4D v12

Upgrade
Windows®/Mac OS®

4D v12 - 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 v12 へようこそ。もっとも広まっている技術に対しオープンであり、そして開発者の生産性や創造性を向上させることを意図した多くの新機能により、このバージョンは 4D 製品ラインの進化における大きな新しいステップとなりました。

4D の開発者コミュニティで要望されていた多くの機能が、アーキテクチャや開発環境、およびランゲージレベルで統合されました。これにはランゲージエディタの向上、PHP スクリプトの実行、UUID の管理、オブジェクトプロパティを管理するコマンドの向上、そして XML と SVG のサポート拡張が含まれます。

4D アプリケーションにコンポーネントをインストールできるようになったため、すべてのデータベースでそれを使用することができるようになりました。

印刷機能が強化され、新しい PRINT OBJECT コマンドによりリストボックスの印刷や PDF 印刷のサポートが拡張されました。

新しいフォームオブジェクトやウィジェットはモダンで洗練されたインタフェースの作成を加速させます。ステッパー、スタイル付きテキスト、日付および時間のセレクタ (datepicker)、原文の検索エリア、そして階層リストボックスまでもがこれらの新しい機能に含まれます。サブフォームも多くの向上の恩恵を受け、特にコンポーネントとしての利用が容易になりました。新しい、設定済みのオブジェクトライブラリにより、これらの機能にすばやくアクセスできます。

最後に、4D の SQL ランゲージに、データの複製と同期、および同一セッション内で異なる 4D データベースを開いたり閉じたりすることのできる、パワフルな機能が追加されました。

これらの新機能はすべて以下の章で説明しています :

- アーキテクチャとデータベース
- 開発環境
- フォームとオブジェクト
- ランゲージ
- SQL エンジン
- 4D Server 管理

4D Server 64-bit バージョンの 4D Server v12 は現時点でまだ開発中です。ベータバージョンは弊社の Web サイト (http://www.4d.com/) に接続することで評価していただけます。

旧データベースの変換

4th Dimension、4D、あるいは 4D Server のバージョン 6.x、2003.x、2004.x、および 11.x で作成されたデータベースは、ストラクチャおよびデータファイルとも 4D バージョン 12 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する必要があり、元のバージョンでは開けなくなります。
- バージョン 11 のデータベースファイルは直接バージョン 12 に変換されます。一度変換されるとストラクチャファイルをバージョン 11 で開くことはできなくなります。特定の条件下では、データファイルを再びバージョン 11 で開くことが可能です (16 ページ " バージョン 11 のデータベース " 参照)。

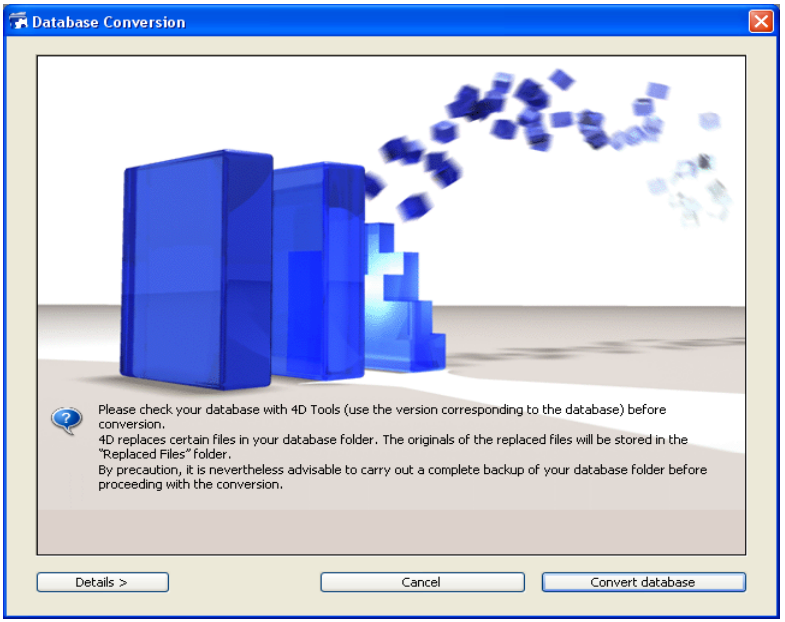
Note: インタプリタのストラクチャファイルを変換できます。ファイルにコンパイルコードが含まれていてもかまいませんが、変換後に再度データベースをコンパイルする必要があります。

バージョン 6.x、2003.x そして 2004.x のデータベース

4D データベースエンジンレベルに行われた構造的な変更のため、旧データベースをバージョン 11 や 12 に変換する際にはストラクチャとデータファイルをより深く変換する必要があります。この変換は特別なウィザードを通じて行います。事前に注意を表示して、ウィザードは変換前にオリジナルデータベースのコピーを作成するので、いつでも元のバージョンに戻ることができます。

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、必要であれば修復) することをお勧めします。オリジナルのデータベースのバージョンに対応する 4D Tools を使用してください。

旧データベースを変換するには、4D v12 のデータベースを開くダイアログボックスで目的のデータベースを選択します。変換ウィザードが自動で表示されます :

