードで構築された階層リストがあるとき :

```
<>HL:=New list
$ID:=30
APPEND TO LIST(<>HL;"Martin";$ID)
  //5 つのパラメーター
SET LIST ITEM PARAMETER(<>HL;$ID;"Firstname";"Phil")
SET LIST ITEM PARAMETER(<>HL;$ID;"Birthday";"01/02/1978")
SET LIST ITEM PARAMETER(<>HL;$ID;"Male";True)  // ブール
SET LIST ITEM PARAMETER(<>HL;$ID;"Age";33)  // 数値
SET LIST ITEM PARAMETER(<>HL;$ID;"City";"Dallas")
```

リスト中で "Martin" が選択されているとき、以下のコードを使用してそのパラメーターを取得できます :

```
ARRAY TEXT(arrParamNames; 0)
GET LIST ITEM PARAMETER ARRAYS(*;"<>HL";arrParamNames)
  // arrParamNames{1} = "Firstname"
  // arrParamNames{2} = "Birthday"
  // arrParamNames{3} = "Male"
  // arrParamNames{4} = "Age"
  // arrParamNames{5} = "City"
```

パラメーターの値も取得したい場合、以下のようにします :

```
ARRAY TEXT(arrParamNames; 0)
ARRAY TEXT(arrParamValues; 0)
GET LIST ITEM PARAMETER ARRAYS(*;"<>HL";arrParamNames;
arrParamValues)
   // arrParamValues{1} = "Phil"
   // arrParamValues{2} = "01/02/1978"
   // arrParamValues{3} = "1"
   // arrParamValues{4} = "33"
   // arrParamValues{5} = "Dallas"
```

HTTP クライアント

新しい "HTTP クライアント " テーマには 4D アプリケーションと HTTP サーバーとの通信を可能にするコマンドが含まれます。これらのコマンドを使用すれば (プロクシを経由した場合でも) HTTP プロトコルを使用したデータ交換が容易に行えます。

HTTP AUTHENTICATE

HTTP AUTHENTICATE(name; password{; authMethod}{; *})

引数	型		説明
name	テキスト	→	ユーザー名
password	テキスト	→	ユーザーパスワード
authMethod	倍長整数	→	認証メソッド : 0 または省略時 = 指定しない , 1=BASIC, 2=DIGEST
*	演算子	→	指定時 : プロクシに対する認証

HTTP AUTHENTICATE コマンドを使用して、認証を要求する HTTP サーバーにリクエストを送信することができます。BASIC と DIGEST メソッドがサポートされています。またプロクシに認証情報を送ることもできます。

name と password 引数には識別情報を渡します。この情報はエンコードされ、HTTP Request や HTTP Get コマンドを使用して送信される HTTP リクエストに追加されます。

オプションの authMethod 引数を使用して使用する認証方法を指定できます。"HTTP Client" テーマの以下の定数を使用できます :

定数 (値)	説明

HTTP Basic (1)	BASIC 認証を使用
HTTP Digest (2)	DIGEST 認証を使用

authMethod 引数を省略するか 0 を渡した場合、4D が適切なメソッドを選択するよう指示したことになります。この場合 4D は追加のリクエストを送信して認証方法のネゴシエーションを行います。

* 引数を渡すと HTTP プロクシ用の認証を行います。この設定はクライアントと HTTP サーバーの間に認証を必要とするプロクシが存在する場合に必須となります。HTTP サーバーも認証を要求する場合、二重の認証が必要です。

デフォルトで認証情報はプロセスが終了するまで保持されます。HTTP SET OPTION コマンドを使用して認証情報をリクエストごとに削除するよう設定することができます。

- 認証を行うリクエストの例題 :

```
  // DIGEST モードの認証
HTTP AUTHENTICATE("httpUser";"123";2)
  // プロクシに認証情報を送信する
HTTP AUTHENTICATE("ProxyUser";"456";*)
$httpStatus:=HTTP Get(...)
```

参照 : HTTP SET OPTION

HTTP Get

HTTP Get (url; response {;headerNames; headerValues} {;*}) → 倍長整数

引数	型		説明
url	テキスト	→	リクエスト送信先 URL
response	テキスト | BLOB | ピクチャー	←	リクエストの結果
headerNames	テキスト配列	→ ←	ヘッダー名
headerValues	テキスト配列	→ ←	ヘッダー値
*	演算子	→	指定時 : 接続を保持する (keep-alive) 省略時 : 自動で接続を閉じる
戻り値	倍長整数	←	HTTP ステータスコード

HTTP Get コマンドは指定した URL に HTTP GET リクエストを送信し、HTTP サーバーからのレスポンスを処理します。

url 引数にはリクエストの送信先 URL を渡します。シンタックスは以下の通りです :

```
http://[{user}:[{password}]@]host[:{port}][/{path}][?{queryString}]
```

例えば以下のような文字列を渡せます :

```
http://www.myserver.com
http://www.myserver.com/path
http://www.myserver.com/path?name="jones"
https://www.myserver.com/login (*)
http://123.45.67.89:8083
http://john:smith@123.45.67.89:8083
```

(*) HTTPS リクエストを行う場合でも証明書の発行局は検証されません。

コマンド実行後、response 引数はサーバーから返される結果を受け取ります。この結果はレスポンスからヘッダーを取り除いたボディー部です。response には異なる型の変数を渡すことができます :

- テキスト : 期待される結果がテキストの場合。
- BLOB: 期待される結果がバイナリの場合。
- ピクチャー : 期待される結果がピクチャーの場合。

BLOB を渡した場合、テキストやピクチャー、その他 (.wav, .zip, etc.) どのようなタイプのコンテンツでも受け取ることができます。受け取った内容を復元するのは開発者の仕事です (ヘッダーは BLOB に含まれません)。サーバーから返された結果が response の変数型に対応しない場合、変数は空のままです。

headerNames と headerValues にはリクエストヘッダーの名前と値をそれぞれ格納した配列を渡します。
このコマンド実行後、これらの配列には HTTP サーバーから返されたレスポンスのヘッダー情報で置き換えられます。これにより特に Cookie を管理できます。

* 引数を使用してサーバー接続時に keep-alive メカニズムを有効にできます。デフォルトではこの引数が省略されると、keep-alive は有効になりません。

コマンドからは標準の HTTP ステータスコードが返されます (200=OK 等)。HTTP ステータスコードについては RFC 2616 を参照してください。ネッ トワークに関連する理由 (DNS 解決に失敗した、サーバーに接続できないなど ...) により、サーバーに接続できない場合、コマンドは 0 を返

し、エラーが生成されます。このエラーは ON ERR CALL コマンドを使用してインストールされたエラー処理メソッドで処理できます。

- 4D Web サイトから 4D ロゴを取得する :

```
C_TEXT(URLPic_t)
URLPic_t:="http://www.4d.com/sites/all/themes/dimention/images/
                                                home/logo4D.jpg"
ARRAY TEXT(HeaderNames_at;0)
ARRAY TEXT(HeaderValues_at;0)
C_PICTURE(Pic_i)
$httpResponse:=HTTP Get(URLPic_t;Pic_i;HeaderNames_at;
                                                HeaderValues_at)
```

- RFC を取得する :

```
C_TEXT(URLText_t)
C_TEXT(Text_t)
URLText_t:="http://tools.ietf.org/rfc/rfc1.txt"
ARRAY TEXT(HeaderNames_at;0)
ARRAY TEXT(HeaderValues_at;0)
$httpResponse:=HTTP Get(URLText_t;Text_t;HeaderNames_at;
                                                HeaderValues_at)
```

- ビデオを取得する :

```
C_BLOB(vBlob)
$httpResponse:=HTTP Get("http://www.example.com/video.flv";vBlob)
BLOB TO DOCUMENT(vBlob;"video.flv")
```

参照 : HTTP Request

HTTP GET OPTION

HTTP GET OPTION(option; value) → 倍長整数

引数	型		説明
option	倍長整数	→	取得するオプションのコード
value	倍長整数	←	オプションの現在の値

HTTP GET OPTION コマンドは (HTTP Get や HTTP Request コマンドで実行される次のリクエストでクライアントにより使用される) HTTP オプションの現在値を返します。

現在値はデフォルト設定あるいは HTTP SET OPTION コマンドで設定された値です。

注 設定されるオプションはカレントプロセスで有効です。コンポーネントから呼び出された場合、コンポーネント内で有効となります。

参照 : HTTP SET OPTION

HTTP Request

HTTP Request (httpMethod; url; contents; response {;headerNames; headerValues} {;*}) → 倍長整数

引数	型		説明
httpMethod	テキスト	→	リクエストで使用する HTTP メソッド
url	テキスト	→	リクエストの送信先 URL
contents	テキスト | BLOB | ピクチャー	→	リクエストボディの内容
response	テキスト | BLOB | ピクチャー	←	レスポンスの内容
headerNames	テキスト配列	→ ←	ヘッダーフィールド名
headerValues	テキスト配列	→ ←	ヘッダーフィールド値
*	演算子	→	指定時 : 接続を保持する (keep-alive) 省略時 : 自動で接続を閉じる
戻り値	倍長整数	←	HTTP ステータスコード

HTTP Request コマンドは指定した URL に任意のメソッドで HTTP リクエストを送信し、HTTP サーバーのレスポンスを処理することを可能にします。

httpMethod 引数には HTTP リクエストのメソッドを渡します。HTTP Client テーマの以下の定数を使用できます :

定数 (値)	説明
HTTP DELETE Method (DELETE)	RFC 2616 参照
HTTP GET Method (GET)	RFC 2616 参照。HTTP Get コマンドを使用するのと同じ

HTTP HEAD Method (HEAD)	RFC 2616 参照
HTTP OPTIONS Method (OPTIONS)	RFC 2616 参照
HTTP POST Method (POST)	RFC 2616 参照
HTTP PUT Method (PUT)	RFC 2616 参照
HTTP TRACE Method (TRACE)	RFC 2616 参照

リクエストの送信先を url 引数に渡します。使用されるシンタックスは以下の通りです :

```
http://[{user}:[{password}]@]host[:{port}][/{path}][?{queryString}]
```

例えば以下の文字列を渡すことができます :

```
http://www.myserver.com
http://www.myserver.com/path
http://www.myserver.com/path?name="jones"
https://www.myserver.com/login (*)
http://123.45.67.89:8083
http://john:smith@123.45.67.89:8083
```

(*) HTTPS リクエストを行う場合でも証明書の発行局は検証されません。

contents 引数にはリクエストのボディを渡します。この引数に渡されるデータはリクエストの HTTP メソッドにより異なります。
テキストや BLOB、ピクチャーデータを送信できます。content-type が指定されない場合、以下のタイプが使用されます :

- テキスト : text/plain - UTF8
- BLOBs: application/byte-stream
- ピクチャー : 既知の MIME type (best for Web).

コマンド実行後、response 引数はサーバーから返された結果を受け取ります。この結果はレスポンスからヘッダーを取り除いたボディー部になります。
response には異なる型のレスポンスを渡せます :

- テキスト : 期待される結果がテキストまたは XML ツリー参照の場合。
- BLOB: 期待される結果がバイナリの場合。
- ピクチャー : 期待される結果がピクチャーの場合。

サーバーから返された結果が response の変数型に対応しない場合、変数は空のままで OK システム変数に 0 が設定されます。

headerNames と headerValues にはリクエストヘッダーの名前と値をそれぞれ格納した配列を渡します。
このコマンド実行後、これらの配列には HTTP サーバーから返されたレスポンスのヘッダー情報で置き換えられます。これにより特に Cookie を管理できます。

* 引数を使用してサーバー接続時に keep-alive メカニズムを有効にできます。デフォルトではこの引数が省略されると、keep-alive は有効になりません。

コマンドからは標準の HTTP ステータスコードが返されます (200=OK 等)。HTTP ステータスコードについては RFC 2616 を参照してください。ネットワークに関連する理由 (DNS 解決に失敗した、サーバーに接続できないなど ...) により、サーバーに接続できない場合、コマンドは 0 を返し、エラーが生成されます。このエラーは ON ERR CALL コマンドを使用してインストールされたエラー処理メソッドで処理できます。

- リモートデータベースからレコードを削除する :

```
C_TEXT($response)
$body_t:="{record_id:25}"
$httpStatus_l:=HTTP Request(HTTP DELETE Method;"database.example.com";$body_t;$response)
```

- リモートデータベースにレコードを追加する :

```
C_TEXT($response)
$body_t:="{fName:'john',fName:'Doe'}"
$httpStatus_l:=HTTP Request(HTTP PUT Method;"database.example.com";$body_t;$response)
```

参照 : HTTP Get

HTTP SET OPTION

HTTP SET OPTION(option; value)

引数	型		説明
option	倍長整数	→	設定するオプションのコード
value	倍長整数	→	オプションの値

HTTP SET OPTION コマンドは (HTTP Get や HTTP Request コマンドで実行される次のリクエストでクライアントにより使用される) HTTP オプションの値を設定します。設定するオプションの数だけこのコマンドを呼び出します。

注 設定されるオプションはカレントプロセスに対し有効です。コンポーネントの場合そのコンポーネント内で有効です。

option 引数には設定するオプションの番号を、value 引数にはオプションの新しい値を渡します。option 引数には HTTP Client テーマの以下の定義済み変数のいずれかを指定できます :

定数 (値)	説明
HTTP Timeout (1)	値 = クライアントリクエストのタイムアウト (秒単位)。このタイムアウトはサーバーからのレスポンスを何秒待つかを指定します。タイムアウト時間を経過するとクライアントはセッションを閉じ、リクエストは失われます。 デフォルトでタイムアウトは 20 秒に設定されています。ネットワークの状態やリクエストの特性に応じてこの値を変更できます。
HTTP Follow redirect (2)	値 = 0 (リダイレクトを許可しない) または 1 (リダイレクトを許可する) デフォルト値 = 1
HTTP Max redirect (3)	値 = 受け入れるリダイレクト数の最大値 デフォルト値 = 2
HTTP Display auth dial (4)	値 = 0 (ダイアログボックスを表示しない) または 1 (ダイアログボックスを表示する)。デフォルト値 : 0 このオプションを使用して、HTTP Get や HTTP Request を実行した際に認証ダイアログボックスを表示するかどうかを指定できます。デフォルトで認証ダイアログボックスは表示されず、HTTP AUTHENTICATE コマンドを使用して認証を行います。しかし認証ダイアログを表示してユーザーに認証 ID を入力させたい場合、value に 1 を渡します。ダイアログボックスは認証が必要な場合のみ表示されます。

HTTP Reset auth settings (5)	値 = 0 (情報を削除しない) または 1 (情報を削除する)。デフォルト値 : 0 このオプションを使用して 4D に、各 HTTP Get や HTTP Request コマンド実行毎に、ユーザーの認証情報 (ユーザー名、パスワード、認証メソッド) をリセットするよう指示できます。デフォルトでこれらの情報は保持され、各リクエストで再利用されます。value に 1 を渡すとコマンド実行毎にこれらの情報は削除されます。この設定に関わらず、これらの情報はプロセスが終了すると削除されます。
HTTP Compression (6)	値 = 0 (圧縮しない) または 1 (圧縮する)。デフォルト値 : 1 このオプションを使用して、クライアント / サーバー間通信を効率化するための圧縮メカニズムの有効 / 無効を切り替えることができます。このメカニズムが有効になっていると、HTTP クライアントはサーバーのレスポンスに応じて deflate または gzip 圧縮を使用します。

オプションはどのような順番で設定してもかまいません。おなじオプションを複数回設定した場合、最後に設定された値が使用されます。

参照 : HTTP GET OPTION

リストボックス

リストボックステーマのコマンドに数多くの変更が加えられました。新しいコマンドが追加され、また既存のコマンドもいくつか変更されました。

注 新しい **On Delete Action** がリストボックスでも利用可能です (110 ページの "Form event" 参照)。

新しいコマンド

LISTBOX Get column formula

LISTBOX Get column formula ({*; }object) → 文字列

引数	型	説明

*		→	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)
戻り値	文字列	←	列に割り当てられたフォーミュラ

LISTBOX Get column formula コマンドは object と * 引数で指定したリストボックス列に割り当てられたフォーミュラを返します。フォーミュラはリストボックスプロパティのデータソースが**カレントセレクション**または**命名セレクション**の場合のみ使用できます。列にフォーミュラが割り当てられていない場合、コマンドは空の文字列を返します。

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。
この引数にはリストボックス列を指定しなければなりません。

参照 : LISTBOX SET COLUMN FORMULA

LISTBOX Get footer calculation

LISTBOX Get footer calculation({*; }object) → 倍長整数

引数	型		説明
*		→	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)
戻り値	倍長整数	←	計算のタイプ

LISTBOX Get footer calculation コマンドは object と * 引数で指定したリストボックスのフッターエリアに割り当てられた自動計算のタイプを返します。

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。

object 引数を使用して以下を指定します :

- フッターエリアの変数またはオブジェクト名。この場合コマンドはこのエリアに割り当てられた自動計算のタイプを返します。

- リストボックス列の変数またはオブジェクト名。この場合コマンドはこの列のフッターエリアに割り当てられた自動計算のタイプを返します。

返される値を Listbox footer calculation テーマの定数と比較することができます (132 ページの "LISTBOX SET FOOTER CALCULATION" 参照)。

参照 : LISTBOX SET FOOTER CALCULATION

LISTBOX Get footers height

LISTBOX Get footers height ({*; }object{; unit}) → 倍長整数

引数	型		説明
*		→	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)
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
戻り値	倍長整数	←	行の高さ

LISTBOX Get footers height コマンドは object と * 引数で指定したリストボックスのフッター行の高さを返します。

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のフッターいずれかを指定できます。

unit 引数を省略した場合、デフォルトで高さはピクセル単位で返されます。異なる単位を指定するには unit 引数に List box テーマの以下の定数を渡します :

単位 (値)	説明
Listbox pixels (0)	高さをピクセルで指定 (デフォルト)。
Listbox lines (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。

行の高さの計算に関する詳細は 31 ページの " 行、ヘッダー、フッターの高さ " を参照してください。

参照 : LISTBOX SET FOOTERS HEIGHT

LISTBOX GET GRID

LISTBOX GET GRID ({* ;} object; horizontal; vertical)

引数	型		説明
*		→	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)
horizontal	ブール変数	←	True: 表示、False: 非表示
vertical	ブール変数	←	True: 表示、False: 非表示

LISTBOX GET GRID コマンドは object と * 引数で指定したリストボックスオブジェクトの縦横グリッド線の表示 / 非表示状態をそれぞれ返します。

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。

コマンドは horizontal と vertical に、対応する線が表示されているか (True)、いないか (False) を返します。デフォルトでグリッドは表示されています。

LISTBOX GET GRID COLORS

LISTBOX GET GRID COLORS ({* ;} object; hColor; vColor)

引数	型		説明
*		→	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)
hColor	倍長整数	←	横グリッドの RGB カラー値
vColor	倍長整数	←	縦グリッドの RGB カラー値

LISTBOX GET GRID COLORS コマンドは object と * 引数で指定したリストボックスの縦横グリッドカラーをそれぞれ返します。

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。

コマンドは hColor と vColor に RGB カラーの値を返します。RGB カラーに関する詳細は OBJECT SET RGB COLORS コマンドの説明を参照してください。

LISTBOX Get headers height

LISTBOX Get headers height ({*; }object{; unit}) → 倍長整数

引数	型		説明
*		→	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)
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
戻り値	倍長整数	←	行の高さ

LISTBOX Get 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返します。

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のヘッダーいずれかを指定できます。

unit 引数を省略した場合、デフォルトで高さはピクセル単位で返されます。異なる単位を指定するには unit 引数に List box テーマの以下の定数を渡します :

unit (値)	説明
Listbox pixels (0)	高さをピクセルで指定 (デフォルト)。
Listbox lines (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。

行の高さの計算に関する詳細は 31 ページの " 行、ヘッダー、フッターの高さ " を参照してください。

参照 : LISTBOX SET HEADERS HEIGHT

LISTBOX Get locked columns

LISTBOX Get locked columns ({*; }object) → 倍長整数

引数	型		説明
*		→	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)
戻り値	倍長整数	←	横スクロールしない列の数