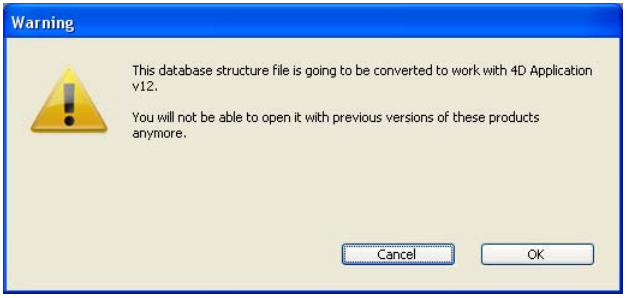


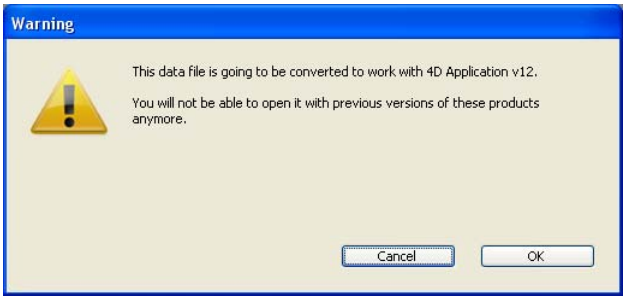
データベースを変換ボタンをクリックし、ストラクチャとデータファイルを変換する標準の処理を開始します。

- このダイアログボックスと変換処理に関する詳細情報は 4D Design Reference マニュアルの " 過去のデータベースの変換 "、または 4D v11 SQL アップグレードリファレンスの " 前バージョンからのデータベースの移行 " を参照してください。
- 特定の古いバージョンおよび廃止されたメカニズムはサポートされておらず、変換時に削除または置き換えられます。詳細は 4D v11 SQL アップグレードリファレンスを参照してください。変更点に関する詳細な説明は以下のアドレスからダウンロードできる " 変換の手引き (v11)" を参照してください : http://doc.4d.com/Home/home.ja.html

バージョン 11 のデータベース

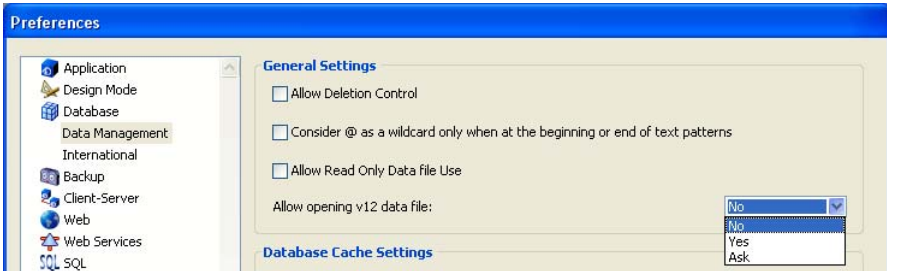
バージョン 11 からのデータベース変換は、ストラクチャファイルが 4D v12 で開かれるときに直接行われます。2 つの警告が連続して表示され、ファイルが変換されること、そしてそれらのファイルは以前のバージョンで開けなくなることを知らせます :





変換したデータファイルをバージョン 11 で開くことができる点に留意してください。これを行うにはバージョン 11 の側で、環境設定の " データベース / データ管理 " ページにある **v12 データファイルを開くことを許可** を使用して、明示的に認可しなくてはなりません (4D v11 SQL r6 以降のみ):

v12 のデータファイルを開く 4D v11 SQL r6 環境設定オプション



データベーステーブルに v12 特有の属性が適用されている場合、このオプションは注意を持って使用されなければなりません (データが変更されるリスクがあります)。

Macros.xml ファイルの更新

"Macros.xml" ファイルを使用すると、メソッドエディタでマクロコマンドを使用することができます (Design Reference マニュアル参照)。このファイルはユーザ環境設定フォルダにあります :

- **Windows XP**: C:\Documents and Settings\ ユーザ名 \Application Data\4D\Macros v2\
- **Windows VISTA/Windows 7**: C:\Users\ ユーザ名 \AppData\Roaming\4D\Macros v2\
- **Macintosh**: {Startup disk}:Users: ユーザ名 :Library:Preferences:4D:Macros v2:

4D バージョン 12 では、新しい SQL コマンド (Chapter 6 の "SQL" 参照) の使用を容易にするために、新しいマクロコマンドを利用できるようになりました。"Macros.xml" ファイルはカスタマイズが可能なので、新しいバージョンの 4D をインストールしても既存のファイルを自動で置き換えることはしません。4D v12 の新しい SQL マクロコマンドを使用するためには以下のいずれかを行います :

- ファイルをカスタマイズしていない場合、"Macros v2" フォルダ内の "Macros.xml" ファイルを削除し、4D を起動します。新しいファイルが自動で作成されます。
- ファイルの内容をカスタマイズしている場合、"Macros v2" フォルダ内の "Macros.xml" ファイルに新しいマクロを手作業で追加します。マクロの新しいテンプレートファイルは 4D アプリケーションフォルダの以下の場所で見つけることができます :
  Mac OS バージョン : 4D:Contents:Resources:ja.lproj
  Windows バージョン : 4D\Resources\ja.lproj

最低動作環境

4D バージョン 12 製品ラインのアプリケーションは最低以下の環境を必要とします :

	Windows	Mac OS
プロセッサ	Pentium 4 プロセッサ	Intel® プロセッサ
システム	Windows Vista, Windows XP, Windows 7	Mac OS version 10.5 以降
RAM メモリ	1 GB (2 GB 推奨)	
画面解像度	1280 * 1024 ピクセル	

廃止予定の機能に関する終了予定について

特定の以前の機能は互換性を保つため 4D v12 でも保持されていますが、次の 4D メージャーバージョンでは完全に取り除かれる予定です。開発するアプリケーションを進化させる目的においても、これらの機能を今後使用せず、開発を新しい方法に適合させることをお勧めします。

プラットフォーム名によるアピアランス

バージョン 2003 まで、データベースやフォーム、オブジェクトに特定のプラットフォームアピアランスを割り当てることが可能でした。さまざまなオプション Mac OS 7、Windows 3.11/NT 3.51、Windows 95/95/2000/NT4、Mac OS9、そして Mac テーマを指定できました。この方法は既に廃止予定であり、4D v12 は変換されたデータベースにおいてこれらのアピアランスを使用できる最後のバージョンとなります。将来のバージョンでは、4D はこのタイプの属性を持つすべてのオブジェクトに、自動でシステムアピアランスを割り当てます。

Web サーバーのコンテキストモード

4D v12 はコンテキストモードの Web サーバーをサポートする最後のバージョンとなります。この特別なモードとメカニズムは次のバージョンでは動作しなくなります。

2 アーキテクチャとデータベース

4D アプリケーションのアーキテクチャおよび統合データベースエンジンに関連する新しい 4D v12 の機能を紹介します :

- UUID のサポート
- 文字やテキストフィールドのスタイルタグを処理する新しいオプション
- SQL を使用した新しいデータ複製のオプション
- 新しい読み込みと書き出しのオプション
- 4D アプリケーションレベルへのコンポーネントのインストール

UUID のサポート

4D v12 は UUID (Universally Unique Identifiers) の完全なサポートを提供します。4D 開発者はデータベースにこれを統合できます。

概要

UUID はユニーク ID のタイプであり、もとは Open Software Foundation (OSF) により標準化されました。UUID はユニークであるように設計され、同じ UUID が生成されるリスクは事実上ゼロです。UUID でラベルづけされた情報は命名の衝突の恐れなく、後で統合することが可能です。

ひとつの UUID は 16-byte (128 bit) の番号で、32 の 16 進文字が含まれています。UUID は非標準の形式 (一連の 32 文字 [A-F, a-f, 0-9]、例えば 550e8400e29b41d4a716446655440000) または標準の形式 (8,4,4,4,12 桁でグループ化、例えば 550e8400-e29b-41d4-a716-446655440000) で表現できます。

4D v12 で、UUID は ("UUID フォーマット " がチェックされた) 特別な文字タイプのフィールドに特別なフォーマットで格納されます。このフィールドに格納する値は自動もしくはプログラムにより生成できます。

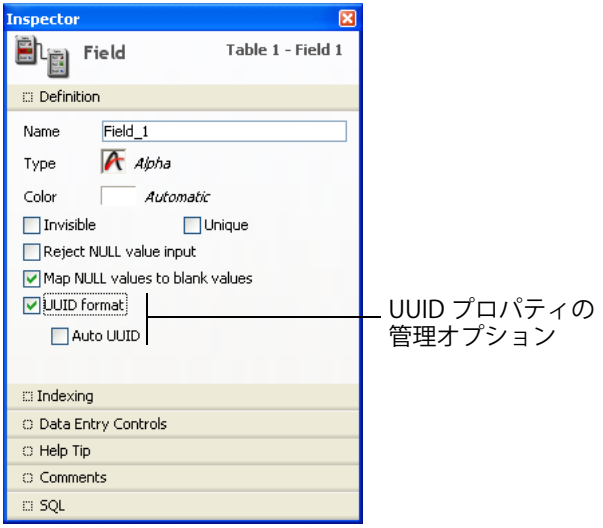
4D v12 では以下のことが可能です :

- UUID フォーマットの値を格納するフィールドを指定する (16 byte)
- 新しいオプションを使用して、UUID フォーマット属性を持つフィールドに自動で UUID を生成する
- UUID フォーマット属性を持つフィールドに、新しい Generate UUID コマンドまたは独自のアルゴリズムを使用して UUID を格納する
- UUID 値を使用してレコードを検索する (UUID は文字フィールドに格納されていますが、UUID は数値です。UUID フィールドに対して " 含む " タイプの検索や "@" ワイルドカードを使用した検索を行うことはできません。)

Note: UUID に関連するフィールドプロパティは 4D SQL コマンドを使用してアクセスできます (290 ページの "UUID フィールドのサポート " 参照)。

フィールドの新しい UUID プロパティ

4D v12 で UUID をサポートするため、2 つの新しいオプションがフィールドインスペクタに追加されました :



これらのオプションにアクセスするには、文字タイプを選択します。

UUID フォーマット

このプロパティは、そのフィールドが UUID 識別子を格納するために使用されるということを示します。このフィールドに格納されるデータは UUID フォーマットでなければなりません (32 文字 (A-F, a-f, 0-9) の組み合わせ)。**Auto UUID** プロパティ、新しい Generate UUID コマンド、あるいはカスタムアルゴリズムを使用できます。