LISTBOX Get locked columns コマンドは object と * 引数で指定したリストボックスで横スクロールしない列の数を返します。

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。

プロパティリストや LISTBOX SET LOCKED COLUMNS コマンドを使用して横スクロールしない列数を設定できます。詳細は 34 ページの " 横スクロールしない列 " を参照してください。

この設定がされたエリア内でプログラムを使用して列が挿入されたり削除されたりすると、このコマンドから返される列数も変更されます。例えば横スクロールしない列をひとつ削除すると、コマンドから返される列数はひとつ減ります。同様にこのエリアにプログラムで列を挿入すると、この列は自動でロックされ、ロックされた列数は一つ増えます。

しかしコマンドは列の表示 / 非表示状態は考慮しません。

参照 : LISTBOX SET LOCKED COLUMNS

LISTBOX Get static columns

LISTBOX Get static columns ({*; }object) → 倍長整数

引数	型		説明
*		→	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)

戻り値　　倍長整数　←　ドラッグで移動しない列数

LISTBOX Get static columns コマンドは object と * 引数で指定したリストボックス中のドラッグで移動しない列数を返します。

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。

スタティック列はプロパティリストや LISTBOX SET STATIC COLUMNS コマンドを使用して設定できます。

スタティック列内でプログラムを使用して列が挿入されたり削除されたりすると、このコマンドから返される列数も変更されます。
しかしコマンドは列の表示 / 非表示状態は考慮に入れません。

注　ドラッグで移動しない列と横スクロールしない列は異なる機能です。詳細は 34 ページの " 横スクロールしない列 " を参照してください。

参照 : LISTBOX SET STATIC COLUMNS

LISTBOX SET COLUMN FORMULA

LISTBOX SET COLUMN FORMULA ({*; }object; formula; dataType)

引数	型		説明
*		→	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)
formula	文字列	→	列に割り当てる 4D フォーミュラ
dataType	倍長整数	→	フォーミュラの結果型

LISTBOX SET COLUMN FORMULA コマンドは object と * 引数で指定したリストボックス列に割り当てられた formula を変更します。フォーミュラはリストボックスプロパティのデータソースが**カレントセレクション**あるいは**命名セレクション**の場合のみ使用できます。

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。
この引数はリストボックス列を指定しなければなりません。

formula には以下のような、有効な式を指定できます :

- インストラクション
- フォーミュラエディターで生成したフォーミュラ
- 4D コマンドの呼び出し
- プロジェクトメソッドの呼び出し

コマンドが呼び出されるとフォーミュラが解析され実行されます。

注 アプリケーションランゲージに左右されない、4D コマンドを呼び出すフォーミュラを指定するためには **Command Name** を使用します。

dataType 引数にはフォーミュラ実行時に返されるデータの型を指定します。この引数には Field and Variable Types テーマ内の定数を渡します。フォーミュラの結果が期待するデータ型に一致しない場合、エラーが生成されます。

参照 : LISTBOX Get column formula

LISTBOX SET FOOTER CALCULATION

LISTBOX SET FOOTER CALCULATION ({*; }object; calculation)

引数	型		説明
*		→	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)
calculation	倍長整数	→	フッターエリアの計算タイプ

LISTBOX SET FOOTER CALCULATION コマンドは object と * 引数で指定したリストボックスのフッターに割り当てる自動計算を設定します。

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。

object 引数では以下を指定できます :

- フッターエリアの変数あるいはオブジェクト名。この場合コマンドはこのエリアに適用されます。
- リストボックス列の変数あるいはオブジェクト名。この場合コマンドはこの列のフッターエリアに適用されます。

- リストボックスの変数あるいはオブジェクト名。この場合コマンドはリストボックスのすべてのフッターエリアに適用されます。

calculation 引数には実行する計算式を指定する Listbox footer calculation テーマの定数を渡します :

自動計算 (値)	説明
Listbox footer custom (1)	4D は計算を行いません。フッター変数はプログラムを使用して計算されなければなりません。
Listbox footer min (2)	数値、日付、時間、またはブール型の列に使用します。
Listbox footer max (3)	数値、日付、時間、またはブール型の列に使用します。
Listbox footer sum (4)	数値、ブールまたは時間型の列に使用します。
Listbox footer count (5)	数値、テキスト、日付、時間、ブール、またはピクチャー型の列に使用します。
Listbox footer average(6)	数値または時間型の列に使用します。
Listbox footer std deviation (7)	数値または時間型の列に使用します (配列タイプのリストボックスのみ)。
Listbox footer variance (8)	数値または時間型の列に使用します (配列タイプのリストボックスのみ)。
Listbox footer sum squares (9)	数値または時間型の列に使用します (配列タイプのリストボックスのみ)。

定義済み計算式では列のすべての値が計算対象になることに留意してください。これには非表示行も含まれます。表示されている行に計算対象を限定したい場合は Listbox footer custom 定数を使用し、計算をカスタマイズしなければなりません。

列のデータ型や (object にリストボックスを指定した場合) リストボックス中の 1 つの列でも設定した calculation と互換性がない場合、フッターは変更されずエラー 18 が生成されます。
列にフォーミュラが設定されている場合 (セレクションタイプのリストボックス)、エラー 10 が生成されます。

参照 : LISTBOX Get footer calculation

LISTBOX SET FOOTERS HEIGHT

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

引数	型		説明
*		→	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	→	行の高さ
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

LISTBOX SET FOOTERS HEIGHT コマンドは object と * 引数で指定したリストボックスのフッタ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フッターを指定できます。

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単にはピクセルです。単位を変更するには List box テーマの定数を unit 引数に渡します :

unit (値)	説明
Listbox pixels (0)	高さをピクセルで指定 (デフォルト)。
Listbox lines (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。

行の高さの計算に関する詳細は 31 ページの " 行、ヘッダー、フッターの高さ " を参照してください。

参照 : LISTBOX Get footers height

LISTBOX SET HEADERS HEIGHT

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

引数	型		説明
*		→	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

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 :

単位 (値)	説明
Listbox pixels (0)	高さをピクセルで指定 (デフォルト)。
Listbox lines (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。

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。

行の高さの計算に関する詳細は 31 ページの " 行、ヘッダー、フッターの高さ " を参照してください。

参照 : LISTBOX Get headers height

LISTBOX SET LOCKED COLUMNS

LISTBOX SET LOCKED COLUMNS({*; }object; numColumns)

引数	型		説明
*		→	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)
numColumns	倍長整数	→	横スクロールしない列の数

LISTBOX SET LOCKED COLUMNS コマンドは object と * 引数で指定したリストボックスで左端からはじめて numColumns 列をロックします。

ロックされた列はリストボックスの左側に表示され、スクロールされることはありません。詳細は 34 ページの " 横スクロールしない列 " を参照してください。

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。

numColumns には 1 からリストボックスの総列数までの値を渡せます。列のロックを取り除くには numColumns に 0 を渡します。

参照 : LISTBOX Get locked columns

LISTBOX SET STATIC COLUMNS

LISTBOX SET STATIC COLUMNS({*; }object; numColumns)

引数	型		説明
*		→	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)
numColumns	倍長整数	→	ドラッグで移動しない列数

LISTBOX SET STATIC COLUMNS コマンドは object と * 引数で指定したリストボックス中のスタティック列の数を左端から numColumns に設定します。

リストボックス内でスタティック列は移動することができません。この機能は以前のバージョンの 4D に既に存在していましたが、設定はプロパティリストを使用しなければなりませんでした。

注 スタティック列とロックされた列は異なる機能です。詳細は 34 ページの " 横スクロールしない列 " を参照してください。

参照 : LISTBOX Get static columns

変更されたコマンド

LISTBOX DELETE ROWS

LISTBOX DELETE ROWS ({*; }object; vPosition{; numRows})

引数	型		説明
*		→	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)
vPosition	倍長整数	→	削除する行の位置
numRows	倍長整数	→	削除する行数

互換性に関する注意 このコマンドは以前 LISTBOX DELETE ROW という名称でした。

LISTBOX DELETE ROWS コマンドは一回の呼び出しで複数の行を削除できるようになりました。これを行うには numRows 引数に削除する行数を指定するだけです。

numRows が省略された場合デフォルトで 1 行だけが削除されます。

参照 : LISTBOX INSERT ROWS

LISTBOX INSERT COLUMN

LISTBOX INSERT COLUMN ({*; }object; colPosition; colName; colVariable; headerName; headerVar{; footerName; footerVar})

引数	型		説明
*		→	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)
colPosition	倍長整数	→	列を挿入する位置
colName	文字列	→	列オブジェクト名

colVariable	配列 , フィールド , 変数	→	列配列名またはフィールドまたは変数
headerName	文字列	→	列ヘッダーオブジェクト名
headerVar	整数変数	→	列ヘッダー変数
footerName	文字列	→	列フッターオブジェクト名
footerVar	変数	→	列フッター変数

LISTBOX INSERT COLUMN コマンドに 2 つの引数 footerName と footerVar が追加されました。これらの引数は挿入される列のフッターオブジェクト名と変数を設定するために使用されます。

LISTBOX INSERT COLUMN FORMULA

LISTBOX INSERT COLUMN FORMULA({*; }object; colPosition; colName; formula; dataType; headerName; headerVar{; footerName; footerVar})

引数	型		説明
*		→	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)
colPosition	倍長整数	→	挿入する列の位置
colName	文字列	→	列オブジェクト名
formula	文字列	→	列に関連付ける 4D フォーミュラ
dataType	倍長整数	→	フォーミュラの結果型
headerName	文字列	→	列ヘッダーオブジェクト名
headerVar	整数変数	→	列ヘッダー変数
footerName	文字列	→	列フッターオブジェクト名
footerVar	変数	→	列フッター変数

LISTBOX INSERT COLUMN FORMULA コマンドに 2 つの引数 footerName と footerVar が追加されました。これらの引数は挿入される列のフッターオブジェクト名と変数を設定するために使用されます。

LISTBOX INSERT ROWS

LISTBOX INSERT ROWS ({*; }object; vPosition{; numRows})

引数	型		説明
*		→	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)
vPosition	倍長整数	→	行の挿入位置
numRows	倍長整数	→	挿入する行数

互換性に関する注意

このコマンドは以前のバージョンで LISTBOX INSERT ROW という名称でした。

LISTBOX INSERT ROWS コマンドは一回の呼び出しで複数行を挿入できるようになりました。これを行うには numRows 引数に挿入する行数を指定するだけです。

numRows を省略した場合デフォルトで 1 行のみが挿入されます。

参照 : LISTBOX DELETE ROWS

LISTBOX GET ARRAYS

LISTBOX GET ARRAYS ({* ;} object; arrColNames; arrHeaderNames; arrColVars; arrHeaderVars; arrColsVisible; arrStyles {; arrFooterNames; arrFooterVars})

引数	型		説明
*		→	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)
arrColNames	文字列配列	←	列オブジェクト名
arrHeaderNames	文字列配列	←	ヘッダーオブジェクト名
arrColVars	ポインター配列	←	列変数へのポインターまたはフィールドへのポインターまたは Nil

arrHeaderVars	ポインター配列	←	ヘッダー変数へのポインター
arrColsVisible	ブール配列	←	各列の表示状態
arrStyles	ポインター配列	←	スタイル、カラー、行表示配列へのポインターまたは Nil
arrFooterNames	文字列配列	←	列フッターオブジェクト名
arrFooterVars	ポインター配列	←	列フッター変数へのポインター

LISTBOX GET ARRAYS コマンドはリストボックスフッターに関連付けられたオブジェクト名と変数を返すようになりました :

- arrFooterNames 配列にはリストボックスの各列フッターのオブジェクト名が返されます。
- arrFooterVars 配列にはリストボックスの各列フッターに関連付けられた変数へのポインターが返されます。

LISTBOX GET CELL POSITION

LISTBOX GET CELL POSITION ({*; }object; column; row{; colVar})

このコマンドを階層リストボックスで生成される以下のフォームイベントで呼び出すことができるようになりました :

- **On Expand**
- **On Collapse**

この点に関する詳細は 37 ページの " 階層リストボックスで使用する新しいイベント " を参照してください。

LISTBOX Get information

LISTBOX Get information ({*; }object; info) → 倍長整数

新しい 2 つの定数を info 引数に指定して、列フッターの表示と高さに関する情報を取得できるようになりました。これらの新しい定数は "List Box" テーマにあります :

info (値)	説明
Listbox display footer (8)	0= 非表示 , 1= 表示
Listbox footer height(9)	ピクセル単位の高さ

LISTBOX Get information(vLB;Listbox footer height) ステートメントはフッターが表示されている場合 LISTBOX Get footers height コマンドと同じ値を返す点に留意してください。しかしフッターが表示されていない場合、LISTBOX Get information は 0 を返しますが、LISTBOX Get footers height は依然として理論的なフッターの高さを返します。

LISTBOX Get rows height

LISTBOX Get rows height ({*; }object{; unit}) → 倍長整数

引数	型		説明
*		→	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)
unit	倍長整数	→	高さの単位 : 0 または省略時 = ピクセル , 1 = 行
戻り値	倍長整数	←	行の高さ

LISTBOX Get rows height コマンドはオプションで unit 引数を受け入れるようになりました。この引数を使用して取得するリストボックスの行の高さの単位を指定できます。"List Box" テーマの以下のいずれかの定数を使用できます :

unit (値)	説明
Listbox pixels (0)	ピクセル単位 (デフォルト)
Listbox lines (1)	行単位。4D は指定されたフォント属性に基づき行の高さを計算する。

この引数を省略した場合、以前のバージョンと同様ピクセル単位の高さが返されます。

参照 : LISTBOX SET COLUMN FORMULA

LISTBOX GET TABLE SOURCE

LISTBOX GET TABLE SOURCE ({* ;} object; tableNum{; name{; highlightName}})

引数	型		説明
*		→	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)
tableNum	倍長整数	←	指定されたテーブル番号
name	文字列	←	設定された命名セレクション名、カレントセレクションの場合空の文字列
highlightName	文字列	←	ハイライトセット名

LISTBOX GET TABLE SOURCE コマンドはリストボックスに関連付けられたハイライトセット名を highlightName 引数に返すようになりました。ハイライトセットが指定されていない場合、空の文字列が返されます。

参照 : LISTBOX SET TABLE SOURCE

LISTBOX SET ROWS HEIGHT

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

引数	型		説明
*		→	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	→	行の高さ
unit	倍長整数	→	高さの単位 : 0 または省略時 = ピクセル , 1 = 行

LISTBOX SET ROWS HEIGHT コマンドはオプションの引数 unit を受け入れるようになりました。この引数を使用して設定する行の高さの単位を指定できます。"List Box" テーマの以下のいずれかの定数を指定できます :

unit (値)	説明
Listbox pixels (0)	ピクセル単位 (デフォルト)
Listbox lines (1)	行単位。4D は指定されたフォント属性に基づき行の高さを計算する。

この引数を省略した場合、以前のバージョンと同様ピクセル単位で高さを指定します。

参照 : LISTBOX Get rows height

LISTBOX SET TABLE SOURCE

LISTBOX SET TABLE SOURCE ({* ;} object; tableNum | name {; highlightName}})

引数	型		説明
*		→	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)
tableNum | name	倍長整数 , 文字列	→	使用するカレントセレクションが属するテーブル番号または使用する命名セレクション名
highlightName	文字列	→	ハイライトセット名

LISTBOX SET TABLE SOURCE コマンドは highlightName 引数に、リストボックスに割り当てるハイライトセット名を受け入れるようになりました。ハイライトセットはリストボックス上でユーザーがハイライトしたレコードを管理するために使用されます。

参照 : LISTBOX GET TABLE SOURCE

名称変更されたコマンド

以下のコマンドは 4D v13 で名称が変更されました :

以前の名称	v13 の新しい名称
LISTBOX SHOW GRID	**LISTBOX SET GRID**
LISTBOX INSERT ROW	LISTBOX INSERT ROWS
LISTBOX DELETE ROW	LISTBOX DELETE ROWS

名称変更された定数

"List box" テーマの以下の定数は 4D v13 で名称が変更されました :

以前の名称	v13 の新しい名称
Add to listbox selection	**Listbox add to selection**
Display listbox header	**Listbox display header**
Display listbox hor scrollbar	**Listbox display hor scrollbar**
Display listbox ver scrollbar	**Listbox display ver scrollbar**
Position listbox hor scrollbar	**Listbox hor scrollbar position**
Position listbox ver scrollbar	**Listbox ver scrollbar position**

Remove from listbox selection	**Listbox remove from selection**
Replace listbox selection	**Listbox replace selection**

オブジェクトプロパティ

OBJECT Get auto spellcheck

OBJECT Get auto spellcheck({*; }object) → ブール

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
戻り値	ブール	←	True = 自動スペルチェック False = 自動スペルチェックなし

OBJECT Get auto spellcheck コマンドは object と * 引数で指定したオブジェクトのカレントプロセスの**自動スペルチェック**オプションに関する設定値を返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドは自動スペルチェックが object に対し有効になっていれば True、そうでなければ False を返します。

参照 : OBJECT SET AUTO SPELLCHECK

OBJECT GET DRAG AND DROP OPTIONS

OBJECT GET DRAG AND DROP OPTIONS ({*; }object; draggable; automaticDrag; droppable; automaticDrop)

引数	型		説明
*		→	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)
draggable	倍長整数	←	0 = False, 1 = True
automaticDrag	倍長整数	←	0 = False, 1 = True
droppable	倍長整数	←	0 = False, 1 = True
automaticDrop	倍長整数	←	0 = False, 1 = True

OBJECT GET DRAG AND DROP OPTIONS コマンドは object と * 引数で指定したオブジェクトのカレントプロセスのドラッグ& ドロップオプションを返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドはデザインモードや OBJECT SET DRAG AND DROP OPTIONS コマンドを使用してカレントプロセスに設定された、現在のドラッグ& ドロップオプションを返します。

各引数には対応するオプションが有効 (1) か無効 (0) かに基づき、1 または 0 が返されます。返された値を Object Properties テーマの以下の定数と比較することもできます :

定数 (値)	**説明**
Draggable True (1)	プログラムモードでのドラッグが有効。
Draggable False (0)	プログラムモードでのドラッグが無効。
Draggable auto True (1)	自動モードでのドラッグが有効。
Draggable auto False (0)	自動モードでのドラッグが無効。
Droppable True (1)	プログラムモードでのドロップを受け入れる。
Droppable False (0)	プログラムモードでのドロップを受け入れない。
Droppable auto True (1)	自動モードでのドロップを受け入れる。
Droppable auto False (0)	自動モードでのドロップを受け入れない。

参照 : OBJECT SET DRAG AND DROP OPTIONS

OBJECT Get focus rectangle invisible

OBJECT Get focus rectangle invisible({*; }object) → ブール

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
戻り値	倍長整数	←	True = フォーカスの四角を隠す False = フォーカスの四角を表示する

OBJECT Get focus rectangle invisible コマンドは object と * 引数で指定したオブジェクトのフォーカスの四角に関するカレントプロセス内の表示オプションを返します。この設定は、フォームエディターのプロパティリストの入力可能オブジェクトで利用できる**フォーカスの四角を隠す**オプションに対応します。このコマンドはデザインモードや OBJECT SET FOCUS RECTANGLE INVISIBLE コマンドを使用して指定されたこのオプションに関する現在の設定値を返します。

注 このオプションは Mac OS でのみ利用できます。Windows では効果がありません。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドはフォーカスの四角が隠されていると True を、表示されていると False を返します。

参照 : OBJECT SET FOCUS RECTANGLE INVISIBLE

OBJECT Get help tip

OBJECT Get help tip ({*; }object) → テキスト

引数	型		説明
*		→	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)

戻り値		←	テキスト　オブジェクトのヘルプメッセージ

OBJECT Get help tip コマンドは object と * 引数で指定したオブジェクトに割り当てられたカレントプロセスのヘルプメッセージを返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドは指定したオブジェクトにデザインモードあるいは OBJECT SET HELP TIP コマンドでカレントプロセスに関連付けられた、現在のヘルプメッセージを返します。返される文字列はフォームが実行されたときに表示されるメッセージです。4D 参照や xliff resname などの可変な値が含まれる場合、文脈に基づき解析された結果が返されます。

- ピクチャーボタンのタイトルがヘルプメッセージとして格納されています。このタイトルは xliff ファイルに記述されていて、アプリケーションのカレント言語に基づき変わります :

```
OBJECT SET HELP TIP(*;"button1";":xliff:btn_discover")
$helpmessage:=OBJECT Get help tip (*;"button1")
  // $helpmessage には例えば日本語環境では " はい "、
  // 英語環境では "YES" が返されます。
  //(xliff にそのように記述されていれば )
```

参照 : OBJECT SET HELP TIP

OBJECT Get keyboard layout

OBJECT Get keyboard layout({*; }object) → 文字列

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
戻り値	文字列	←	レイアウトのランゲージコード "" = レイアウトなし

OBJECT Get keyboard layout コマンドは object と * 引数で指定したオブジェクトにカレントプロセスで割り当てられたカレントキーボードレイアウトを返します。

注 このプロパティはデータ入力中に使用されるランゲージスクリプトを強制します。これは ASCII 互換モードでのみ利用可能であり、Unicode モードでは無視されます。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドは使用する言語の (RFC3066、ISO639、そして ISO3166 に基づく) コードを示す文字列を返します。詳細は SET DATABASE LOCALIZATION コマンドの説明を参照してください。

参照 : OBJECT SET KEYBOARD LAYOUT

OBJECT GET RESIZING OPTIONS

OBJECT GET RESIZING OPTIONS ({*; }object; horizontal; vertical)

引数	型		説明
*		→	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)
horizontal	倍長整数	←	横リサイズオプション
vertical	倍長整数	←	縦リサイズオプション

OBJECT GET RESIZING OPTIONS コマンドは object と * 引数で指定したオブジェクトのカレントプロセスのリサイズオプションを返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

このコマンドはデザインモードや OBJECT SET RESIZING OPTIONS コマンドを使用してプロセスに設定された、リサイズに関するオプションの現在の設定値を返します。これらのオプションはフォームウィンドウのサイズが変更されたときのオブジェクトの表示方法を規定します。

horizontal 引数には横方向のリサイズオプション設定値が返されます。この値を Object Properties テーマの以下の定数と比較できます :

定数 (値)	説明

Resize horizontal grow (1)	ウィンドウが横方向に拡げられたら、オブジェクトの幅も同じ比率だけ右に拡大する。
Resize horizontal move (2)	ウィンドウの幅が拡げられたら、オブジェクトも同じだけ右方向に移動する。
Resize horizontal none (0)	ウィンドウの幅が変更されても、オブジェクトの位置及びサイズを変更しない。

vertical 引数には縦方向のリサイズオプション設定値が返されます。この値を Object Properties テーマの以下の定数と比較できます :

定数 (値)	説明
Resize vertical grow (1)	ウィンドウが縦方向に拡げられたら、オブジェクトの高さも同じ比率だけ下に拡大する。
Resize vertical move (2)	ウィンドウの高さが拡げられたら、オブジェクトも同じだけ下方向に移動する。
Resize vertical none (0)	ウィンドウの高さが変更されても、オブジェクトの位置及びサイズを変更しない。

参照 : OBJECT SET RESIZING OPTIONS

OBJECT GET SHORTCUT

OBJECT GET SHORTCUT({*; }object; key{; modifiers})

引数	型		説明
*		→	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)
key	文字列	←	オブジェクトに割り当てられたキー
modifiers	倍長整数	←	モディファイアーキーマスクまたはマスクの組み合わせ

OBJECT GET SHORTCUT コマンドは object と * 引数で指定されたオブジェクトに割り当てられたカレントプロセスのキーボードショートカットを返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

key 引数にはそのオブジェクトに割り当てられたキー (標準キーの場合)、またはキーの値を示すブラケットで囲まれた文字列 (ファンクションキーの場合) が返されます。この値を Shortcut and Associated Keys テーマの定数と比較できます (OBJECT SET SHORTCUT コマンド参照)。

modifiers 引数にはそのショートカットに関連付けられたモディファイアーキーを表す値が返されます。複数のモディファイアーキーが組み合わせて指定されている場合、コマンドはその合計値を返します。この値を Events (Modifiers) テーマの以下の定数と比較できます ::

定数 (値)	説明
Shift key mask (512)	Windows および Mac OS
Command key mask (256)	Windows = Ctrl キー、Mac OS = Command キー
Option key mask (2048)	Windows = Alt キー、Mac OS = Option キー
Control key mask (4096)	Mac OS のみ

ショートカットにモディファイアーキーが関連付けられていない場合、modifiers には 0 が返されます。

注 object 引数がフォーム中の複数のオブジェクトに該当し、それらに異なる値が設定されている場合、key には空の文字列 ("") が、そして modifiers には 0 が返されます。

参照 : OBJECT SET SHORTCUT

OBJECT GET SUBFORM

OBJECT GET SUBFORM({*; }object; tablePtr; detailSubform{; listSubform})

引数	型		説明
*		→	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)

tablePtr	Table	←	フォームが属するテーブルへのポインター
detailSubform	テキスト	←	サブフォームの詳細フォーム名
listSubform	テキスト	←	サブフォームのリストフォーム名 (テーブルフォーム)

OBJECT GET SUBFORM コマンドは object と * 引数で指定したサブフォームオブジェクトに関連付けられたフォームの名前を返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

tablePtr 引数には使用されるフォームが属するテーブルへのポインターが返されます。サブフォームにプロジェクトフォームが指定されている場合、この引数には Nil になります。

detailSubform 引数にはサブフォームで使用されている詳細フォーム名が返されます。

listSubform 引数にはサブフォームで使用されているリストフォーム名が返されます。リストフォームが指定されていない場合空の文字列が返されます。

参照 : OBJECT SET SUBFORM, OBJECT GET SUBFORM CONTAINER SIZE

OBJECT GET SUBFORM CONTAINER SIZE

OBJECT GET SUBFORM CONTAINER SIZE (width; height)

引数	型		説明
width	倍長整数	←	サブフォームオブジェクトの幅
height	倍長整数	←	サブフォームオブジェクトの高さ

OBJECT GET SUBFORM CONTAINER SIZE コマンドは親フォーム中に表示されているカレントサブフォームオブジェクトの幅と高さをピクセル単位で返します。

このコマンドはサブフォームとして使用され、サブフォームオブジェクトの中に表示されているフォームのメソッドから呼び出されなければなりません。コマンドはサブフォームを含んでいるオブジェクトの幅と高さを返します。

このコマンドは例えば、サブフォームオブジェクトの特性に基づきサブフォームオブジェクトのサイズを変更したり移動したりしなければならないようなケースで使用できます。On Load や On Resize フォームイベントでサブフォームはこのコマンドを呼び出し、自身のオブジェクトを表示するためのスペースを計算できます。

- コマンドがサブフォームとして使用されていないフォームから呼び出された場合、コマンドはフォームウィンドウの現在のサイズを返します。
- 画面表示が関連しない状況でコマンドが呼び出された場合 (例えば印刷時)、width と height には 0 が返されます。

参照 : OBJECT SET SUBFORM, OBJECT GET SUBFORM

OBJECT Get vertical alignment

OBJECT Get vertical alignment ({*; }object) → 倍長整数

引数	型		説明
*		→	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)
戻り値	倍長整数	←	行揃えのタイプ

OBJECT Get vertical alignment コマンドは object と * 引数で指定したオブジェクトのテキスト縦位置タイプを示す値を返します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

注 このコマンドの一度の呼び出しを複数のオブジェクトに対して実行した場合、最後のオブジェクトの縦位置値のみが返されます。

返される値は Object Properties テーマの以下のいずれかの定数に対応します :

定数	型	値
Align top	倍長整数	2
Align center	倍長整数	3
Align bottom	倍長整数	4

縦位置は以下のタイプのフォームオブジェクトに設定できます :

- リストボックス
- リストボックス列
- リストボックスヘッダーおよびフッター

参照 : OBJECT SET VERTICAL ALIGNMENT

OBJECT SET AUTO SPELLCHECK

OBJECT SET AUTO SPELLCHECK({*; }object; autoSpellcheck)

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
autoSpellcheck	ブール	→	True = 自動スペルチェック False = 自動スペルチェックなし

OBJECT SET AUTO SPELLCHECK コマンドは object と * 引数で指定されたオブジェクトの**自動スペルチェック**オプションをカレントプロセス内で動的に設定します。このオプションを使用して、オブジェクト (テキスト型オブジェクトのみ) にデータが入力される際の自動スペルチェックを有効 / 無効にできます。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

この機能を有効にするには autoSpellcheck に True を、無効にするには False を渡します。

参照 : OBJECT Get auto spellcheck

OBJECT SET DRAG AND DROP OPTIONS

OBJECT SET DRAG AND DROP OPTIONS ({*; }object; draggable; automaticDrag; droppable; automaticDrop)

引数	型		説明
*		→	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)
draggable	倍長整数	→	0 = False, 1 = True
automaticDrag	倍長整数	→	0 = False, 1 = True
droppable	倍長整数	→	0 = False, 1 = True
automaticDrop	倍長整数	→	0 = False, 1 = True

OBJECT SET DRAG AND DROP OPTIONS コマンドは object と * 引数で指定したオブジェクトのドラッグ & ドロップオプションをカレントプロセス内で動的に設定します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

各引数にはそれぞれのオプションを有効にするか (1) 無効にするか (0) を示す値を渡します。Object Properties テーマの以下の定数を使用できます :

- draggable:

定数 (値)	説明
Draggable True (1)	プログラムモードでのドラッグが無効。
Draggable False (0)	プログラムモードでのドラッグが無効。

- automaticDrag (テキストフィールドや変数、コンボボックス、リストボックスでのみ使用できます):

定数 (値)	説明
Draggable auto True (1)	自動モードでのドラッグが有効。
Draggable auto False (0)	自動モードでのドラッグが無効。

- droppable:

定数 (値)	説明
Droppable True (1)	プログラムモードでのドロップを受け入れる。
Droppable False (0)	プログラムモードでのドロップを受け入れない。

- automaticDrop (テキストフィールドや変数、コンボボックス、リストボックスでのみ使用できます):

定数 (値)	説明
Droppable auto True (1)	自動モードでのドロップを受け入れる。
Droppable auto False (0)	自動モードでのドロップを受け入れない。

♦ テキストエリアの自動ドラッグ & ドロップを設定します :

OBJECT SET DRAG AND DROP OPTIONS (*;"Comments";Draggable False; Draggable auto True; Droppable False; Droppable auto True)

参照 : OBJECT GET DRAG AND DROP OPTIONS

OBJECT SET FOCUS RECTANGLE INVISIBLE

OBJET SET FOCUS RECTANGLE INVISIBLE({*; }object; invisible)

引数	型		説明
*		→	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)
invisible	ブール	→	True = フォーカスの四角を隠す False = フォーカスの四角を表示する

OBJECT SET FOCUS RECTANGLE INVISIBLE コマンドは object と * 引数で指定したオブジェクトのフォーカス四角のカレントプロセスの表示オプションを変更します。この設定は、フォームエディターのプロパティリストの入力可能オブジェクトで利用できる**フォーカスの四角を隠す**オプションに対応します。

注 このオプションは Mac OS でのみ利用できます。Windows では効果がありません。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

invisible 引数に True を渡すとフォーカスの四角が隠され、False を渡すと表示されます。

参照 : OBJECT Get focus rectangle invisible

OBJECT SET HELP TIP

OBJECT SET HELP TIP({*; }object; helpTip)

引数	型		説明
*		→	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)
helpTip	テキスト	→	ヘルプメッセージの内容

OBJECT SET HELP TIP コマンドは object と * 引数で指定したオブジェクトに割り当てられたヘルプをカレントプロセス内で動的に変更します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

フォームが実行されると、マウスカーソルがフィールドやオブジェクト上に移動したとき、ヘルプ Tip を表示させることができます。ヘルプ Tip はデザインモードのフォームエディターやストラクチャーエディターでも設定できます。

メッセージとして表示する内容を helpTip 引数に渡します。以下のいずれかを渡すことができます :

- 文字列。例えば " 区切り文字として / を使用 " 等。
- ヘルプ Tip を表示しないようにするには空の文字列 ("")。

参照 : OBJECT Get help tip

OBJECT SET KEYBOARD LAYOUT

OBJECT SET KEYBOARD LAYOUT({*; }object; languageCode)

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
languageCode	文字列	→	RFC3066 ISO639 そして ISO3166 言語コード、"" = 変更しない

OBJECT SET KEYBOARD LAYOUT コマンドは object と * 引数で指定したオブジェクトに割り当てられたキーボードレイアウトをカレントプロセス内で動的に変更します。

注 このプロパティはデータ入力中に使用されるランゲージスクリプトを強制します。これは ASCII 互換モードでのみ利用可能であり、Unicode モードでは無視されます。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

languageCode には使用する言語の (RFC3066、ISO639、そして ISO3166 に基づく) コードを示す文字列を渡します。詳細は SET DATABASE LOCALIZATION コマンドの説明を参照してください。

参照 : OBJECT Get keyboard layout

OBJECT SET RESIZING OPTIONS

OBJECT SET RESIZING OPTIONS({*; }object; horizontal; vertical)

引数	型		説明
*		→	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)
horizontal	倍長整数	→	横リサイズオプション
vertical	倍長整数	→	縦リサイズオプション

OBJECT SET RESIZING OPTIONS コマンドは object と * で指定したオブジェクトのリサイズオプションをカレントプロセス内で動的に変更します。これらのオプションを使用して、フォームウィンドウのサイズが変更されたときにオブジェクトをどのように表示するかを指定できます。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

horizontal 引数には横方向のリサイズオプションを渡します。Object Properties テーマの以下の定数を使用できます ::

定数 (値)	説明

Resize horizontal grow (1)	ウィンドウが横方向に拡げられたら、オブジェクトの幅も同じ比率だけ右に拡大する。
Resize horizontal move (2)	ウィンドウの幅が拡げられたら、オブジェクトも同じだけ右方向に移動する。
Resize horizontal none (0)	ウィンドウの幅が変更されても、オブジェクトの位置及びサイズを変更しない。

vertical 引数には縦方向のリサイズオプションを渡します。Object Properties テーマの以下の定数を使用できます :

定数 (値)	説明
Resize vertical grow (1)	ウィンドウが縦方向に拡げられたら、オブジェクトの高さも同じ比率だけ下に拡大する。
Resize vertical move (2)	ウィンドウの高さが拡げられたら、オブジェクトも同じだけ下方向に移動する。
Resize vertical none (0)	ウィンドウの高さが変更されても、オブジェクトの位置及びサイズを変更しない。

参照 : OBJECT GET RESIZING OPTIONS

OBJECT SET SHORTCUT

OBJECT SET SHORTCUT({*; }object; key{; modifiers})

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字列)、省略時 object は変数またはフィールド
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) 変数またはフィールド (* 省略時)
key	文字列	→	オブジェクトに割り当てるキー
modifiers	倍長整数	→	モディファイアーキーマスクまたはマスクの組み合わせ

OBJECT SET SHORTCUT コマンドは object と * で指定したオブジェクトのキーボードショートカットをカレントプロセスで動的に変更します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

key 引数にはオブジェクトに関連付けるキーを指定する文字列を渡します。以下のいずれかを渡すことができます :

- "B" のような標準のキーの名前
- Shortcut and Associated Keys テーマの定数やその値 :

定数	型	値
Shortcut with F1	文字列	[F1]
Shortcut with F2	文字列	[F2]
Shortcut with F3	文字列	[F3]
Shortcut with F4	文字列	[F4]
Shortcut with F5	文字列	[F5]
Shortcut with F6	文字列	[F6]
Shortcut with F7	文字列	[F7]
Shortcut with F8	文字列	[F8]
Shortcut with F9	文字列	[F9]
Shortcut with F10	文字列	[F10]
Shortcut with F11	文字列	[F11]
Shortcut with F12	文字列	[F12]
Shortcut with F13	文字列	[F13]
Shortcut with F14	文字列	[F14]
Shortcut with F15	文字列	[F15]
Shortcut with Carriage Return	文字列	[return]
Shortcut with Enter	文字列	[enter]
Shortcut with Backspace	文字列	[backspace]
Shortcut with Tabulation	文字列	[tab]
Shortcut with Escape	文字列	[esc]

Shortcut with Delete	文字列	[del]
Shortcut with Home	文字列	[home]
Shortcut with End	文字列	[end]
Shortcut with Help	文字列	[help]
Shortcut with Page Up	文字列	[page up]
Shortcut with Page Down	文字列	[page down]
Shortcut with Left Arrow	文字列	[left arrow]
Shortcut with Right Arrow	文字列	[right arrow]
Shortcut with Up Arrow	文字列	[up arrow]
Shortcut with Down Arrow	文字列	[down arrow]

modifiers 引数にはショートカットに割り当てるひとつ以上のモディファイアーキーを渡します。

注 modifiers 引数を省略した場合、設定されたキーが押されるとそのオブジェクトが即座に有効になります。例えばボタンに "H" キーを関連付けた場合、**H** キーを押すとボタンが押されたことになります。このような機能は特定のインターフェースに使用されます。

modifiers 引数を設定するには、Events (Modifiers) テーマのひとつ以上の "mask" タイプ定数を渡します :

定数 (値)	**説明**
Shift key mask (512)	Windows および Mac OS
Command key mask (256)	Windows = Ctrl キー、Mac OS = Command キー
Option key mask (2048)	Windows = Alt キー、Mac OS = Option キー
Control key mask (4096)	Mac OS のみ

♦ アプリケーションの言語に基づき、異なるショートカットを割り当てたいとします。On Load フォームイベントで以下のコードを実行します :

```
Case of
  :(vLang="JA")
    OBJECT SET SHORTCUT(*; "SortButton";"T";Command key mask+Shift
```

```
key mask) // 日本語の場合 Ctrl+Shift+T
   :(vLang="US")
      OBJECT SET SHORTCUT(*; "SortButton";"O";Command key mask+Shift
key mask) // 英語の場合 Ctrl+Shift+O
End case
```

参照 : OBJECT GET SHORTCUT

OBJECT SET SUBFORM

OBJECT SET SUBFORM({*; }object{; aTable}; detailSubform{; listSubform})

引数	型		説明
*		→	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)
aTable	Table	→	フォームのテーブル (テーブルフォームの場合)
detailSubform	テキスト	→	サブフォームの詳細フォーム名
listSubform	テキスト	→	サブフォームのリストフォーム名 (テーブルフォーム)

OBJECT SET SUBFORM コマンドは object と * 引数で指定したサブフォームオブジェクトに割り当てられる詳細フォームおよびオプションでリストフォームを動的に変更します。

注 このコマンドを使用してサブフォームのタイプ (プロパティリストの出力サブフォームプロパティ) を変更することはできません。このプロパティはデザインモードで設定しなければなりません。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

aTable 引数には使用するフォームが属するテーブルを渡します。ページモードのサブフォームとしてプロジェクトフォームを指定する場合、この引数を省略します。

detailSubform 引数には詳細サブフォームとして使用するフォームの名前を渡します。

listSubform 引数にはリストサブフォームとして使用するフォームの名前を渡します。この引数はリストタイプのサブフォームを変更する場合のみ指定できます。

ページモードのサブフォームを変更する場合、このコマンドをいつでも実行できます。カレントセレクションは変更されません。
しかしリストサブフォームを変更する場合、リストを表示しているときのみ変更可能です。リスト中がダブルクリックされ詳細フォームが表示されている状態でこのコマンドを実行すると、エラーが生成されます。

参照 : OBJECT GET SUBFORM

OBJECT SET VERTICAL ALIGNMENT

OBJECT SET VERTICAL ALIGNMENT ({*; }object; alignment)

引数	型		説明
*		→	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)
alignment	倍長整数	→	行揃えコード

OBJECT SET VERTICAL ALIGNMENT コマンドは object と * 引数で指定されたオブジェクトに適用される行揃えのタイプを変更します。

オプションの * 引数を渡すと、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 は変数であり、文字列ではなく変数参照を渡します。

alignment には Object Properties テーマの以下のいづれかの定数を渡します :

定数	型	値
Align default	倍長整数	1
Align top	倍長整数	2
Align center	倍長整数	3
Align bottom	倍長整数	4

行テキスト揃えは以下のオブジェクトに適用できます :

- リストボックス

- リストボックス列
- リストボックスヘッダーやフッター

参照 : OBJECT Get vertical alignment

名称変更されたコマンド

v13 で新規に追加されたコマンドとの整合性を保つため、既存のコマンドの名称が変更されました :