このフィールドに UUID フォーマットに合致しない文字列を格納しようとすると、4D はそれを自動で再フォーマットします。4D は文字列の最初の 16 文字のコードを 16 進フォーマットに変換し、結果を格納します。残りの文字は切り取られます。文字が 16 文字に満たない場合、足りない文字はスペース (16 進で "20") で埋められます。非文字フィールドを UUID フィールドに変更したときにも同じ処理が値に対して行われます。レコードロード時に値が再フォーマットされ、再び格納されます。
このメカニズムは正しくフォーマットされた値を取得するためにのみ行われます。いかなる状況においても、格納される値のユニーク性は保証されません。適切なツールを使用して有効な UUID を生成するのは開発者に任されています (" 自動 UUID" オプション、Generate UUID コマンド、またはカスタムアルゴリズム)。

UUID フォーマットプロパティを持つフィールドはフォームに表示可能で、入力することもできます。値は大文字で表示されます。小文字で表示したい場合は、変数を使用します。

Note: - UUID フォーマットのフィールドにキーワードインデックスや選択リストを割り当てることはできません。
- UUID フォーマットを持つフィールド間でリレーションを作成できます。しかし、標準の文字フィールドと UUID フォーマットのフィールド間ではリレーションを作成できません。

自動 UUID

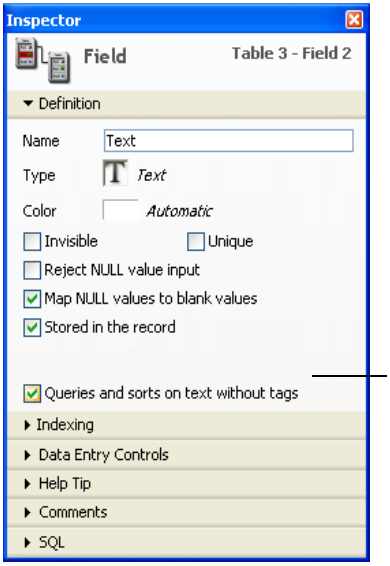
このオプションは **UUID フォーマット**プロパティがチェックされているときにのみ有効になります。

自動 UUID プロパティはレコード作成時に自動で UUID をフィールド中に生成するために使用できます。

Note: データが読み込まれる際、このプロパティがチェックされていても 4D は新しい ID を生成せず、読み込まれた値を使用します (フォーマットが有効でなければ変換を行います)。しかし読み込まれた値が空の場合、UUID が自動で生成されます。

スタイルタグ付きフィールドのクエリとソート

新しい**クエリとソートをタグなしテキストで行う**プロパティがテキストおよび文字フィールドで利用できます :



このオプションがチェックされていると、フィールドに格納されているデータへのクエリやソートは、含まれているかもしれないスタイルタグを無視して行われます。

このオプションは、フォーム上のひとつのテキストエリア内で異なるスタイルを適用できる 4D v12 の新しい機能に関連します。この新機能は 94 ページの " リッチテキスト " で説明しています。スタイルの設定は HTML タグの挿入によって行われます。これらのタグはエリアが表示されるときに解釈されます。

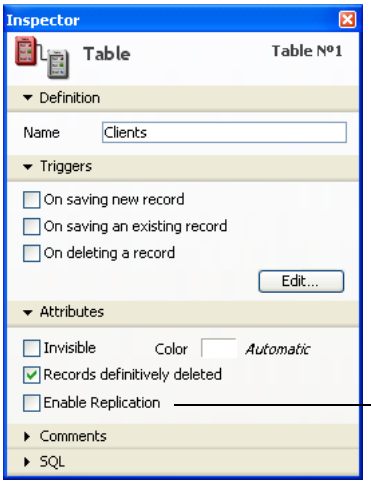
スタイルタグはデータとともに格納されます。例えばテキストフィールドに "week end" と書くと、4D は "week <SPAN STYLE = "color : #D81E05">end</SPAN>" を格納します。この動作はフォームレベルでユーザに対し透過的です。しかしクエリやソートにおいて、4D にスタイルタグを無視させるために特別な指定が必要です。クエリとソートをタグなしテキストで行うオプションがストラクチャエディタでチェックされている場合のみ、"week end" をクエリで見つけることができます。

Note: このオプションを設定してフィールドに対し検索を行う場合、4D の以下のコードを実行するのと同等です :
QUERY BY FORMULA(Object Get plain text(thefield)="thevalue")

データの複製

4D v12 は洗練されたデータ複製システムを構築するために使用する、新しいメカニズムを提供します。これらのメカニズムは SQL ランゲージに基づきます。

4D データベースのストラクチャでは、新しいプロパティ**複製を有効にする**を使用して、テーブルごとに複製に必要な情報の格納を有効にできます : .



このオプションに関する詳細とレコード複製メカニズムに関する詳細は、279 ページの "SQL による複製 " を参照して t ください。

主キーの設定

4D v12 ではストラクチャエディタで直接テーブルの主キーを管理できます。

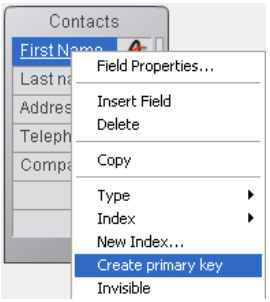
SQL ランゲージにおいて、主キーはテーブルのレコード (行) をユニークに特定するための列 (フィールド) を識別するために使用します。主キーの設定は特に、4D テーブルにおいてレコードの複製のために必要です (279 ページの "SQL による複製 " 参照)。SQL ランゲージで、主キーは列リスト中 PRIMARY KEY 句を使用して設定されます。