以前のバージョンの名称	v13 の新しい名称
OBJECT Get alignment	**OBJECT Get horizontal alignment**
OBJECT SET ALIGNMENT	**OBJECT SET HORIZONTAL ALIGNMENT**

互換性に関する注意 4D v13 では "Object alignment" 定数テーマが存在しません。このテーマの定数は新しい "Object Properties" にあります。

統計関数

4D v13 では " 統計関数 " テーマのコマンドの能力が拡張され、(二次元配列を含む) 数値型の配列にも適用できるようになりました。

Average

Average (series) → 実数

引数	型		説明
series	フィールド	配列	→ 平均を求めるデータ
戻り値	実数		← series の平均値

Average の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題ではセレクション中トップ 15 の平均点を計算します :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
ORDER BY([Exams];[Exams]Exam_Grade;<)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
ARRAY REAL($ArrGrades;15)
vAverage:=Average($ArrGrades)
```

Max

Max (series) → 実数

引数	型		説明
series	フィールド	配列	→ 最大値を求めるデータ
戻り値	実数		← series 中の最大値

Max の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題では配列中の最も大きな数を取得します :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
vMax:=Max($ArrGrades)
```

Min

Min (series) → 実数

引数	型		説明
series	フィールド	配列	→ 最小値を求めるデータ
戻り値	実数		← series 中の最小値

Min の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題では配列中で最も小さい値を取得します :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
vMin:=Min($ArrGrades)
```

Std deviation

Std deviation (series) → 実数

引数	型		説明
series	フィールド	配列	→ 標準偏差を求めるデータ
戻り値	実数		← series の標準偏差

Std deviation の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題では配列に格納された一連の値の標準偏差を求めます :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
vStdDev:=Std deviation($ArrGrades)
```

Sum

Sum (series) → 実数

引数	型		説明
series	フィールド	配列	→ 合計を求めるデータ
戻り値	実数		← series の合計

Sum の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例では配列中の値の合計値を求めます :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
vSum:=Sum($ArrGrades)
```

Sum squares

Sum squares (series) → 実数

引数	型		説明
series	フィールド	配列	→ 平方和を求めるデータ
戻り値	実数		← series の平方和

Sum squares の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題では配列に格納された一連の値の平方和を取得します :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
```

vSumSquares:=**Sum squares**($ArrGrades)

Variance

Variance (series) → 実数

引数	型		説明
series	フィールド	配列	→ 分散を求めるデータ
戻り値	実数		← series の分散

Variance の series 引数に一次元または二次元の配列を渡せるようになりました。この配列は整数、倍長整数、または実数配列でなければなりません。

- この例題では配列に格納された値の分散を求めます :

```
ARRAY REAL($ArrGrades;0)
QUERY([Exams];[Exams]Exam_Date=!01/07/11!)
SELECTION TO ARRAY([Exams]Exam_Grade;$ArrGrades)
vVariance:=Variance($ArrGrades)
```

ピクチャー

Equal pictures

Equal pictures (picture1; picture2; mask) → ブール

引数	型		説明
picture1	ピクチャー	→	元のソースピクチャー
picture2	ピクチャー	→	比較するピクチャー
mask	ピクチャー	←	結果のマスク
戻り値	ブール	←	2 つのピクチャーが同じ場合 True、そうでなければ False

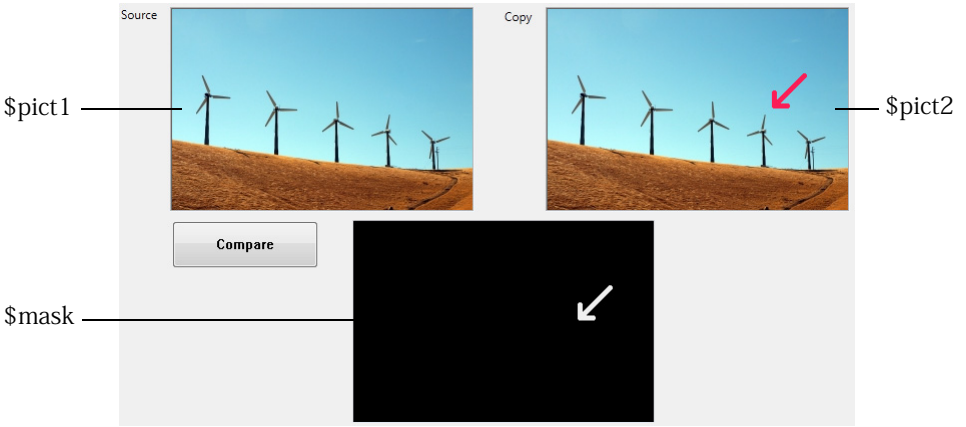
Equal picturess コマンドは 2 つのピクチャーのサイズと内容を厳密に比較します。

picture1 には元のピクチャーを、picture2 には比較したいピクチャーを渡します。

- ピクチャーのサイズが異なる場合、コマンドは False を返し、mask 引数には空のピクチャーが返されます。

- ピクチャーのサイズが同じで内容が異なる場合、コマンドは False を返し、mask 引数には 2 つのピクチャーを比較したピクチャーマスクの結果が返されます。この比較はピクセルごとに行われます。黒の背景上に、一致しないピクセルが白で表されます。
- 両ピクチャーが全く同じ場合、コマンドは True を返し、mask 引数には黒のピクチャーが返されます。

♦ この例題では 2 つのピクチャー (pict1 と pict2) を比較し、結果のマスクを取得します :



以下は比較を実行するコードです :

$equal := **Equal pictures** ($pict1; $pict2; $mask)

Get picture file name

Get picture file name (picture) → テキスト

引数	型		説明
picture	Picture field or variable	→	デフォルト名を取得するピクチャー
戻り値	テキスト	←	ピクチャーファイルのデフォルト名

Get picture file name コマンドは引数に渡されたピクチャーのカレントデフォルト名を返します。

デフォルト名はピクチャーをディスクファイルに書き出す際に使用されます。この名前はピクチャーフィールドや変数にピクチャーファイルを読み込んだ時の元の名前あるいは SET PICTURE FILE NAME コマンドにより設定されているかもしれません。詳細は 20 ページの " ピクチャーファイルのデフォルト名 " を参照してください。

ピクチャーにデフォルト名が設定されていない場合、コマンドは空の文字列を返します。

参照 : SET PICTURE FILE NAME

GET PICTURE KEYWORDS

GET PICTURE KEYWORDS(picture; arrKeywords{; *})

引数	型		説明
picture	ピクチャー	→	割り当てられたキーワードを取得するピクチャー
arrKeywords	テキスト配列	←	キーワードを受け取る配列
*	*	→	指定時 : 重複するキーワードを取り除く

GET PICTURE KEYWORDS コマンドは引数に渡したピクチャーに割り当てられたキーワードのリストを arrKeywords 配列に返します。

IPTC/Keywords メターデータを使用して設定されたキーワードだけが返されます。他のタイプのメタデータはこのコマンドから無視されます。このコマンドは、このタイプのメタデータをサポートするピクチャー (JPEG, TIFF, 等) に対してのみ動作します。

注 4D は IPTC/Keywords タイプのメタデータをインデックスすることが可能です (17 ページの " ピクチャーキーワードのインデックス " 参照)。

* 引数を渡すと、コマンドはキーワードの重複しない値のみを返します。つまりリスト中のすべての値がユニークになります。

ピクチャーに IPTC/Keywords メターデータが含まれない場合、コマンドは空の配列を返し、エラーは生成されません。

注 このコマンドから返される結果はデータベース設定の " 非文字・非数字のみをキーワード区切り文字とする " の現在値により異なります。

SET PICTURE FILE NAME

SET PICTURE FILE NAME(picture; fileName)

引数	型		説明
picture	ピクチャー	→	デフォルト名を設定するピクチャー
fileName	テキスト	→	デフォルトピクチャー名

SET PICTURE FILE NAME コマンドは引数に渡したピクチャーのデフォルト名を設定あるいは変更します。

デフォルト名はピクチャーをディスクファイルに書き出す際のデフォルト名として使用されます。フィールドの内容が変数や他のフィールドにコピーされると、デフォルト名もコピーされます。詳細は 20 ページの " ピクチャーファイルのデフォルト名 " を参照してください。

参照 : Get picture file name

印刷

印刷プレビューの機能を向上させる 2 つの新しいコマンドが追加されました。

バージョン 13 より、4D は Windows で XML Paper Specification (XPS) 技術を使用して印刷可能ドキュメントのプレビューを生成します (Mac OS では OS 組み込みの PDF を使用します)。4D v13 の XPS サポートに関する詳細は 46 ページの "Windows での XPS 印刷プレビュー " を参照してください。

Get print preview

Get print preview → ブール

引数	型		説明
このコマンドは引数を必要としません			
戻り値	ブール	←	True: 印刷プレビューを行う False: 印刷プレビューを行わない

Get print preview コマンドはカレントプロセスで SET PRINT PREVIEW(True) が呼ばれている場合、True を返します。

ユーザーは印刷ダイアログでこのオプションを変更できる点に留意してください。最終的な印刷モードを判定するには Is in print preview コマンドを使用します。

参照 : Is in print preview

Is in print preview

Is in print preview→ ブール

引数	型	説明
このコマンドは引数を必要としません		

戻り値	ブール	←	True: 印刷プレビュー False: 印刷プレビューでない

Is in print preview コマンドは印刷ダイアログボックスで**印刷プレビュー**オプションが選択されている場合 True を返し、そうでなければ False を返します。この設定はカレントプロセスに対し有効です。

Get print preview コマンドをと異なり、Is in print preview はユーザーがダイアログボックスでの設定を終えた後の、オプションの最終的な値を返します。なのでこのコマンドを使用すれば実際に印刷がプレビューモードで行われるのかどうかを判定できます。

- この例題はすべてのタイプの印刷に対応します :

```
SET PRINT PREVIEW(True) // デフォルトで印刷プレビューを行う
PRINT SETTINGS
If (OK=1)
    // ユーザーが印刷先を変更しているかもしれない
  If(Is in print preview)  // 印刷プレビューなら True
    OUTPUT FORM([Invoices] ;"toScreen")
  Else
    OUTPUT FORM([Invoices] ;"toPrinter"
  End if
  PRINT SELECTION ([Invoices])
  If( Is in print preview = False)
    [Invoices]Printed := True
    SAVE RECORD ([Invoices])
    vPrint:= vPrint+1   // 印刷カウンターを更新
  End if
End if
```

参照 : Get print preview

クエリ

GET QUERY DESTINATION

GET QUERY DESTINATION (destinationType; destinationObject{; destinationPtr})

引数	型		説明
destinationType	倍長整数	←	0 = カレントセレクション、1 = セット、2 = 命名セレクション、3 = 変数
destinationObject	文字列	←	セット名、命名セレクション名、または空の文字列

destinationPtr	ポインター	←	destinationType=3 のとき変数へのポインター

GET QUERY DESTINATION コマンドは実行中のプロセスのクエリ結果の格納先を返します。デフォルトでクエリの結果はカレントセレクションとして反映されます。しかしこの動作は SET QUERY DESTINATION コマンドを使用して変更できます。

destinationType 引数にはクエリの格納先を示す値が、そして destinationObject 引数には格納先の名前が返されます (指定されている場合)。destinationType に返される値は Queries テーマの定数と比較できます :

destinationType (値)	説明
Into current selection (0)	destinationObject は空の文字列
Into set (1)	destinationObject にはセット名が返される
Into named selection (2)	destinationObject には命名セレクション名が返される
Into variable (3)	destinationObject i は空の文字列 (destinationPtr 引数を使用します)

クエリの格納先がローカル変数の場合 (destinationType = 3)、コマンドはこの変数へのポインターを destinationPtr 引数に返します。

- 一時的にクエリの格納先を変更し、後で元に戻す :

```
GET QUERY DESTINATION($vType; $vName; $ptr)
  // 現在の設定値を取得
SET QUERY DESTINATION (Into set; "$temp")
  // 一時的に格納先を変更
QUERY(...) // 検索実行
SET QUERY DESTINATION($vType; $vName; $ptr)
  // 元の設定に戻す
```

参照 : SET QUERY DESTINATION

Get query limit

Get query limit → 倍長整数

引数	型		説明
戻り値	倍長整数	←	クエリ結果の制限数値 0 = 制限なし

Get query limit コマンドはカレントプロセスでクエリ結果として返されるレコードの上限値を返します。

この上限値は SET QUERY LIMIT コマンドを使用して設定できます。

デフォルトでは制限がなく、この場合コマンドは 0 を返します。

SET QUERY DESTINATION

SET QUERY DESTINATION (destinationType; {destinationObject{; destinationPtr}})

引数	型		説明
destinationType	倍長整数	→	0= カレントセレクション , 1= セット , 2= 命名セレクション , 3= 変数
destinationObject	文字列 , Variable	→	セット名、命名セレクション名、変数、または空の文字列
destinationPtr	ポインター	→	destinationType=3 のとき変数へのポインター

SET QUERY DESTINATION コマンドは新しい引数 destinationPtr を受け入れるようになりました。この引数を使用してクエリ結果の格納先が変数のとき、異なるシンタックスを使用できるようになります :

SET QUERY DESTINATION (Into variable;""; ->$MyVar)

スペルチェッカー

この新しいテーマには 4D の組み込みスペルチェッカーをプログラムで制御するためのコマンドが集められています。これらのコマンドの名前は "SPELL" から始まります。

注 以前のバージョンでは 2 つの既存のコマンド SET DICTIONARY と SPELL CHECKING が " ツール " テーマにありました。4D v13 では SET DICTIONARY コマンドが SPELL SET CURRENT DICTIONARY に名称変更されました。

Hunspell 辞書のサポート

バージョン 13 より、4D はオープンソースの "Hunspell" 辞書をサポートします。この辞書に関する詳細は以下のサイトを参照してください :
http://hunspell.sourceforge.net/

4D は MySpell や OpenOffice 2.x と同じフォーマットを使用します。つまり同じ名前の .aff と .dic ファイルです。例えば "fr-modern" 辞書は fr-modern.aff と fr-modern.dic ファイルで構成されます。

4D アプリケーションで Hunspell 辞書を使用するには、以下のいずれかの場所に .aff と .dic ファイルをインストールしなければなりません :

- 4D アプリケーション内 : <4D>/Resources/Spellcheck/Hunspell/
- 4D データベース内 : <Database_Files>/Resources/Hunspell

注 4D v13 より、辞書は 4D アプリケーションの 4D Extensions サブフォルダーには保存されなくなりました。

両方の場所とも互換性があります : まずデータベースフォルダーが解析され、次に 4D アプリケーションフォルダー内の辞書が追加されます。同じ名前の辞書が両方の場所にインストールされている場合、データベースフォルダーの辞書が優先されます。この動作により特別な辞書を 4D データベースフォルダーにカプセル化することができます。

Hunspell 辞書は以下からダウンロードできます :
http://wiki.services.openoffice.org/wiki/Dictionaries

ユーザーは既存の辞書あるいは新しく作成した辞書を追加できます。辞書を作成する処理は (以前のバージョンの 4D で使用されていた) Cordial ユーザー辞書と同じです。ユーザー辞書は UTF-8 フォーマットで格納します。

SPELL ADD TO USER DICTIONARY

SPELL ADD TO USER DICTIONARY(words)

引数	型		説明
words	テキスト	テキスト配列	→	ユーザー辞書に追加する単語または単語リスト

SPELL ADD TO USER DICTIONARY コマンドはカレントのユーザー辞書に単語を追加します。

ユーザー辞書は、ユーザーがカレント辞書に追加した単語を含む辞書のことです。この辞書は UserDictionaryxxx.dic という名前のファイルで (xxx はカレント辞書の ID です)、カレントの 4D フォルダーに自動で作成されます。使用される各カレント辞書ごとにユーザー辞書があります。

words には、ユーザー辞書に追加する単語を含む文字あるいは文字配列を渡します。すでに登録されている単語は無視されます。

- ユーザー辞書に特定の名刺を追加する :

```
ARRAY TEXT($arrTwords;0)
APPEND TO ARRAY($arrTwords;"4D")
APPEND TO ARRAY($arrTwords;"Wakanda")
APPEND TO ARRAY($arrTwords;"Clichy")
SPELL ADD TO USER DICTIONARY($arrTwords)
```

参照 : SPELL CHECK TEXT

SPELL CHECK TEXT

SPELL CHECK TEXT(text; errPos; errLength; checkPos; arrSuggest)

引数	型		説明
text	テキスト	→	チェックするテキスト
errPos	倍長整数	←	未知の単語の最初の文字位置
errLength	倍長整数	←	未知の単語の長さ
checkPos	倍長整数	→	チェックを開始する位置
arrSuggest	テキスト配列	←	推奨候補リスト

SPELL CHECK TEXT コマンドは text 引数の内容を、checkPos の位置からチェックし、最初に見つかった未知の単語の位置と長さを返します。

このコマンドはこの未知の単語の最初の文字の位置を errPos に、そしてその長さを errLength に返します。arrSuggest 配列には、スペルチェッカーが提案する修正候補のリストが返されます。

スペルチェックがエラーなしで開始され未知の単語が見つかった場合、OK システム変数に 0 が設定されます。チェック中に初期化エラーが発生するか未知の単語が見つからなかった場合、OK は 1 に設定されます。

- テキスト中にあるかもしれないエラーをカウントします :

```
$pos:=1
$errCount:=0
ARRAY TEXT($tErrors;0)
ARRAY TEXT ($tSuggestions;0)
Repeat
   SPELL CHECK TEXT($myText;$errPos;$errLength;$pos;$tSuggestions)
   If(OK=0)
      $errCount:=$errCount+1 // エラーカウンター
      $errorWord:=Substring($myText;$errPos;$errLength)
      APPEND TO ARRAY($errors;$errorWord) // エラーの配列
```

```
      $pos:=$errPos+$errLength // チェックを続ける
   End if
Until (OK=1)
   // 最終的に $errCount=Size of array($errorWord)
```

参照 : SPELL CHECKING

SPELL Get current dictionary

SPELL Get current dictionary → 倍長整数

引数	型		説明
このコマンドは引数を必要としません			
戻り値	倍長整数	←	スペルチェックに使用される辞書の ID

SPELL Get current dictionary コマンドは使用中の辞書の ID 番号を返します (Cordial または Hunspell)。

- カレント辞書の言語を表示します :

```
   // ロードされた辞書のリスト
SPELL GET DICTIONARY LIST($IDs_al;$Codes_at;$Names_at)
$curLangCode:=SPELL Get current dictionary // 例えば 196608
$countryName:= $Names_at{Find in array($IDs_al;$curLangCode)}
   // メッセージを表示
ALERT("Current dictionary: "+$countryName) // Spanish
```

参照 : SPELL SET CURRENT DICTIONARY

SPELL GET DICTIONARY LIST

SPELL GET DICTIONARY LIST(langID; langFiles; langNames)

引数	型		説明
langID	倍長整数配列	←	言語のユニーク ID
langFiles	テキスト配列	←	インストールされた言語ファイルの名前
langNames	テキスト配列	←	言語のローカル名

SPELL GET DICTIONARY LIST コマンドはマシンにインストールされた辞書ファイルの ID、ファイル名、言語名をそれぞれ langID、langFiles そして langNames 配列に返します。

このコマンドは 4D v13 から使用可能になった Hunspell 辞書 (172 ページの "Hunspell 辞書のサポート " 参照)、およびスペルチェッカー機能を持つすべての 4D バージョンで利用可能な Cordial 辞書の ID リストを返します。

- langID は自動で生成され、SPELL SET CURRENT DICTIONARY コマンド (SET DICTIONARY コマンドの新しい名称) で使用される ID 番号を受け取ります。
ID はユニークで、ファイル名に基づいている点に留意してください。このコマンドは主に開発時に使用します。データベース実行するたびに ID を再生成する必要はありません。
- langFiles はマシンにインストールされた辞書ファイルの名前を受け取ります。Cordial 辞書の場合標準名が返されます (フランス辞書の場合 "fr_FR"、英語辞書の場合 "en_GB" 等)。Hunspell 辞書ではファイル名 (拡張子なし) が返されます。
- langNames はカレントのアプリケーション言語で表現される言語名を受け取ります。例えばフランス語辞書は日本語システムでは " フランス語 (フランス)"、英語システムでは "French (France)" が返されます。
Hunspell 辞書の場合、言語名の後に "- Hunspell" が付加されます。この引数は 4D にとって既知のファイルにのみ有効です。(カスタムファイルなど) 未知のファイルの場合、"N/A - Hunspell" が返されます。(ファイルが有効である限り) これにより辞書の使用が妨げられることはありません。ID 番号を SPELL SET CURRENT DICTIONARY コマンドに渡すことができます。

♦ Hunspell 辞書に "fr-classic+reform1990.aff "、"fr-classic+reform1990.dic"、"fr-dentist.aff " および "fr-dentist.dic" を配置したとします :

```
ARRAY LONGINT($langID;0)
ARRAY TEXT($dicName;0)
ARRAY TEXT($langDesc;0)
SPELL GET DICTIONARY LIST($langID;$dictName;$langDesc)
```

$langID	$dictName	$langDesc
65536	en_GB	English (UK)
65792	en_US	English (USA)
131072	de_DE	German (Germany)
196608	es_ES	Spanish

$langID	$dictName	$langDesc
262144	fr_FR	French (France)
589824	nb_NO	Norwegian Bokmal (Norway)
1074036166	fr-classic+reform1990	French (France) - Hunspell
1073901273	fr-dentist	No description - Hunspell

SPELL SET CURRENT DICTIONARY

SPELL SET CURRENT DICTIONARY {(dictionary)}

引数	型		説明
dictionary	倍長整数 | テキスト	→	ID or Name of dictionary to use If omitted = restore default dictionary

注 以前のバージョンの 4D でこのコマンドは SET DICTIONARY という名称でした。

SPELL SET CURRENT DICTIONARY コマンドのシンタックスが Hunspell 辞書サポートのために変更されました (172 ページの "Hunspell 辞書のサポート " 参照):

- dictionary 引数はオプションになりました。この引数を渡さずにこのコマンドを呼び出すと、デフォルトの辞書が再設定されます。
- dictionary 引数に文字列を渡せるようになりました。辞書ファイル名に対応する Hunspell 辞書を渡すことができます (拡張子なし)。辞書が正しくロードされると OK 変数に 1 が設定されます。

♦ Hunspell フォルダーに配置した "fr-classic" 辞書をロードする :

```
SPELL SET CURRENT DICTIONARY("fr-classic")
  // SPELL SET CURRENT DICTIONARY ("FR-classic.dic") も使用可
```

参照 : SPELL Get current dictionary

文字列

GET TEXT KEYWORDS

GET TEXT KEYWORDS(text; arrKeywords{; *})

引数	型		説明
text	テキスト	→	元のテキスト

arrKeywords	テキスト配列	←	キーワードを受け取る配列
*		→	指定した場合、ユニークキーワード

GET TEXT KEYWORDS コマンドは text を個々の単語に分割し、arrKeywords 配列の要素にして返します。

4D はテキストを個々の単語に分割する際、キーワードインデックスを構築するのと同じアルゴリズムを使用します。これは ICU ライブラリに基づきます。どのようにテキストが単語に分割されるのかに関するより詳しい情報は以下の Web ページを参照してください :
http://www.unicode.org/reports/tr29/#Word_Boundaries

注 ユーザーからのリクエストに基づき、フランス語とイタリア語に関しては例外が設けられました : 母音または "h" が続くアポストロフィ (') は単語区切り文字として扱われます。例えば文字列 "L'homme" や "l'arbre" は "L'"+"homme" や "l'"+"arbre" などのように分割されます。

注 使用されるアルゴリズムはデータベース設定の**非文字・非数字のみをキーワード区切り文字とする**の設定により異なります。

text 引数には単語に分割する元のテキストを渡します。スタイル付きテキストを渡すことができ、この場合スタイルタグは無視されます。

arrKeywords にはテキストから取り出された単語のリストを受信するテキスト配列を渡します。

オプションの * 引数を渡すと、コマンドは arrKeywords 配列内に重複しない値を返します。引数が省略された場合はデフォルトでテキストから取り出された単語がすべて格納され、複数回出現するものについてはその数だけ要素が作成されます。

このコマンドは、4D と同じキーワードを使用しつつ、大量のテキストを含むレコードを検索する、簡素な方法を提供します。例えば "10,000 Jean-Pierre BC45" を含むテキストにおいて、このテキストがキーワード "10,000" + "Jean-Pierre" + "BC45" に分割されるとき、配列には 3 要素が含まれます。プログラムでこの配列をループし、% 演算子を使用してこれらのキーワードを 1 つ以上含むレコードを検索することができます (例題参照)。

- 検索エリアフォームにユーザーは 1 つ以上の単語を入力できます。ユーザーが検索実行を指示したら、プログラムはユーザーが入力した単語を最低 1 つ MyField フィールドに含むレコードを検索します。

```
  // vSearch はフォーム上の検索エリア変数名
GET TEXT KEYWORDS(vSearch;arrSearch;*)
  // ユーザーが同じ単語を複数回入力した場合に備え * を渡す
CREATE SET([MyTable]; "Totalfound")
$n:=Size of array(arrSearch)
For($i;1;$n)
  QUERY([MyTable];[MyTable]MyField % arrSearch{$i})
  CREATE SET([MyTable]; "found")
  UNION("Totalfound";"found";"Totalfound")
End for
USE SET("Totalfound")
```

- 先の例と同じフォームで、ユーザーが入力した単語を MyField フィールドにすべて含むレコードを検索します。

```
  // vSearch はフォーム上の検索エリア変数名
GET TEXT KEYWORDS(vSearch;arrSearch;*)
$n:=Size of array(arrSearch)
QUERY([MyTable];[MyTable]ID >= 0;*)
  // 検索初期化 = all records
For($i;1;$n)
  QUERY([MyTable];&;[MyTable]MyField % arrSearch{$i};*)
    // 検索をスタック
End for
QUERY([MyTable]) // 検索実行
```

- テキスト中の単語をカウント :

```
GET TEXT KEYWORDS(vText;arrWords) // 全単語
$n:=Size of array(arrWords)
GET TEXT KEYWORDS(vText;arrWords;*) // 重複単語を除く
$m:=Size of array(arrWords)
ALERT(" このテキストには "+String($n)+" 個の単語が含まれ、重複を取り除くと "+String($m)+" 個です。")
```

ストラクチャーアクセス

Get external data path

Get external data path (aField) → テキスト

引数	型	説明

aField	テキスト , BLOB, または ピクチャー フィールド	→	外部ストレージの場所を取得するフィールド
戻り値	テキスト	←	外部ストレージファイルのフルパス名

Get external data path コマンドはカレントレコードの、aField 引数に渡したフィールドデータの外部ストレージファイルのフルパス名を返します。aField 引数にはテキスト、BLOB、またはピクチャー型のフィールドを渡さなくてはなりません。

特にこのコマンドを使用して外部ファイルの再コピーが可能となります。

注 v13 の外部ストレージに関しては 21 ページの " データをデータファイル外に保存 " を参照してください。

このコマンドは以下の場合空の文字列を返します :

- フィールドデータが外部ファイルとして格納されていない場合。
- フィールドが Null 値 (外部ファイルが作成されていない) の場合。

参照 : SET EXTERNAL DATA PATH

RELOAD EXTERNAL DATA

RELOAD EXTERNAL DATA (aField)

引数	型		説明
aField	テキスト , BLOB, または ピクチャー フィールド	→	外部ストレージからリロードを行うフィールド

RELOAD EXTERNAL DATA コマンドは BLOB、ピクチャー、およびテキスト型フィールドに割り当てられた外部ストレージの内容をメモリにリロードします。

このコマンドは、フィールドデータがディスクファイルからメモリにロードされているとき、他のアプリケーションによってディスクファイルが更新されたような場合に使用します (外部ストレージファイルは常に書き込み可能です)。例えばピクチャーフィールドに称されるピクチャーは、ピクチャーエディターで更新し保存することが可能です。

このような場合、以下の 2 つの理由でデータを再読み込みしなければなりません :

- フォーム上にフィールドが表示されている場合、その内容を再描画するため。
- レコード保存時に、ディスク上の更新されたピクチャーをメモリ上の古いバージョンで上書きしないため。

注 RELOAD EXTERNAL DATA コマンドは 4D ローカルモードおよび 4D Server 上でのみ動作します。4D リモートモードで個々にフィールドをリロードすることはできません。4D リモートモードの場合 (例えば LOAD RECORD コマンドを使用して) レコード全体をリロードしなければなりません。

SET EXTERNAL DATA PATH

SET EXTERNAL DATA PATH(aField; path)

引数	型		説明
aField	テキスト , BLOB, またはピクチャーフィールド	→	ストレージの場所を設定するフィールド
path	Text | 倍長整数	→	外部ストレージのパス名およびファイル名、または 0 = ストラクチャー定義を使用する 1 = デフォルトフォルダーを使用する

SET EXTERNAL DATA PATH コマンドは aField 引数に渡したフィールドの、カレントレコードの、外部ストレージの場所を設定あるいは変更します。

4D ではテキスト、BLOB、およびピクチャー型のフィールドデータをデータファイルの外部に格納することができます。この機能に関する詳細な説明は 21 ページの " データをデータファイル外に保存 " を参照してください。

このコマンドで指定される設定は、カレントレコードがディスクに保存されるときにのみ適用されます。アプリケーションストラクチャーに設定されたパラメーターは変更されません。カレントレコードがキャンセルされると、コマンドはなにも行いません。

一度コマンドが実行されると、4D は自動でレコードフィールドとディスク上のファイルとのリンクを保守します。パスを変更したい場合を除き、再度コマンドを実行する必要はありません。

path にはカスタムパス名または自動的な場所を指定する定数いずれかを渡すことができます :

- **ファイルへのカスタムパス名**
この場合外部ストレージをカスタムモードで使用することになります。特定の 4D データベース機能はこのモードを自動では利用できません (22 ページの " 自動モード / カスタムモード " 参照)。
データファイルに対する相対パスあるいは絶対パスを指定できます。パスにはストレージファイルのファイル名と拡張子を含めます。拡張子は実際のデータ型と一致しなければなりません (保存時に自動で変換されることはありません)。システムシンタックスを使用しなければなりません。データベース外部ファイル (databaseName.ExternalData) のデフォルトフォルダーを含むどのフォルダーでも指定できます。この場合、これらのファイルはデータベースが保存されるときに含まれます。フォルダーが存在しない場合、4D が自動で作成します (適切な権限がないなどのため作成に失敗した場合エラーが生成されます)。
外部ファイルをデータファイルと同階層か、そのサブフォルダーに保存する場合、4D は指定されたパスがデータファイルに対し相対的であるとみなし、データファイルフォルダーが移動されたり名称変更されたりした場合でもそのリンクを保守します。
これにより複数のレコードで同じ外部ファイルを共有することが可能な点に留意してください。この外部ファイルに対して行われた変更はすべてのレコードに対して有効です。この場合、複数のプロセスが同時に同じフィールドを変更できるならば、セマフォーを使用して同時アクセスを制限しなければなりません。そうでなければ外部ファイルが損傷するリスクがあります。

- **自動的な場所**
Data file maintenance テーマの、以下の 2 つの定数を指定できます :

定数 (値)	説明
Use structure definition (0)	4D はストラクチャーに設定されたフィールドデータの格納設定を使用します (25 ページの " ストラクチャーエディターで外部への格納を設定する " 参照)。外部ストレージから内部ストレージに変更しても、外部ファイルは削除されません。

Use default folder (1)	引数として渡されたフィールドのデータは databaseName.ExternalData という名前のデフォルトフォルダーに保存されます。このフォルダーはデータファイルと同歳層に作成されます。このモードでは、外部データを、それがデータファイル内にあるときと同様に、4D が管理します。

注 SET EXTERNAL DATA PATH コマンドは 4D ローカルモードまたは 4D Server でのみ実行できます。リモートモードの 4D ではなにも行いません。

- ピクチャーデータが特定のサイズ以上の場合、データをデータファイルの外部に保存します。フォームの保存ボタンに以下のように記述します :

```
If (Picture size ([Photos]InputField) > 1300000)
    // 大きなピクチャーを外部ファイルに保存する
  SET EXTERNAL DATA PATH ([Photos]InputField ;
    "C:\\Storage\\LargePicts\\" + String(Record number)
    + ".jpg")
Else
    // 小さなサイズのデータはストラクチャー定義通りに保存する
  SET EXTERNAL DATA PATH ([Photos]InputField ; Use structure definition)
End if
```

参照 : Get external data path

システムドキュメント

COPY DOCUMENT

COPY DOCUMENT (sourceName; destinationName{; *})

引数	型		説明
sourceName	文字列	→	コピーするファイルやフォルダーのパス名
destinationName	文字列	→	ファイルやフォルダーコピーの配置先名またはパス名
*		→	存在する場合、既存のドキュメントを上書きする

既存の COPY DOCUMENT コマンドはファイルだけでなくフォルダーもコピーできるようになりました。このコマンドを使用してフォルダー全体、あるいはフォルダー内のひとつのファイルをコピーできます。

フォルダーを指定するためには、sourceName と destinationName に渡す文字列がプラットフォームに対応したフォルダー区切り文字で終わっていなければなりません。例えば Windows では "C:\\Element\\" はフォルダーを表し、"C:\\Element" はファイルを表します。

フォルダーをコピーするには、その完全パス名を sourceName に渡します。このフォルダーは既に存在しなければなりません。

フォルダーを sourceName 引数に設定した場合、destinationName 引数もフォルダーでなければなりません。ディスクにすでに存在する完全フォルダーパス名を渡さなければなりません。
sourceName で指定したフォルダーと同じ名前のフォルダーが destinationName で指定した場所に既に存在し、そのフォルダーが空でない場合、4D は項目をコピーする前にフォルダーの内容を確認します。オプションの * 引数を渡していない場合、同じ名前のファイルが既に存在すればエラーが生成されます。この引数を渡した場合、その場所のファイルは削除され、コピーするドキュメントで置き換えられます。

ファイルをフォルダーにコピーするために、sourceName にファイルを、destinationName にフォルダーを渡すことができます。

- 指定したフォルダーに同じ名前でファイルをコピー :

  COPY DOCUMENT("C:\\Projects\\DocName";"C:\\Projects\\")

- そ指定したフォルダーに同じ名前でファイルをコピー。既存のファイルは上書きする :

  COPY DOCUMENT("C:\\Projects\\DocName";"C:\\Projects\\"; *)

- フォルダーを他のフォルダーにコピー (両フォルダーは既に存在しなければなりません):

  COPY DOCUMENT("C:\\Projects\\";"C:\\Archives\\2011\\")

CREATE FOLDER

CREATE FOLDER(pathname{; *})

引数	型		説明
pathname	文字列	→	作成する新しいフォルダーのパス名
*		→	パス中に存在しないフォルダーを作成する

既存の CREATE FOLDER コマンドはまだ存在しないフォルダー階層を作成できるようになりました。これを行うには * 引数を渡すだけです。

この場合、folderPath にドキュメントパス名を渡すこともできます。ドキュメント名は無視され、folderPath に指定されたフォルダー階層が再帰的に作成されます。

- "C:\Archives\2011\January\" フォルダー階層を作成 :

```
CREATE FOLDER ("C:\\Archives\\2011\\January\\";*)
```

- 既存の "C:\Archives\" フォルダーに "\February\" サブフォルダーを作成 :

```
CREATE FOLDER ("C:\\Archives\\2011\\February\\Doc.txt";*)
  // "Doc.txt" は無視される
```

DOCUMENT LIST

DOCUMENT LIST (pathname; documents{; options})

引数	型		説明
pathname	テキスト	→	ボリューム、ディレクトリ、またはフォルダーへのパス名
documents	テキスト配列	→	この場所にあるドキュメントの名前
options	倍長整数	→	取得するリストを指定するオプション

既存の DOCUMENT LIST コマンドはオプションの引数 options を受け入れるようになりました。この引数を使用して documents 配列に受け取る情報を指定できます。

options 引数には "System Documents" テーマの以下の新しい定数を指定できます (これらの定数を加算して渡すこともできます):

定数 (値)	説明
Recursive parsing (1)	documents 配列は指定したフォルダーに含まれるすべてのファイルとサブフォルダーを格納
Absolute path(2)	documents 配列は絶対パス名を格納
Posix path (4)	documents 配列は POSIX フォーマットパス名を格納
Ignore invisible (8)	不可視ドキュメントをリストに含めない

注 - 相対モードにおける Recursive parsing オプション (option 1 のみ) では、プラットフォームに応じてサブフォルダー内のドキュメント名は ":" また

は "\" から始まります。
- 相対モードにおける Posix path オプション (option 4 のみ) では、パスは "/" で始まりません。
- 絶対モードにおける Posix path オプション (option 4 + 2) では、パスは常に "/" で始まります。

- フォルダー中のすべてのドキュメントをリスト (以前のシンタックス):

DOCUMENT LIST("C:\";arrFiles)
-> arrFiles:
Text1.txt
Text2.txt

- 絶対モードでフォルダー中のすべてのドキュメントをリスト :

DOCUMENT LIST("C:\";arrFiles; Absolute path)
-> arrFiles:
C:\Text1.txt
C:\Text2.txt

- 再帰 (相対) モードですべてのドキュメントをリスト :

DOCUMENT LIST("C:\";arrFiles;Recursive parsing)
-> arrFiles:
Text1.txt
Text2.txt
\Folder1\Text3.txt
\Folder1\Text4.txt
\Folder2\Text5.txt
\Folder2\Folder3\Picture1.png

- 再帰 (絶対) モードですべてのドキュメントをリスト :

DOCUMENT LIST("C:\MyFolder\";arrFiles;Recursive parsing+Absolute path)
-> arrFiles:
C:\MyFolder\MyText1.txt
C:\MyFolder\MyText2.txt
C:\MyFolder\Folder1\MyText3.txt
C:\MyFolder\Folder1\MyText4.txt
C:\MyFolder\Folder2\MyText5.txt
C:\MyFolder\Folder2\Folder3\MyPicture1.png

- 再帰 POSIX (相対) モードですべてのドキュメントをリスト :

DOCUMENT LIST("C:\MyFolder\";arrFiles;Recursive parsing+Posix path)
-> arrFiles:
MyText1.txt
MyText2.txt
Folder1/MyText3.txt

Folder1/MyText4.txt
Folder2/MyText5.txt
Folder2/Folder3/MyPicture1.png

Select document

Select document (directory ; fileTypes ; title ; options {; selected}) → テキスト

(ファイルを開くダイアログボックスで使用できる詳細機能を指定するための) options 引数に "System Documents" の新しい定数を指定できるようになりました :

File name entry (32)	ユーザーにダイアログへのファイル名の入力を許可する。ファイルは保存されません。ファイルの作成は開発者が行わなくてはなりません。Document システム変数は更新されます。

- ユーザーが指定したカスタムドキュメントを作成する :

```
$doc:= Select document (System folder(Documents folder)+"Report.pdf";"pdf";"Report name:"; File name entry)
If(OK=1)
   BLOB TO DOCUMENT(Document;$blob) // $blob には作成するドキュメントの内容が格納されている
End if
```

名称変更された定数

"System Documents" テーマにおいて、既存の Is a directory 定数が **Is a folder** に名称変更されました。この新しい名前は Mac OS および Windows インターフェースの仕様に沿うものです。

システム環境

LOG EVENT

LOG EVENT ({outputType ;} message {; importance})

outputType 引数に新しい Into 4D Diagnostic Log 定数を使用できるようになりました :

定数 (値)	説明
Into 4D Diagnostic Log (5)	診断ログファイルが有効である場合に、メッセージを 4D の診断ログファイルに記録するよう指示します。

診断ログファイルは SET DATABASE PARAMETER を使用して有効にできます (Get database parameter, SET DATABASE PARAMETER 参照)。

System folder

System folder {(type)} → テキスト

既存の System folder コマンドに 2 つの新機能が追加されました :

- type 引数に使用できる新しい定数 Documents folder が "System Folder" テーマに追加されました :

定数 (値)	説明
Documents folder(17)	ユーザーの "Documents" フォルダー

- Mac OS で以下の定数を指定した場合、新しい値が返されます :
  - User Preferences_User
  - User Preferences_All

今後は ~/Library/Application Support/ フォルダーが返されます (4D Runtime Volume License の場合 ~/Library/Application Support/MyApp/Preferences)。
以前のバージョンではデフォルトで ~/Library/Preferences/ が返されました。