4D ではストラクチャエディタでコンテキストメニューを使用して直接主キーを作成したり取り除いたりできます。

▶ 主キーを作成するには :

1 テーブルの主キーを構成するフィールドを選択する。

2 右クリックして、コンテキストメニューから主キーを作成 コマンドを選択する :



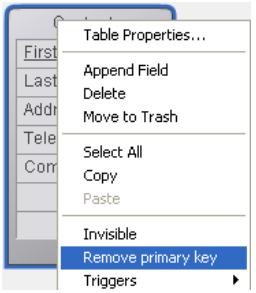
フィールドは主キーに含まれ、エディタ中で下線が表示されます。そして SQL の説明には PRIMARY KEY キーワードが表示されます。

主キーに属するフィールドには重複した値が含まれてはなりません。テーブルのレコードに既に重複した値が存在する場合、そのことを示す警告ダイアログボックスが表示されます。

Note: 主キーに属する列は NULL 値を許可しません。

▶ テーブルから主キーを取り除くには :

1 主キーを含むテーブルを右クリックし、コンテキストメニューから主キーを取り除く コマンドを選択する :



確認ダイアログボックスが表示されます。**OK** をクリックすると即座にキーは取り除かれます。

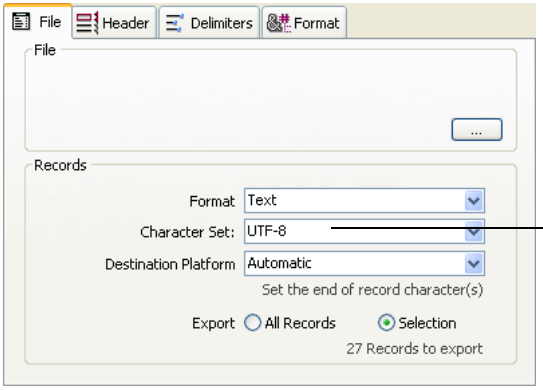
データの読み込みと書き出し

4D 標準のデータ読み込み / 書き出しダイアログボックスに新しいオプションが追加され、Unicode のテキスト交換と XML フォーマットのサポートが向上しました。:

- 文字セットの選択 (読み込みおよび書き出し)
- Byte Order Mark の統合 (書き出し)
- 新しい XML 書き出しオプション

文字セットの選択

4D v12 の読み込みおよび書き出しダイアログボックスでは、特定のフォーマットで、データ交換に使用する文字セットを設定できるようになりました :



このメニューはテキスト、SYLT、そして XML (書き出し) のファイルフォーマットで利用できます。このメニューには IANA で定義された標準的な文字セットのリストが含まれます (詳細な情報は以下の Web サイトを参照してください : http://www.iana.org/assignments/character-sets)。

Note: FDIFF および DBF フォーマットでは文字セットメニューが "IBM437" に固定されます。またこのメニューは 4D フォーマットでは使用できません。

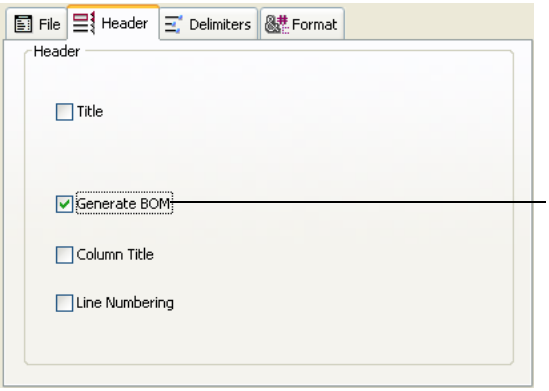
- 書き出しの場合、このメニューを使用してデータ書き出しに使用する文字セットを指定できます。
- 読み込みの場合、このメニューを使用して読み込まれたデータのエンコーディングを指定できます。書き出しファイルに BOM が含まれる場合、読み込みデータのエンコーディングは定義済みとなるので、このメニューは利用不可になります (次の段落参照)。

デフォルトで読み込みと書き出しに選択されるエンコーディングは UTF-8 または USE CHARACTER SET コマンドが実行済みならこのコマンドで指定された文字セットです。読み込みおよび書き出しダイアログボックスでエンコーディングを指定しても、アプリケーションのカレント文字セットは変更されない点に留意してください。

Note: **書き出し先プラットフォーム** メニューはレコードの終端文字を設定するために使用され、新しいオプション**自動** (書き出しプラットフォームに基づきレコード終端文字を設定)、**Unix** (終端文字 = line break) そして **カスタム** (区切り文字ページを表示) が含まれます。

Byte Order Mark の統合

BOM を生成オプションが標準の書き出しダイアログボックスの " ヘッダ " ページに追加されました :



このオプションはテキストフォーマットでデータを書き出す際にのみ利用できます (テキストまたは固定長テキスト)。

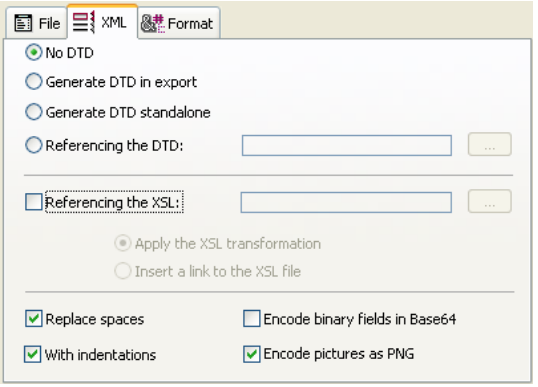
BOM を生成オプションが選択されていると、4D は書き出しファイルのヘッダに Byte Order Mark (BOM) を付加します。