ツール

Generate digest

Generate digest (param ; algorithm) → テキスト

引数	型		説明
param	BLOB | テキスト	→	Digest キーを取得する BLOB やテキスト
algorithm	倍長整数	→	キーの生成に使用するアルゴリズム 0= MD5、1 = SHA1
戻り値	テキスト	←	Digest キーの値

Generate digest コマンドは BLOB やテキストの Digest キーを指定したアルゴリズムで生成して返します。

現在 4D では 2 つのアルゴリズム、**MD5** (Message Digest 5) と **SHA-1** (Secure Hash 1) が利用できます。これらのアルゴリズムは異なるハッシュ関数です :

- MD5 では 16 byte の値が計算され、16 進形式で 32 文字が返されます。
- SHA-1 では 20 byte の値が計算され、16 進形式で 40 文字が返されます。

同じオブジェクトに対してはすべてのプラットフォーム (Mac/Windows, 32/64 bits) で同じ値が返されます。データベースの実行モード (Unicode あるいは非 Unicode) に関わらず、計算は引数として渡されたテキストの UTF-8 表現に対して実行されます。

注 コマンドを空のテキストや BLOB に対して実行すると、計算結果として以下が返されます (エラーにはなりません):
"d41d8cd98f00b204e9800998ecf8427e" (MD5)
"da39a3ee5e6b4b0d3255bfef95601890afd80709" (SHA-1)

param 引数にはテキストまたは BLOB フィールドや変数を渡します。Generate digest コマンドはダイジェストキーを文字列として返します。

algorithm 引数で使用するハッシュ関数を指定します。Digest type テーマの以下の定数を使用できます :

定数 (値)	説明
MD5 Digest (0)	MD5 アルゴリズムを使用
SHA1 Digest (1)	SHA-1 アルゴリズムを使用

ダイジェストキーの計算が失敗した場合コマンドはエラーを生成し、空の文字列を返します。このエラーは ON ERR CALL でインストールされるエラー処理メソッドで処理できます。

注 4D Pack の既存のコマンド AP Get file MD5 digest は廃止されます。今後使用してはいけません。

- この例題では MD5 アルゴリズムを使用して 2 つのドキュメントを比較します :

```
PLATFORM PROPERTIES($Platf;$Syst;$vlMachine)
  // 一番目のドキュメントを読み込みモードで開く
$Same:=True
$vhDocRef1:=Open document("";"*";Read Mode))
If (OK=1)  // ドキュメントが選択されたら
  DOCUMENT TO BLOB(Document;$FirstBlob)  // ドキュメントをロード
  If (OK=1)
    If ($Platf=Mac OS)
      DOCUMENT TO BLOB(Document;$FirstBlobRF;*)
        // Mac OS ではリソースフォークもロード
      $MD5_1RF:=Generate digest($FirstBlobRF;MD5 Digest)
    End if

      // 二番目のドキュメントを読み込みモードで開く
```

```
      $vhDocRef2:=Open document("";"*";Read Mode))
      If (OK=1)
        DOCUMENT TO BLOB(Document;$SecondBlob)
        If (OK=1)
          If ($Platf=Mac OS)
            DOCUMENT TO BLOB(Document;$SecondBlobRF;*)
            $MD5_2RF:=Generate digest($SecondBlobRF;MD5 Digest)
            If ($MD5_1RF#$MD5_2RF) // ダイジェストを比較
                                          $Same:=False
            End if
          End if
          $MD5_1:=Generate digest($FirstBlob;MD5 Digest)
          $MD5_2:=Generate digest($SecondBlob;MD5 Digest)
          If (($MD5_1#$MD5_2) | ($Same=False))
            ALERT(" 異なるドキュメントです ")
          End if
        End if
      End if
    End if
  End if
```

- この例題ではテキストのダイジェストキーを取得します :

```
$key1:=Generate digest("The quick brown fox jumps over the lazy dog.";
                                          MD5 Digest)
  // $key1 は "e4d909c290d0fb1ca068ffaddf22cbd0"
$key2:=Generate digest("The quick brown fox jumps over the lazy dog.";
                                          SHA1 Digest)
  // $key2 は "408d94384216f890ff7a0c3528e8bed1e0b01621"
```

PROCESS 4D TAGS

このコマンドは以前 PROCESS HTML TAGS という名称で、"Web サーバー " テーマにありました。動作に変更はありません。

Web サーバー

新機能、特に自動セッション管理を処理するための新しいコマンドが追加されました。Web サーバーの新機能については Chapter 4 の "Web サーバー " で説明しています。

WEB CLOSE SESSION

WEB CLOSE SESSION (sessionID)

引数	型		説明
sessionID	テキスト	→	セッション UUID

WEB CLOSE SESSION コマンドは sessionID 引数で指定された既存のセッションを破棄します。指定されたセッションが存在しない場合、コマンドはなにも行いません。

Web プロセスや他のプロセスからこのコマンドが呼び出されると :

- ブラウザーに送信される cookie の有効期限が 0 に設定されます。
- 開発者がセッション情報を保存できるようにするために On Web Session Suspend データベースメソッドが呼び出されます。
- カレントセレクションや変数などのプロセスオブジェクトが消去され、レコードのロックが解除されます。

このコマンド実行後、Web クライアントが当該 cookie を使用して 4D Web サーバーにアクセスすると、新しいセッションが開始され新しい cookie がクライアントに送信されます。

WEB GET BODY PART

WEB GET BODY PART(part ; contents ; name{ ; mimeType ; fileName})

引数	型		説明
part	倍長整数	→	パート番号
contents	BLOB, テキスト	←	パートを受け取る変数
name	テキスト	←	input 要素の name 属性値
mimeType	テキスト	←	ファイルの MIME タイプ
fileName	テキスト	←	送信されたファイルの名前

このコマンドは Web プロセスのコンテキストで使用され、マルチパートリクエストのボディ部を解析します。

part 引数には解析対象のパート番号を渡します。総パート数は WEB Get body part count コマンドで取得できます。

contents 引数にはパートのコンテンツが返されます。取得するパートがファイルの場合、BLOB 型の引数を渡さなければなりません。Web フォームから送信されるテキストデータの場合、テキスト型の引数を渡すことができます。

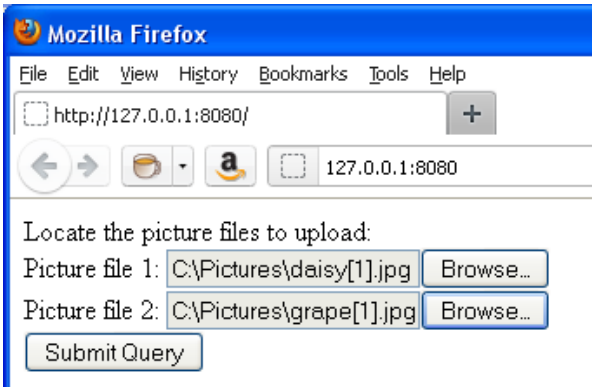
name 引数にはフォームの input 要素の name 属性値が返されます。

mimeType と fileName 引数には、送信されたファイルの MIME タイプと名前が返されます。fileName はフォーム要素 <input type="file"> を使用してファイルが送信された場合のみ値が返されます。

mimeType と fileName はオプションですが、使用する場合はペアで渡さなければなりません。

注 マルチパートリクエストのコンテキストでは、WEB GET VARIABLES コマンドで取得できる名前配列にはフォームのすべてのパートが含まれます。順番は WEB GET BODY PART コマンドと同じです。フォーム中のパートの位置を取得するためにこのコマンドを使用できます。

♦ この例題では Web フォームから HTTP サーバーに画像を何枚かアップロードし、それらを返された Web ページ上に表示します。Web フォームは以下のように表示されます :



HTML ソースは以下の通りです :

```
<body>
    <form enctype="multipart/form-data" action="/4DACTION/GetFile/"
method="post">
      Locate the picture files to upload: <br>
      Picture file 1: <input name="file1" type="file"><br>
      Picture file 2: <input name="file2" type="file"><br>
      <input type="submit">
    </form>
    <hr/>
  <!--4DSCRIPT/galleryInit-->
  <!--4Dloop aFileNames-->
    <img src="/photos/<!--4Dvar aFileNames{aFileNames}-->"/>
  <!--4Dendloop-->
</body>
```

2 つの 4D メソッドがページから呼ばれています :

■ ページを送信する際、4DSCRIP タグで呼び出される galleryInit 。このメソッドは指定されたフォルダー内に存在するピクチャー名の配列を作成します。

- ブラウザーからのリクエストを処理する GetFile メソッド。

galleryInit のコードは以下の通りです :

```
C_TEXT($vDestinationFolder)
ARRAY TEXT(aFileNames;0)
C_LONGINT($i)
$vDestinationFolder:=Get 4D folder(HTML Root Folder)+"photos"+Folder separator //"WebFolder/photos" folder
DOCUMENT LIST($vDestinationFolder;aFileNames)
```

GetFile のコードは以下の通りです :

```
C_TEXT($vPartName;$vPartMimeType;$vPartFileName;$vDestinationFolder)
C_BLOB($vPartContentBlob)
C_LONGINT($i)
$vDestinationFolder:=Get 4D folder(HTML Root Folder)+"photos"+Folder separator
For ($i;1;WEB Get body part count) //for each part
   WEB GET BODY PART($i;$vPartContentBlob;$vPartName;$vPartMimeType;$vPartFileName)
   If ($vPartFileName#"")
      BLOB TO DOCUMENT($vDestinationFolder+$vPartFileName;$vPartContentBlob)
   End if
End for
WEB SEND HTTP REDIRECT("/") // return to page
```

参照 : WEB Get body part count

WEB Get body part count

WEB Get body part count → 倍長整数

引数	型		説明
このコマンドは引数を必要としません			
戻り値	倍長整数	←	ボディ中のパート数

WEB Get body part count コマンドは受信したボディに含まれるパートの数を返します。

参照 : WEB GET BODY PART

WEB Get Current Session ID

WEB Get Current Session ID → テキスト

引数	型		説明
このコマンドは引数を必要としません			
戻り値	テキスト	←	セッション UUID

WEB Get Current Session ID コマンドはカレントの Web リクエストのセッション ID を返します。この ID は 4D が自動で生成します。

このコマンドが Web セッション管理のコンテキストの外で呼び出されると、コマンドは空の文字列を返します。

WEB GET OPTION

WEB GET OPTION(selector ; value)

引数	型		説明
selector	倍長整数	→	取得するオプションのコード
value	倍長整数 , テキスト	←	オプションの値

WEB GET OPTION コマンドは 4D Web サーバー処理に関するオプションの現在の設定値を取得するために使用します。

selector 引数には取得する Web オプションを指定する値を渡します。Web Server テーマの定数を使用できます。定数の詳細は WEB SET OPTION コマンドの説明を参照してください。

参照 : WEB SET OPTION

WEB GET SESSION EXPIRATION

WEB GET SESSION EXPIRATION(sessionID ; expDate ; expTime)

引数	型		説明
sessionID	テキスト	→	セッション UUID
expDate	Date	←	cookie 有効期限日
expTime	Time	←	cookie 有効期限時刻

WEB GET SESSION EXPIRATION コマンドは sessionID に渡された UUID のセッションの cookie の有効期限に関する情報を返します。

注 Web レスポンスがクライアントに送信されるたびに、cookie の有効期限はリクエストが行われた時刻 +Web Inactive session timeout (デフォルト

で 24 時間) に設定されます。例えばデフォルト値の状態で :
最初のリクエスト : 月曜日の 1:00
-> 有効期限は月曜日の 09:00
二番目のリクエスト : 月曜日の 1:10
-> 有効期限は月曜日の 09:10
三番目のリクエスト : 火曜日の 4:00 (cookie の有効期限が過ぎている)
-> 新しい cookie 値が生成され、有効期限は火曜日の 12:00

参照 : WEB SET OPTION

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 , Text	→	オプション値

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 :

注 4D v13 では Web サーバーを管理するためのすべての定数が (前のバージョンで "Database Parameters" テーマに合ったものも含め) "Web Server" テーマにまとめられました。Web オプションを設定するためには今後 WEB SET OPTION コマンドを使用してください (Get database parameter, SET DATABASE PARAMETER 参照)。

セレクター (値)	説明
Web Character set (17)	"Database Parameters" と同じ
Web HTTP Compression Level (50)	"Database Parameters" と同じ
Web HTTP Compression Threshold (51)	"Database Parameters" と同じ
Web HTTPS Port ID (39)	"Database Parameters" と同じ

Web Inactive process timeout (78)	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、サーバー上のプロセスは終了されます。その直前に On Web Session Suspend データベースメソッドが呼び出されます。 **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 (72)	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 (16)	"Database Parameters" と同じ
Web Keep session (70)	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。 **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データベース設定の説明を参照してください。
Web Log Recording (29)	"Database Parameters" と同じ
Web Max Concurrent Processes (18)	"Database Parameters" と同じ
Web Max sessions (71)	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。(直前に On Web Session Suspend データベースメソッドが呼び出されます) **取りうる値** : 倍長整数値。数値は Web Max Concurrent Processes オプション設定値を超えることはできません。 **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 (27)	"Database Parameters" と同じ
Web Session cookie name (73)	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。)

"Database Parameters" テーマから移動された定数のスコープは以前のバージョンと同じです。

新しいオプションはローカル Web サーバーに適用されます。リモートモードの 4D から呼び出された場合、このリモート 4D にのみ適用されます。設定は 4D を終了するまで有効で、ストラクチャーには保存されません。

参照 : WEB GET OPTION

Web サーバーテーマコマンドの名称変更

4D v13 より、Web サーバーテーマのコマンドには "WEB" 接頭辞が付加されるようになりました :

4D v13 の名称	**4D v12.x 以前の名称**
WEB CLOSE SESSION	v13 の新規コマンド
WEB GET BODY PART	v13 の新規コマンド
WEB Get body part count	v13 の新規コマンド
WEB Get Current Session ID	v13 の新規コマンド
WEB GET HTTP BODY	GET HTTP BODY
WEB GET HTTP HEADER	GET HTTP HEADER
WEB GET OPTION	v13 の新規コマンド
WEB GET SESSION EXPIRATION	v13 の新規コマンド
WEB GET STATISTICS	WEB CACHE STATISTICS
WEB GET VARIABLES	GET WEB FORM VARIABLES
WEB Is secured connection	Secured Web connection
WEB SEND BLOB	SEND HTML BLOB
WEB SEND FILE	SEND HTML FILE

WEB SEND HTTP REDIRECT	SEND HTTP REDIRECT
WEB SEND RAW DATA	SEND HTTP RAW DATA
WEB SEND TEXT	SEND HTML TEXT
WEB SET HTTP HEADER	SET HTTP HEADER
WEB SET HOME PAGE	SET HOME PAGE
WEB SET ROOT FOLDER	SET HTML ROOT
WEB SET OPTION	v13 の新規コマンド
WEB START SERVER	START WEB SERVER
WEB STOP SERVER	STOP WEB SERVER
WEB Validate digest	Validate Digest Web Password

注 - 廃止される Web Context, SET WEB DISPLAY LIMITS, SET WEB TIMEOUT および SET CGI EXECUTABLE コマンドには "_o_" 接頭辞が付加され、もう使用することはできません。これらのコマンドを実行するとエラー 33 (メソッドまたは関数が実装されていません) が生成されます。
- PROCESS HTML TAGS コマンドは PROCESS 4D TAGS に名称が変更され " ツール " テーマに移動されました。動作に変更はありません。

ウィンドウ

SET WINDOW RECT

SET WINDOW RECT(left ; top ; right ; bottom{; window}{; *})

引数	型		説明
left	倍長整数	→	ウィンドウ内容領域のグローバル左座標
top	倍長整数	→	ウィンドウ内容領域のグローバル上座標
right	倍長整数	→	ウィンドウ内容領域のグローバル右座標
bottom	倍長整数	→	ウィンドウ内容領域のグローバル下座標
window	WinRef	→	ウィンドウ参照番号 , または省略時カレントプロセスの最前面ウィンドウ

*	*	→ 省略時 (デフォルト) = ウィンドウを最前面にする 指定時 = ウィンドウの並びレベルを変更しない

既存の SET WINDOW RECT コマンドは新しいオプション引数 * を受け入れるようになりました。* 引数が省略されるとデフォルトで自動的に、window 引数で指定されたウィンドウが最前面に移動されます (window 引数が使用されている場合)。
最後の引数として * を渡すとこの動作を無効にできます。この場合コマンドはウィンドウの元の順番 (z 座標) を変更しません。

その他の変更点

より明確に分かりやすくするために、4D v13 のランゲージにいくつかの変更が加えられました。

トリガー

- Database event コマンドは **Trigger event** に名称変更されました。
- "Database Events" 定数テーマも "Trigger Events" テーマに名称が変更されました。

このコマンドの動作や定数の意味に変更はありません。

名称が変更されたことで、4D v13 から加えられた新しいシステムイベントとトリガーイベントとの違いが明確になります (84 ページの "On System Event" 参照)。

Web サービス

Web サービステーマのコマンドの名称が変更されました。

Web サービス (クライアント)

"Web サービス (クライアント)" テーマのコマンドに接頭辞が追加されました :

4D v13 の新名称	4D v12.x 以前の名称
WEB SERVICE AUTHENTICATE	AUTHENTICATE WEB SERVICE
WEB SERVICE CALL	CALL WEB SERVICE
WEB SERVICE Get error info	Get Web Service error info
WEB SERVICE GET RESULT	GET WEB SERVICE RESULT

WEB SERVICE SET OPTION	SET WEB SERVICE OPTION
WEB SERVICE SET PARAMETER	SET WEB SERVICE PARAMETER

これらのメソッドの動作に変更はありません。

Web サービス (サーバー)

"Web サービス (サーバー)" テーマのコマンドにも接頭辞が付けられました :

4D v13 の新名称	4D v12.x 以前の名称
SOAP Get info	Get SOAP info
SOAP Request	Is SOAP request
SOAP SEND FAULT	SEND SOAP FAULT

これらのメソッドの動作に変更はありません。

定数の名称変更

"Process Type" テーマの以下の定数の名称が変更されました :

4D v13 の新名称	4D v12.x 以前の名称
_o_Web Process with Context	Web Process with Context
Web Process on 4D Remote	Web Process on 4D Client

"System Documents" テーマの以下の定数の名称が変更されました :

4D v13 の新名称	4D v12.x 以前の名称
Is a folder	Is a directory

"Web Services (Client)" テーマの以下の定数の名称が変更されました :

4D v13 の新名称	4D v12.x 以前の名称
Web Service HTTP Status code	Web Service HTTP Error code

移動されたコマンド

より明確にするために、いくつかのコマンドのテーマが 4D v13 で変更されました :

- Get edited text は " フォームイベント " から " 入力制御 " テーマに移動されました。
- HIGHLIGHT TEXT と GET HIGHLIGHT は " ユーザーインターフェース " から " 入力制御 " テーマに移動されました。

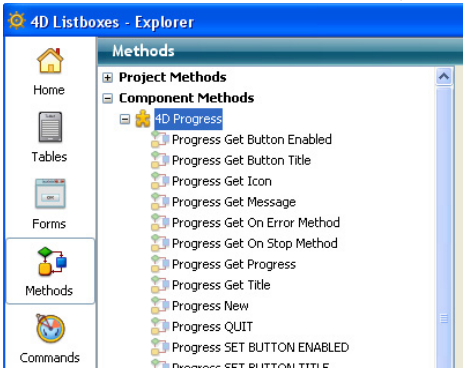
- CANCEL と ACCEPT は " 入力制御 " から " データ入力 " テーマに移動されました。

4D Progress

4D v13 には新しい組み込みコンポーネント、4D Progress が含まれます。このコンポーネントを使用して (Mac OS の Finder のように) 1 つ以上の進捗バーを同じウィンドウに表示することができます。

各進捗バーには Progress New メソッドから自動で ID が割り当てられ、ます。この ID は進捗ダイアログボックスの属性と値を管理するコンポーネントメソッドで使用されます。

このコンポーネントの共有プロジェクトメソッドはエクスプローラーの**コンポーネントメソッド**ページで見ることができます :



Progress Get Button Enabled

Progress Get Button Enabled (id) → ブール

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	ブール	←	True = 停止ボタンを表示 ; False = 停止ボタンを非表示

Progress Get Button Enabled メソッドは id 引数で指定された進捗バーに**停止**ボタンが表示されている場合、True を返します。表示されない場合 (デフォルトオプション) False を返します。

参照 : Progress SET BUTTON ENABLED

Progress Get Button Title

Progress Get Button Title (id) → テキスト

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	テキスト	←	停止ボタンのラベル

注 このメソッドは Windows でのみ使用できます。では停止ボタンにラベルが表示されません。

Progress Get Button Title メソッドは id 引数で指定された進捗バーの停止ボタンのラベルを返します。

デフォルトのラベルは " 停止 " です。メソッドは停止ボタンが表示されていない場合でも設定されているラベルを返す点に留意してください。

参照 : Progress SET BUTTON TITLE

Progress Get Icon

Progress Get Icon (id) → ピクチャー

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	ピクチャー	←	進捗バーのアイコン

Progress Get Icon メソッドは id 引数で指定された進捗バーのアイコンを返します。

参照 : Progress SET ICON

Progress Get Message

Progress Get Message (id) → テキスト

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	テキスト	←	進捗バーメッセージ

Progress Get Message メソッドは id 引数で指定された進捗バーのメッセージを返します。

参照 : Progress SET MESSAGE

Progress Get On Error Method

Progress Get On Error Method → テキスト

引数	型		説明
戻り値	テキスト	←	エラーが発生したときに呼び出されるメソッド (指定した場合)

Progress Get On Error Method は進捗バー使用時エラーが発生した場合に呼び出される、ホストデータベースのプロジェクトメソッド名を返します。

エラーメソッドが指定されていない場合、メソッドは空の文字列を返します。

参照 : Progress SET ON ERROR METHOD

Progress Get On Stop Method

Progress Get On Stop Method(id) → テキスト

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	テキスト	←	停止ボタンがクリックされたときに呼び出されるメソッド (指定した場合)

Progress Get On Stop Method は id 引数で指定された進捗バーの**停止**ボタンがクリックされたときに呼び出される、ホストデータベースのプロジェクトメソッド名を返します。

停止ボタンにメソッドが割り当てられていない場合、メソッドは空の文字列を返します。

参照 : Progress SET ON STOP METHOD

Progress Get Progress

Progress Get Progress (id) → 実数

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	実数	←	進捗バーの値

Progress Get Progress メソッドは id 引数で指定した進捗バーに関連付けられている現在の値を返します。

参照 : Progress SET PROGRESS

Progress Get Title

Progress Get Title (id) → テキスト

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	テキスト	←	進捗バーのタイトル

Progress Get Title メソッドは id 引数で指定された進捗バーのメインタイトルを返します。

参照 : Progress SET TITLE

Progress New

Progress New → 倍長整数

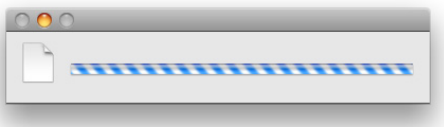
引数	型		説明
戻り値	倍長整数	←	進捗バーの ID

Progress New メソッドは新しい進捗バーを作成し、その ID 番号を返します。この ID は進捗バーが表示されている間はユニークですが、閉じられた後は再利用されることがあります。

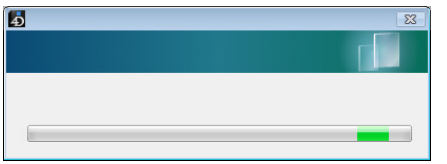
最初にこのメソッドが呼び出されるとローカルプロセスが作成され、メインウィンドウ中央に新しいウィンドウが開かれます。デフォルトでこのウィンドウは :

- 未定義の進捗バーが表示されます。
- メッセージも表示されません。.

Mac OS

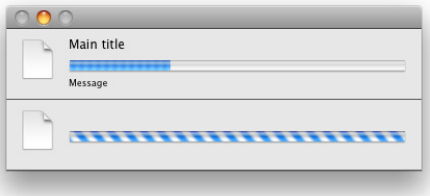


Windows

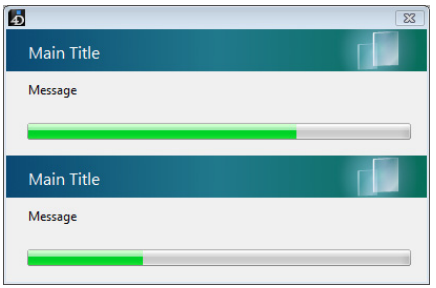


このメソッドを呼び出したとき、すでに進捗バーウィンドウが表示されていた場合、同じプロセス内で新しい進捗バーが下に表示されるようそのウィンドウがリサイズされます。:

Mac OS



Windows



Progress QUIT

Progress QUIT (id)

引数	型		説明
id	倍長整数	→	進捗バーの ID

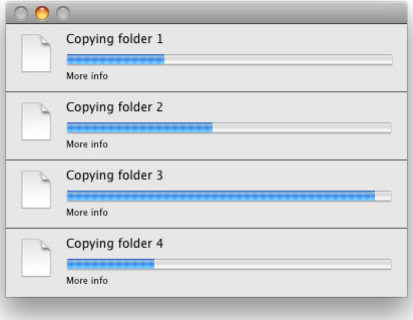
Progress QUIT メソッドは id 引数で指定された進捗バーを閉じます。

id で指定された進捗バーが表示されている唯一のものであれば、ウィンドウ、およびローカルプロセスも閉じられます。そうでない場合表示中の進捗バーだけが表示されるよう、ウィンドウがリサイズされます。

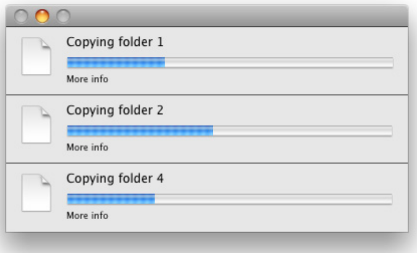
- " フォルダー 3 をコピー中 " 進捗バーの ID が 3 のとき :

Progress QUIT(3)

呼び出し前



呼び出し後



Progress SET BUTTON ENABLED

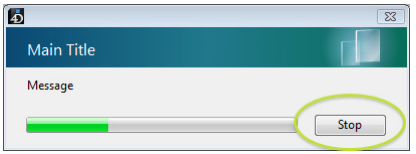
Progress SET BUTTON ENABLED (id; button)

引数	型	説明

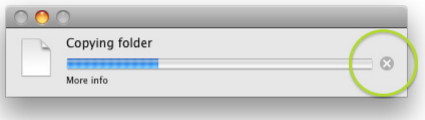
id	倍長整数	→	進捗バーの ID
button	ブール	→	True = 表示 , False = 非表示

Progress SET BUTTON ENABLED メソッドは id で指定された進捗バーの**停止**ボタンの表示 / 非表示を設定するために使用します。

デフォルトで進捗バーには**停止**ボタンが表示されません。button 引数に True を渡すとこのボタンが表示されます。:



Windows



Mac OS

停止ボタンが押されたときの動作は Progress SET ON STOP METHOD メソッドを使用して、または Progress Stopped メソッドの戻り値をテストして、開発者が管理しなくてはなりません。

参照 : Progress Get Button Enabled

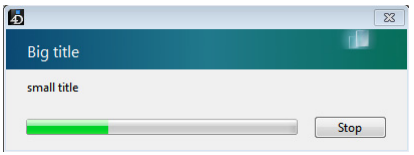
Progress SET BUTTON TITLE

Progress SET BUTTON TITLE (id; title)

引数	型		説明
id	倍長整数	→	進捗バーの ID
title	テキスト	→	停止ボタンのラベル (Windows)

注 このメソッドは Windows のみで利用できます。Mac OS では停止ボタンにラベルがありません。

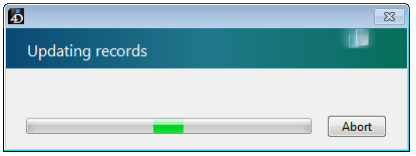
Progress SET BUTTON TITLE メソッドは id 引数で指定された進捗バーの**停止**ボタンのラベルを設定するために使用します。デフォルトでこのボタンのラベルは " 停止 " です :



注 進捗バーの**停止**ボタンはデフォルトでは表示されません。進捗バーにこのボタンを表示させたい場合、Progress SET BUTTON ENABLED メソッドを使用しなければなりません。

- ボタンラベルに " アボート " を使用したい場合 :

```
<>ID:=Progress New
...
Progress SET BUTTON TITLE (<>ID;" アボート ")
```



参照 : Progress Get Button Title

Progress SET FONT SIZES

Progress SET FONT SIZES (sizeTitles{; sizeMessages{; sizeButtons}})

引数	型		説明
sizeTitles	テキスト	→	タイトル用のフォントサイズ
sizeMessages	テキスト	→	メッセージ用のフォントサイズ
sizeButtons	テキスト	→	(Windows) 停止ボタンのフォントサイズ

Progress SET FONT SIZES メソッドはすべての進捗ウィンドウに適用される、各種テキストのフォントサイズを変更するために使用します。

sizeTitles、sizeMessages、そして sizeButtons 引数には使用するフォントサイズを渡します。サイズを変更したくない場合、対応する引数に -1 を渡します。

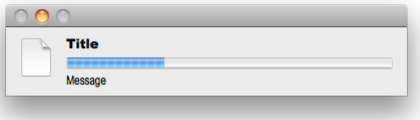
- メッセージのサイズのみを変更する場合 :

```
Progress SET FONT SIZES(-1; 13)
```

- タイトルおよびメッセージのサイズを変個数る場合 :

```
Progress SET FONTS("Arial Black"; "Arial narrow")
```

Progress SET FONT SIZES(13; 12)



参照 : Progress SET FONTS

Progress SET FONTS

Progress SET FONTS (fontTitles{; fontMessages{; fontButtons}})

引数	型		説明
fontTitles	テキスト	→	タイトル用のフォント
fontMessages	テキスト	→	メッセージ用のフォント
fontButtons	テキスト	→	(Windows) 停止ボタン用のフォント

Progress SET FONTS はすべての進捗バーウィンドウで使用される各種フォント名を変更するために使用します。

fontTitles、fontMessages、および fontButtons 引数には使用するフォント名を渡します。フォント名を変更したくない場合、対応する引数に空の文字列 ("") を渡します。

- メッセージ用のフォントのみを変更する場合 :

Progress SET FONTS(""; "Arial")

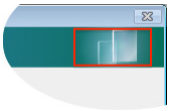
参照 : Progress SET FONT SIZES

Progress SET ICON

Progress SET ICON (id; icon{; foreground}}})

引数	型		説明
id	倍長整数	→	進捗バーの ID
icon	ピクチャー	→	アイコンとして表示するピクチャー
foreground	ブール	→	進捗バーを最前面に表示

Progress SET ICON メソッドは進捗バー内に表示するアイコンを変更するために使用します。デフォルトで以下のアイコンが表示されます :



Windows

Mac OS

id には Progress New から返される進捗バーにユニーク ID を渡します。

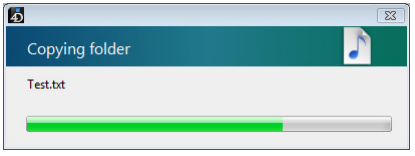
icon には進捗バーウィンドウのアイコンとして使用するピクチャーを渡します。このアイコンの最大サイズは :

- Mac OS, 40 x 40 ピクセル
- Windows, 40 x 80 ピクセル

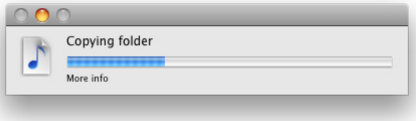
これより小さなアイコンを渡した場合、そのままのサイズで中央に表示されます。制限より大きなアイコンが渡された場合、リサイズされて中央に表示されます。

foreground に True を渡すと、進捗バーはアプリケーションの最前面に表示されます。

カスタムアイコンの例 :



Windows



Mac OS

参照 : Progress Get Icon

Progress SET MESSAGE

Progress SET MESSAGE (id; message{; foreground})

引数	型		説明
id	倍長整数	→	進捗バーの ID
message	テキスト	→	進捗バーに表示するメッセージ
foreground	ブール	→	進捗バーを最前面に表示

Progress SET MESSAGE メソッドを使用して進捗バーに表示されるメッセージを変更できます。

id には Progress New から返される進捗バーにユニーク ID を渡します。

message には (Windows の場合) メインタイトルの下、あるいは (Mac OS の場合) 進捗バーの下に表示に表示するテキストを渡します。

foreground に True を渡すと、進捗バーはアプリケーションの最前面に表示されます。

参照 : Progress Get Message

Progress SET ON ERROR METHOD

Progress SET ON ERROR METHOD (methodName)

引数	型		説明
methodName	テキスト	→	エラーメソッド名

Progress SET ON ERROR METHOD メソッドを使用して、進捗バー使用時にエラー (例えば未知の ID、引数の数が正しくない等) が発生した場合に実行するメソッドを指定できます。

methodName にはエラー発生時に実行する、ホストデータベースのメソッド名を渡します。このメソッドはアプリケーションのすべての進捗ウィンドウ共通です。

注 methodName に渡すメソッドは " コンポーネントとホストデータベースで共有する " プロパティが選択されていなければなりません。そうされていない場合、エラーが返されます。

methodName メソッドが実行されるときには引数が 3 つ渡されます :

- $1 (倍長整数): エラー番号
- $2 (テキスト): エラーテキスト
- $3 (倍長整数): 進捗バーユニーク ID

♦ エラー処理メソッドの例は以下の通りです :

```
C_LONGINT($1)
C_TEXT($2)
C_LONGINT($3)

C_LONGINT($ErrorID)
C_TEXT($ErrorText)
C_LONGINT($ProgressID)
```

```
$ErrorID:=$1
$ErrorText:=$2
$ProgressID:=$3

$Error:=""
$Error:=$Error+" エラー番号 : "+String($ErrorID)+Char(Carriage return)
$Error:=$Error+$ErrorText+Char(Carriage return)
$Error:=$Error+" 進捗 ID: "+String($ProgressID)
ALERT($Error)
```

参照 : Progress Get On Error Method

Progress SET ON STOP METHOD

Progress SET ON STOP METHOD (id; methodName)

引数	型		説明
id	倍長整数	→	進捗バーの ID
methodName	テキスト	→	停止ボタンに割り当てるメソッド

Progress SET ON STOP METHOD メソッドは、ユーザーが進捗バーの**停止**ボタンをクリックしたときに実行されるメソッドを指定するために使用します。

デフォルトで進捗バーには**停止**ボタンが表示されません。ボタンを表示させたい場合 Progress SET BUTTON ENABLED メソッドを使用します。

id には Progress New メソッドから返される進捗バー ID を渡します。

methodName には**停止**ボタンがクリックされたときに呼び出される、ホストデータベースのプロジェクトメソッド名を渡します。このメソッドが実行されるときには第一引数に進捗バーのユニーク ID が渡されます。またこのメソッドはコンポーネントから起動される新規プロセス内で実行されます。

注 methodName に渡すメソッドは " コンポーネントとホストデータベースで共有する " プロパティが選択されていなければなりません。そうされていない場合、エラーが返されます。

参照 : Progress Get On Stop Method

Progress SET PROGRESS

Progress SET PROGRESS (id; progress{; message{; foreground}}})

引数	型	説明

id	倍長整数	→	進捗バーの ID
progress	実数	→	進捗の値 ([0...1] または -1)
message	テキスト	→	進捗バーのメッセージ
foreground	ブール	→	進捗バーを最前面に表示

Progress SET PROGRESS メソッドは進捗バーの値を変更し、進捗ウィンドウに表示されるメッセージを更新するために使用します。特にこのメソッドはループ内で進捗バーを更新するために使用します。

id には Progress New メソッドから返される進捗バー ID を渡します。

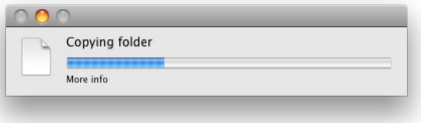
progress には進捗バーの現在値を渡します。 0 から 1 の間の実数値を渡せます。未定義の進捗バー (Mac OS のバーバーショップタイプ) の場合 -1 を渡します。

message には (Windows の場合) メインタイトルの下、あるいは (Mac OS の場合) 進捗バーの下に表示に表示するテキストを渡します。この引数はオプションです。

foreground に True を渡すと、進捗バーはアプリケーションの最前面に表示されます。

- 進捗バーの更新 :

```
$P:=Progress New // 新規バーを作成
   // ループ中で処理を実行
For($i;1;100)
      // ... 処理を行うコード
      // 進捗バーを更新
   $r:=$i/100
   Progress SET PROGRESS ($P;$r;" 詳細情報 ")
End for
   // 処理が終了したら進捗バーを閉じる
Progress QUIT($P)
```



参照 : Progress Get Progress

Progress SET TITLE

Progress SET TITLE (id; title{; progress{; message{; foreground}}})

引数	型		説明
id	倍長整数	→	進捗バーの ID
title	テキスト	→	進捗バーのタイトル
progress	実数	→	進捗バーの値 ([0...1] または -1)
message	テキスト	→	進捗バーのメッセージ
foreground	ブール	→	進捗バーを最前面に表示

Progress SET TITLE メソッドは主に進捗バーのタイトルを設定するために使用します。その他進捗バーウィンドウに表示する情報も設定できます。

id には Progress New メソッドから返される進捗バー ID を渡します。

title には進捗バーウィンドウに表示するメインテキストを渡します。

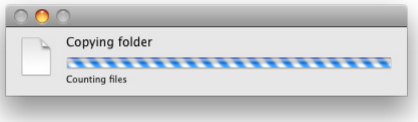
progress には進捗バーの現在値を渡します (オプション)。 0 から 1 の間の実数値を渡せます。未定義の進捗バー (Mac OS のバーバーショップタイプ) の場合 -1 を渡します。

message には (Windows の場合) メインタイトルの下、あるいは (Mac OS の場合) 進捗バーの下に表示に表示するテキストを渡します。この引数はオプションです。

foreground に True を渡すと、進捗バーはアプリケーションの最前面に表示されます。

- シンプルな進捗バーウィンドウを作成 :

```
$P:=Progress New
Progress SET TITLE($P;" フォルダーをコピー中 ";-1;" コピー中のファイル数 ")
```



参照 : Progress Get Title

Progress SET WINDOW VISIBLE

Progress SET WINDOW VISIBLE (visible{; horPos; vertPos{; foreground}})

引数	型		説明
visible	ブール	→	True = 表示 , False = 非表示
horPos	倍長整数	→	ウィンドウの左座標 -1 = 変更しない
vertPos	倍長整数	→	ウィンドウの上座標 -1 = 変更しない
foreground	ブール	→	進捗バーを最前面に表示

Progress SET WINDOW VISIBLE メソッドを使用して、既存の進捗バーウィンドウの表示プロパティを管理できます。

visible 引数を使用してウィンドウの表示 / 非表示を指定できます (デフォルトで表示)。この引数に False を渡すとウィンドウが非表示になり、True を渡すと表示されます。

horPos と vertPos 引数には画面上での進捗バーウィンドウの位置を指定する値を渡します。この値は (Windows の場合) メインアプリケーションウィンドウ、または (Mac OS の場合) スクリーンの左上からのピクセル単位の指定です。
座標を変更したくない場合、各引数に -1 を渡します。

foreground に True を渡すと、進捗バーはアプリケーションの最前面に表示されます。

- 進捗バーウィンドウを左 50 ピクセル、上 100 ピクセルの位置に表示 :

  Progress SET WINDOW VISIBLE(**True**; 50; 100)

- 進捗バーウィンドウを隠す :

  Progress SET WINDOW VISIBLE(**False**)

- 進捗バーウィンドウを表示し、座標は変更せずに最前面にする :

  Progress SET WINDOW VISIBLE(**True**; -1; -1; **True**)