この追加の情報により、BOM をサポートするアプリケーションではテキストの解釈が容易になります。

このオプションはデフォルトで選択されています。このオプションは書き出しに Unicode 文字セットが選択されているときにのみ有効です。そうでない場合 BOM は追加されません。

新しい XML 書き出しオプション

書き出しダイアログボックスの "XML" ページで、新しいオプションを利用できます :



- **バイナリフィールドを Base64 でエンコード** : このオプションを選択すると、4D は書き出されるバイナリフィールド (BLOB やピクチャタイプフィールド) に "data:;base64," ヘッダを追加します。このオプションが選択されていないと、フィールドは Base64 にエンコードされますが、ヘッダは付加されません。
- **インデントあり** : このオプションにより自動インデントが書き出されたデータに適用されます。インデントにより XML 要素の階層が視覚化されます。
- **PNG としてピクチャをエンコード** : このオプションを選択すると、書き出されるピクチャは元のフォーマットにかかわらず自動で PNG フォーマットにエンコードされます。このオプションが選択されないと、ピクチャはネイティブフォーマットにエンコードされます。
ピクチャが SVG で書き出されるとき、ピクチャのプロパティを保持するために、このオプションは選択しないことをお勧めします。

コンポーネントを 4D アプリケーションにインストール

以前のバージョンの 4D では、すべての 4D コンポーネントはそれを使用するデータベースごとに **Components** サブフォルダにインストールする必要がありました。つまり複数の異なる場所にインストールするために、コンポーネントを複製するか、エイリアスを使用する必要がありました。

4D バージョン 12 では、プラグインのように、4D や 4D Server アプリケーションに 4D コンポーネントをインストールすることができます。この方法でインストールされたコンポーネントは、その 4D アプリケーションで開かれるすべてのデータベースで自動的に利用することができます。

Components フォルダを配置可能な場所

4D v12 では、**Components** フォルダを 2 つの異なる場所に配置することができます :

- **実行可能な 4D アプリケーションレベル**
  - Windows: .exe ファイルと同階層
  - Mac OS: アプリケーションパッケージ内 Contents フォルダの直下
  この場合、コンポーネントはこのアプリケーションで開かれるすべてのデータベースで利用できます。
- **データベースストラクチャファイルと同階層** (以前の 4D バージョンで使用されていた場所)
  この場合、コンポーネントはそれがインストールされたデータベースでのみ利用できます。

どちらの場所を使用するかは、コンポーネントをどのように使用するかにより選択することができます。

両方の場所に同じコンポーネントがインストールされている場合、4D はストラクチャと同階層のコンポーネントのみをロードします。
コンパイルされ、4D Volume Desktop にマージされたアプリケーションの場合、同じコンポーネントの複数のインスタンスが存在すると、アプリケーションを開くことができません。

Note: 4D v12 では新しいプロパティを使用して、コンポーネントのプロジェクトフォームをサブフォームとしてホストデータベースに公開できます。この点に関する情報は 78 ページの " コンポーネント中のサブフォーム " を参照してください。

3 開発環境

この章では 4D v12 の開発環境の新機能と変更点について説明します。エディタ、環境設定、そして全体的なインタフェースが関連します :

- 環境設定の再編成
- ストラクチャエディタの新機能
- デザインモード検索機能の改修
- 新しい未使用要素の検索機能
- 新しいコードエディタ
- PHP コードの実行
- MSC でデータを復旧する新機能

フォームやフォームオブジェクトに関する新機能は、71 ページの Chapter 4 " フォームとオブジェクト " で説明しています。

環境設定の再構成

環境設定の管理は 4D v12 で完全に再構成されました。特にすべての 4D アプリケーションで共通なパラメタ (" ユーザ環境設定 ") と、データベースごとの設定 (" データベース設定 ") が分離されました。

そしてさらに、新しい設定が追加されました。

環境設定と設定

4D v12 では、2 つのダイアログボックス (" ユーザ環境設定 " と " データベース設定 ") を使用して、異なる 2 つのスコープの設定を行います。

旧バージョンの 4D の環境設定ダイアログボックスにあるオプションは、スコープとそれらが格納される場所に基づき 2 つのダイアログボックスにそれぞれ再配置されました (後述参照)。それぞれの機能は変わっていません。

Note: **テキスト比較の言語** オプションは環境設定ダイアログボックスの " 一般 " ページと設定ダイアログボックスの " データベース " ページ両方に表示されます。環境設定は新規データベース作成の際にデフォルトとして適用され、設定は開かれているデータベースの設定を変更します。

Note: このマニュアルでは、新しいオプションや機能が変更されたオプションのみを説明します。既存のオプションに関する詳細は 4D Design Reference マニュアルを参照してください。