Progress Stopped

Progress Stopped (id) → ブール

引数	型		説明
id	倍長整数	→	進捗バーの ID
戻り値	ブール	←	True = ユーザーが停止ボタンをクリックした

Progress Stopped メソッドは、id 引数で指定した進捗バーの**停止**ボタンをユーザーがクリックした場合 True を返します。

ユーザーが**停止**ボタンをクリックしたかどうかをテストする必要がある場合、このメソッドを呼び出します。ボタン自身はなにも行わない点に留意してください。

- ループ内での進捗バーの例題 :

```
$ProgressID:=Progress New // 新規進捗バーを作成
   // 進捗バーに停止ボタンを表示
Progress SET BUTTON ENABLED ($ProgressID;True)
For ($i;1;100)
      // 停止ボタンがクリックされていなければ繰り返す ...
   If (Not(Progress Stopped ($ProgressID)))
      Progress SET TITLE ($ProgressID;"Test progress
#"+String($ProgressID))
      Progress SET PROGRESS ($ProgressID;$i/100)
      Progress SET MESSAGE ($ProgressID;String(100*$i/100)+" %")
         (...)
   Else // ユーザーが停止ボタンをクリックした
      $i:=100 // ループを抜ける
   End if
End for
   // 進捗バーを閉じる ( 停止ボタンはなにも行わない )
Progress QUIT ($ProgressID)
```

4D SVG

4D v13 で 4D SVG コンポーネントが更新されました。新しいメソッドが追加され、既存のメソッドが一部変更されました。

新しいメソッド

SVG_ADD_NAMESPACE

SVG_ADD_NAMESPACE (svgObject ; prefix {; URI })

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
prefix	テキスト	→	名前空間の接頭辞
URI	テキスト	→	名前空間の URI

SVG_ADD_NAMESPACE メソッドは svgObject 引数で指定した SVG オブジェクトの DOM ツリーのルートに XML 名前空間属性を追加します。このメソッドを使用して特に、SVG コードの一部に名前空間を追加できます。

prefix には名前空間属性の接頭辞を渡します :

- 標準の SVG 名前空間の場合 "svgNS" (http://www.w3.org/2000/svg)
- 標準 XLink 名前空間の場合 "xlinkNS" (http://www.w3.org/1999/xlink)

この場合 URI 引数を渡す必要はありません。

prefix にカスタム名前空間を渡すこともできます。この場合 URI 引数は必須で、対応する引数を渡します。

♦ 以下のコードは :

SVG_ADD_NAMESPACE($svgRef;"svgNS")

SVG オブジェクトのルートに以下のコードを追加します :

<xmlns="http://www.w3.org/2000/svg">

テーマ : 属性

SVG_Color_from_index

SVG_Color_from_index (index)→ テキスト

引数	型		説明
index	倍長整数	→	カラー番号
戻り値	テキスト	←	index で指定されたカラー

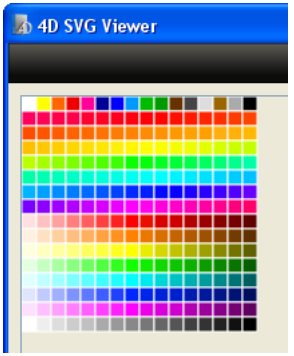
SVG_Color_from_index は index 引数で指定された 4D カラーに対応する SVG カラー名を返します。

index 引数には 4D カラーパレットの番号を渡します。カラー番号は 1 から 256 です。

♦ 以下の例題では 4D カラーパレットを作成しています :

```
$Dom_svg:=SVG_New
$Lon_line:=0
For ($Lon_ii;0;15;1)
    $Lon_column:=0
      For ($Lon_i;1;16;1)
      $Txt_color:=SVG_Color_from_index(($Lon_ii*16)+$Lon_i)
      $Dom_rect:=SVG_New_rect ($Dom_svg;$Lon_column;
```

```
$Lon_line;11;11;0;0;"white";$Txt_color)
        $Lon_column:=$Lon_column+11
        End for
    $Lon_line:=$Lon_line+11
End for
SVGTool_SHOW_IN_VIEWER ($Dom_svg)
```



テーマ : カラーとグラデーション

SVG_DEFINE_STYLE_WITH_ARRAYS

SVG_DEFINE_STYLE_WITH_ARRAYS ((svgObject; namesArrayPointer; valuesArrayPointer {; className {; type {; media {; title }}}})

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
namesArrayPointer	ポインター	→	スタイル名配列へのポインター
valuesArrayPointer	ポインター	→	スタイル値配列へのポインター
className	テキスト	→	CSS クラス名
type	テキスト	→	コンテンツのタイプ
media	テキスト	→	メディアデスクリプター
title	テキスト	→	スタイル名

SVG_DEFINE_STYLE_WITH_ARRAYS は (配列を使用して) svgObject で指定された SVG オブジェクトのスタイルを定義します。

svgObject 引数がルート要素を指している場合、スタイルは "defs" セクションに含まれる "style" として設定されます (内部スタイルシート)。この場合 className 引数は必須であり、省略するとエラーが返されます。

さらに **SVG_SET_CLASS** メソッドに名前を渡して、SVG オブジェクトに className スタイルを割り当てることができます (例題 1 参照)。

- svgObject 引数がルート要素ではない SVG 要素を指す場合、この要素のスタイルとして style が設定されます (インラインスタイル) (例題 2 参照)。

オプションの type 引数を使用して要素コンテンツのスタイルシート言語を指定できます。デフォルト値は "text/css" です。

オプションの media 引数を使用してスタイル情報を適用する対象メディアを指定できます。この引数を省略した場合のデフォルト値は "all" です。CSS2 で認識されるメディアタイプ以外の値を渡した場合、エラーが生成されます。

オプションの title 引数は "title" タイプの属性を追加します。

- 内部スタイル定義の例題 :

```
ARRAY TEXT($arrnames;0)
ARRAY TEXT($arrvalues;0)
APPEND TO ARRAY($arrnames;"fill")
APPEND TO ARRAY($arrvalues;"black")
APPEND TO ARRAY($arrnames;"font-family")
APPEND TO ARRAY($arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY($arrnames;"font-size")
APPEND TO ARRAY($arrvalues;"20px")
APPEND TO ARRAY($arrnames;"text-align")
APPEND TO ARRAY($arrvalues;"center")

$svg:=SVG_New
SVG_DEFINE_STYLE_WITH_ARRAYS($svg;->$arrnames;-
>$arrvalues;"title")
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_SET_CLASS ($object;"title")
```

このメソッドは以下のコードを生成します :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<svg xmlns="http://www.w3.org/2000/svg">
  <defs id="4D">
    <style type="text/css">.title{fill:red;font-family:'Lucida Grande'
Verdana;font-size:20px;text-align:center;}</style>
  </defs>
  <textArea class="title" height="310" width="200" x="10" y="10">Hello
World!</textArea>
</svg>
```

♦ インラインスタイル定義の例題 :

```
ARRAY TEXT($arrnames;0)
ARRAY TEXT($arrvalues;0)
APPEND TO ARRAY($arrnames;"fill")
APPEND TO ARRAY($arrvalues;"black")
APPEND TO ARRAY($arrnames;"font-family")
APPEND TO ARRAY($arrvalues;"'Lucida Grande' Verdana")
APPEND TO ARRAY($arrnames;"font-size")
APPEND TO ARRAY($arrvalues;"20px")
APPEND TO ARRAY($arrnames;"text-align")
APPEND TO ARRAY($arrvalues;"center")

$svg:=SVG_New
$object:=SVG_New_textArea($svg;"Hello World!";10;10;200;310)
SVG_DEFINE_STYLE_WITH_ARRAYS($object;->$arrnames;->$arrvalues)
```

このメソッドは以下のコードを生成します :

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<svg xmlns="http://www.w3.org/2000/svg">
  <textArea height="310" style="fill:red;font-family:'Lucida Grande'
Verdana;font-size:20px;text-align:center;" width="200" x="10"
y="10">Hello World!</textArea>
</svg>
```

テーマ : ストラクチャーと定義

SVG_Get_root_reference

SVG_Get_root_reference (svgObject)→ SVG_Ref

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
戻り値	SVG_Ref	←	ルート SVG 要素

SVG_Get_root_reference は svgObject 引数で指定された参照を持つ SVG オブジェクトのルート要素を返します。

テーマ : ユーティリティ

SVG_SET_DOCUMENT_VARIABLE

SVG_SET_DOCUMENT_VARIABLE (pointer)

引数	型		説明
pointer	ポインター	→	設定する変数へのポインター

SVG_SET_DOCUMENT_VARIABLE メソッドは、SVG_SAVE_AS_PICTURE や SVG_SAVE_AS_TEXT メソッドが呼び出されるたびに更新されるホストデータベースの変数へのポインターを設定します。このメソッドはセッション毎に一回だけ呼び出さなくてはなりません (例えば初期化メソッドで)。

pointer 引数には値を追跡したい変数へのポインターを渡します (通常 Document システム変数)。

リンクを取り除くには Nil を渡します。

テーマ : ドキュメント

SVG_SET_HUE

SVG_SET_HUE (svgObject ; hue)

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
hue	倍長整数	→	色相

SVG_SET_HUE メソッドは svgObject 引数で指定した SVG オブジェクトの色相値を設定します。svgObject は SVG コンテナ (svg、グループ、シンボル、パターン、マーカー等) またはイメージでなければなりません。そうでない場合エラーが返されます。

hue 引数には 0 から 360 までの値を渡します。

テーマ : カラーとグラデーション

SVG_SET_SATURATION

SVG_SET_SATURATION (svgObject ; saturation)

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
saturation	倍長整数	→	彩度

SVG_SET_SATURATION メソッドは svgObject 引数で指定した SVG オブジェクトの彩度値を設定します。svgObject は SVG コンテナ (svg、グループ、シンボル、パターン、マーカー等) またはイメージでなければなりません。そうでない場合エラーが返されます。

saturation 引数には 0 から 100 までの値を渡します。

テーマ : カラーとグラデーション

SVG_Post_comment

SVG_Post_comment (svgObject; comment)→ SVG_Ref

引数	型		説明
svgObject	SVG_Ref	→	SVG オブジェクト参照
comment	テキスト	→	コメントとして追加するテキスト
戻り値	SVG_Ref	←	コメントへの参照

SVG_Post_comment メソッドは svgObject 引数で指定した SVG オブジェクトに、comment に渡したテキストをコメントとして追加します。

このメソッドはコメントの SVG 参照を返します。

- 以下のコードは :

```
C_TEXT($comment)
$comment:="Modified on "+String(Current date)
$ref:= SVG_Post_comment ($svg; $comment )
```

.$svg SVG オブジェクトに以下のコメントを追加します :

```
<!--Modified on 12/12/2011-->
```

テーマ : ユーティリティ

更新されたメソッド

SVG_CLEAR

SVG_CLEAR {(svgObject)}

SVG_CLEAR メソッドに (ルート SVG オブジェクトだけでなく) 有効な SVG オブジェクトを何でも渡せるようになりました。この場合、参照されたオブジェクトは削除されます。

テーマ : ドキュメント

SVG_Define_linear_gradient

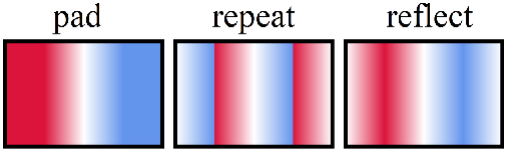
SVG_Define_linear_gradient (parentSVGObject ; id ; startColor ; endColor {; rotation{; spreadMethod{; x1; y1; x2; y2}}})→ 文字列

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素参照
id	文字列	→	グラデーション名
startColor	文字列	→	開始色

endColor	文字列	→	終了色
rotation	整数	→	グラデーションベクトルの回転
spreadMethod	テキスト	→	グラデーション展開方法 (pad, reflect または repeat)
x1	実数	→	グラデーションベクトルの x1 座標 省略時 = 0
y1	実数	→	グラデーションベクトルの y1 座標 省略時 = 1
x2	実数	→	グラデーションベクトルの x2 座標 省略時 = 0
y2	実数	→	グラデーションベクトルの y2 座標 省略時 = 1
戻り値	文字列	←	グラデーション参照

SVG_Define_linear_gradient は以下の新しいオプション引数を受け入れるようになりました :

- spreadMethod: グラデーションが parentSVGObject オブジェクトの範囲内で開始または終了するときの塗りつぶしを指定します。以下のいずれかの文字列を渡せます :
  - "pad": 残りのエリアを塗りつぶすためにターミナルカラーを使用。
  - "reflect": オブジェクトが塗りつぶされるまで、連続的に、開始から終了、そして終了から開始のグラデーションパターンを反映する。
  - "repeat": オブジェクトが塗りつぶされるまで、連続的に、開始から終了、そして開始から終了グラデーションパターンを繰り返す。.



この引数が省略された場合 "pad" 効果が使用されます。

- x1, y1, x2, y2: これらの引数でグラデーションベクトルを指定します。このベクトルは描画エンジンで使用される開始および終了ポイントを提示します。これらの引数には比率 (0...1) で表される割合を渡さなければなりません。

テーマ : ストラクチャーと定義

SVG_New_embedded_image

SVG_New_embedded_image (parentSVGObject ; picture {; x {; y}}{codec})→ SVG_Ref

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素参照
picture	テキスト	→	埋め込むピクチャー
x	倍長整数	→	左上角の X 軸座標
y	倍長整数	→	左上角の Y 軸座標
codec	テキスト	→	使用するコーデック
戻り値	SVG_Ref	←	SVG テキストオブジェクト参照

SVG_New_embedded_image メソッドは新しくオプションの codec 引数を受け入れるようになりました。codec はピクチャーに使用されるコーデックを指定するために使用されます。この引数が省略された場合のデフォルトは ".png" です。

SVG_New_text

SVG_New_text (parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color {; rotation {; lineSpacing {; stretching}}}}}}}}})→ SVG_Ref

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素参照
text	テキスト	→	挿入するテキスト
x	倍長整数	→	X 軸座標
y	倍長整数	→	Y 軸座標
font| styleDef	文字列	→	フォント名またはスタイル定義
size	整数	→	ポイント単位の文字サイズ
style	整数	→	文字のスタイル
alignment	整数	→	整列
color	文字列		テキストカラー
rotation	実数		テキストの回転角度

lineSpacing	実数		ポイント単位の行間
stretching	実数		横ストレッチファクター
戻り値	SVG_Ref	←	SVG テキストオブジェクト参照

SVG_New_text メソッドは異なるシンタックスを受け入れるようになりました :

SVG_New_text($Dom_svg;"Hello World !"; x ; y ; style_definition)

styleDef に完全なスタイル定義を含めることができます。例えば "{font-size:48px;fill:red;}" を渡すと style 属性定義が追加されます :

style="font-size:48px;fill:red;"

この場合他の追加引数は無視されます。

テーマ : テキスト

SVG_New_textArea

SVG_New_textArea (parentSVGObject ; text {; x {; y {; width {; height {; font | styleDef {; size {; style {; alignment}}}}}}})→ SVG_Ref

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素参照
text	テキスト	→	挿入するテキスト
x	倍長整数	→	X 軸座標
y	倍長整数	→	Y 軸座標
width	倍長整数	→	テキストエリアの幅
height	倍長整数	→	テキストエリアの高さ
font | styleDef	文字列	→	フォント名またはスタイル定義
size	整数	→	ポイント単位の文字サイズ
style	整数	→	文字のスタイル
alignment	整数	→	整列
戻り値	SVG_Ref	←	SVG テキストオブジェクト参照

SVG_New_textArea メソッドは異なるシンタックスを受け入れるようになりました :

SVG_New_textArea($Dom_svg;"Hello World !"; x ; y ; width; height; style_definition)

styleDef に完全なスタイル定義を含めることができます。例えば "{font-size:48px;fill:red;}" を渡すと style 属性定義が追加されます :

style="font-size:48px;fill:red;"

この場合他の追加引数は無視されます。

テーマ : テキスト

SVG_New_tspan

SVG_New_tspan (parentSVGObject ; text {; x {; y {; font | styleDef {; size {; style {; alignment {; color}}}}}}})→ SVG_Ref

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素参照
text	テキスト	→	挿入するテキスト
x	倍長整数	→	X 軸座標
y	倍長整数	→	Y 軸座標
font | styleDef	文字列	→	フォント名またはスタイル定義
size	整数	→	ポイント単位の文字サイズ
style	整数	→	文字のスタイル
alignment	整数	→	整列
color	文字列	→	テキストカラー
戻り値	SVG_Ref	←	SVG テキストオブジェクト参照

SVG_New_tspan メソッドに 2 つの変更がありました :

- このメソッドを "textArea" 要素に適用できるようになりました。
- 異なるシンタックスを受け入れるようになりました :

SVG_New_tspan($Dom_svg;"Hello World !"; x ; y ; style_definition)

styleDef に完全なスタイル定義を含めることができます。例えば "{font-size:48px;fill:red;}" を渡すと style 属性定義が追加されます :

style="font-size:48px;fill:red;"

この場合他の追加引数は無視されます。

テーマ : テキスト

SVG_SAVE_AS_PICTURE

SVG_SAVE_AS_PICTURE (svgObject ; document {; codec})

SVG_SAVE_AS_PICTURE は SVG_SET_DOCUMENT_VARIABLE メソッドで指定された変数の値を更新するようになりました。

テーマ : ドキュメント

SVG_SAVE_AS_TEXT

SVG_SAVE_AS_TEXT (svgObject {; document})

SVG_SAVE_AS_TEXT は SVG_SET_DOCUMENT_VARIABLE メソッドで指定された変数の値を更新します。

二番目の引数 document を省略できるようになりました。この引数を省略することは空の文字列 "" を渡すことと同じです。標準のファイルを保存ダイアログが表示され、ユーザーが保存ファイルの場所と名前を指定できます。

テーマ : ドキュメント

SVG_SET_MARKER

SVG_SET_MARKER (svgObject ; url {; position})

url 引数に "none" または空の文字列を渡して、マーカーを取り除くことができるようになりました。

テーマ : 属性

定数の XLIFF ファイル化

4D v13 ではコード中で使用する定数を XLIFF フォーマットで定義することができるようになりました。これらの定数は今まで 4DK# リソースとして格納する必要がありました。4D 開発者は XLIFF を使用してカスタム定数を定義できるようになります。

注 - 4D 定数ファイルは内部的な ID に基づく特別なフォーマットで記述されていて、ユーザー定数を定義するためには使用できません。カスタム定数を定義するにはこの章で説明するフォーマットを使用しなければなりません。
- カスタム定数名の長さの上限は 31 文字です。

定数ファイルの場所

4D アプリケーションでカスタム定数を使用するためには以下の手順を行います :

- XLIFF フォーマットでカスタム定数ファイルを作成する。どのような名前でも使用できますが、拡張子は .xlf でなければなりません (例えば "MyConstants.xlf")。
ファイルの記述形式は後述します。
- このファイルをデータベースまたはコンポーネントのの **Resources** フォルダーに配置し、データベースを開く。
すると開かれたデータベースでこのファイルに定義された定数を利用できるようになります。

注 4D は **Plugins** フォルダーの **Resources** サブフォルダーに配置された XLIFF フォーマットの定数もロードします。

定数ファイルのフォーマット

"My constants" というテーマで定数を作成します。以下の定数を定義します :

定数	型	値
K_first_constant	倍長整数	42
K_second_constant	テキスト	"this"
K_third_constant	実数	3.1415

このためには以下のように XLIFF ファイルを記述します :

```
<group restype="x-4DK#">
  <trans-unit id="1">
    <source>My constants</source>
  </trans-unit>
  <trans-unit id="2" d4:value="42">
    <source>K_first_constant</source>
  </trans-unit>
  <trans-unit id="3" d4:value="this">
    <source>K_second_constant</source>
  </trans-unit>
  <trans-unit id="4" d4:value="3.1415">
    <source>K_third_constant</source>
  </trans-unit>
</group>
```

シンタックスの説明は以下の通りです :

- restype="x-4DK#" は <group> 要素が定数テーマを定義することを示しています。
- グループの最初の <trans-unit> はテーマ名に使用されます。
- <trans-unit> に d4:value 属性が存在する場合、その要素は定数定義として扱われます。低数値は d4:value 属性として定義されます。名前は <source> 要素で指定します。
- 定数の型は d4:value 属性に渡した値に基づきます :
  - 値が番号およびマイナス記号のみを含む場合、倍長整数型。
  - 値が倍長整数範囲を超える、または指数や小数点を含む場合、実数型。
  - 他のケースでは文字型。

  しかし値の後に ":L", ":R" または ":S" を付加して型を強制することもできます。例えば :
  d4:value="42:S" は値が "42" である文字定数を定義します。
- d4:value 属性を含む <trans-unit> 要素の順番は、定数が表示される順番に反映されます。