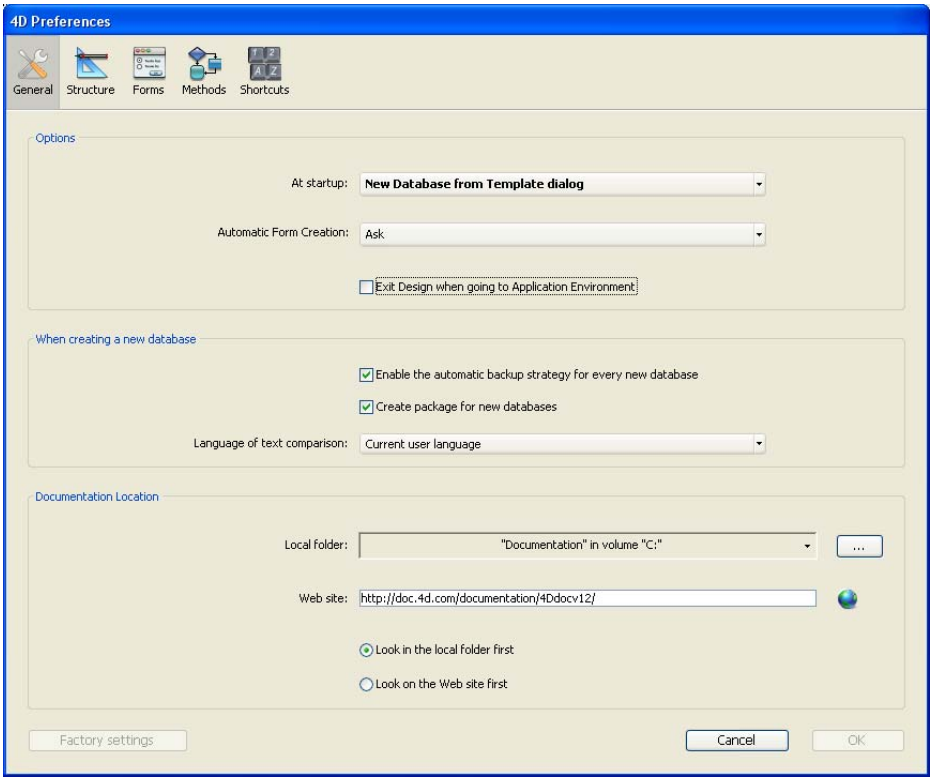
ユーザ環境設定

環境設定ダイアログボックスには、4D の**編集** (Windows) メニューあるいは**アプリケーション** (Mac OS) メニュー中の**環境設定 ...** コマンドからアクセスします :

このメニューコマンドは、データベースが開かれていない時でも選択できます。

ユーザ環境設定では、4D アプリケーション (4D および 4D Server) のデフォルトの振る舞いを指定します。一般的にこれらの設定はデベロッパの作業環境に影響します。例えばメソッドエディタのカラー設定、自動的なフォーム作成オプション設定などが含まれます :

4D ユーザ環境設定

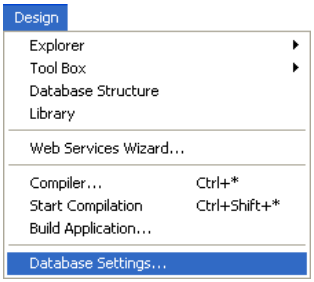


このダイアログボックスで行われた設定は XML フォーマットの環境設定ファイルに格納されます。このファイルは **4D v12 Preferences.4DPreferences** という名称で、カレントユーザの preferences フォルダに配置されます :

- **Windows XP**: C:\Documents and Settings\ ユーザ名 \Application Data\4D
- **Windows Vista/Windows 7**: C:\Users\ ユーザ名 \AppData\Roaming\4D
- **Mac OS**: {Startup disk}:Users: ユーザ名 :Library:Preferences:4D

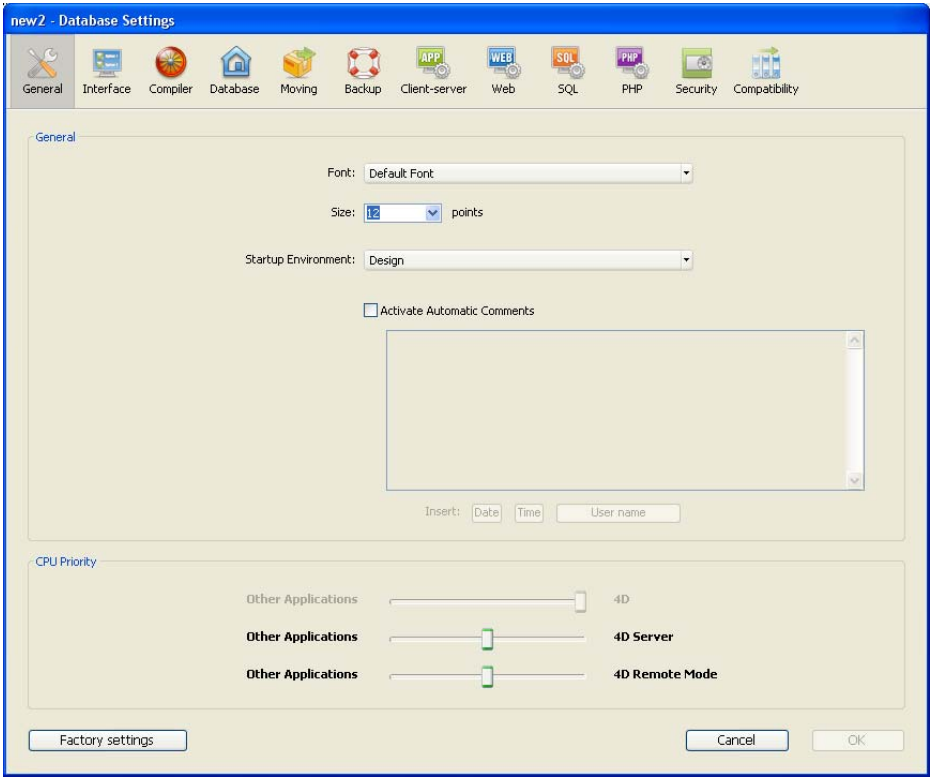
データベース設定

デザインメニューの**データベース設定 ...** コマンドまたは 4D ツールバーの対応するボタンからデータベース設定ダイアログボックスにアクセスできます :



データベース設定では現在開かれているデータベースの設定を行います。これらのパラメタはデータベースに格納され、データベースごとに異なります。設定内容には待ち受けポート番号、デザインモードへのアクセス権、SQL 設定などが含まれます :

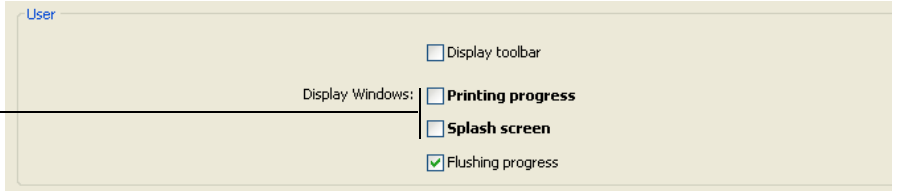
データベース設定



安全のため、データベース設定にはこのダイアログボックスからのみアクセスできます。

パラメタのカスタマイズと " 出荷時設定 "

環境設定および設定ダイアログボックスにおいて、値が変更されたパラメタのラベルは**太字**で表示されます :

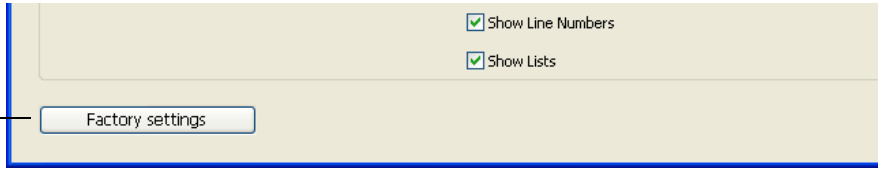


カスタマイズされたパラメタ

パラメタの変更はダイアログボックスや、旧バージョンから変換されたデータベースの場合以前に変更された内容が引き継がれることによって、起こります。

更新マークは、パラメタ値が手動でデフォルト値に置き換えられた際にも保持されます。これにより常にカスタマイズされたパラメタを識別できます。

パラメタをデフォルト値にリセットし、ラベルの更新マークを取り除くには、新しい**出荷時設定** ボタンをクリックします :



パラメタのリセット

このボタンをクリックすると、現在のページのすべてのパラメタがリセットされます。このボタンは現在のページで 1 つ以上のパラメタが変更されると、有効になります。

SET DATABASE PARAMETER コマンドとの相互作用

特定の設定 (PHP インタプリタの IP 番号など) はデータベース設定ダイアログボックスの他、SET DATABASE PARAMETER コマンドでも変更できます。

これらのツールはお互いに補完しながら動作します :

- データベース設定ダイアログで設定されたパラメタは常にすべてのマシンでデフォルトで使用されます。
- パラメタが SET DATABASE PARAMETER コマンドで更新されると、変更はそのマシン上でかつカレントのセッション中のみ有効です。データベース設定ダイアログボックスの設定値は変更されません。

  つまり、開発者は個別の条件に基づき、パラメタをローカルに変更できます (例えば別の PHP インタプリタを使用するなど)。

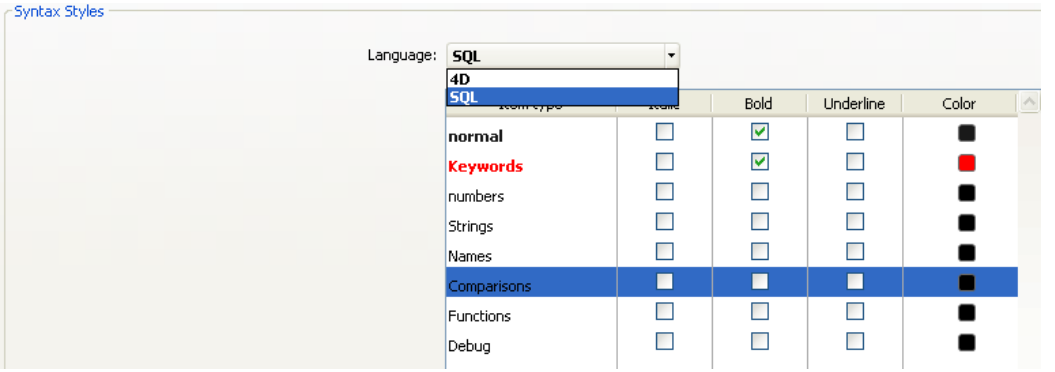
Note: SET DATABASE PARAMETER コマンドの特定の " 以前の " オプションは異なるスコープを持ちます。詳細はこのコマンドの説明を参照してください。

新しい環境設定

4D v12 では新しいユーザ環境設定を使用して、作業環境をカスタマイズできます。

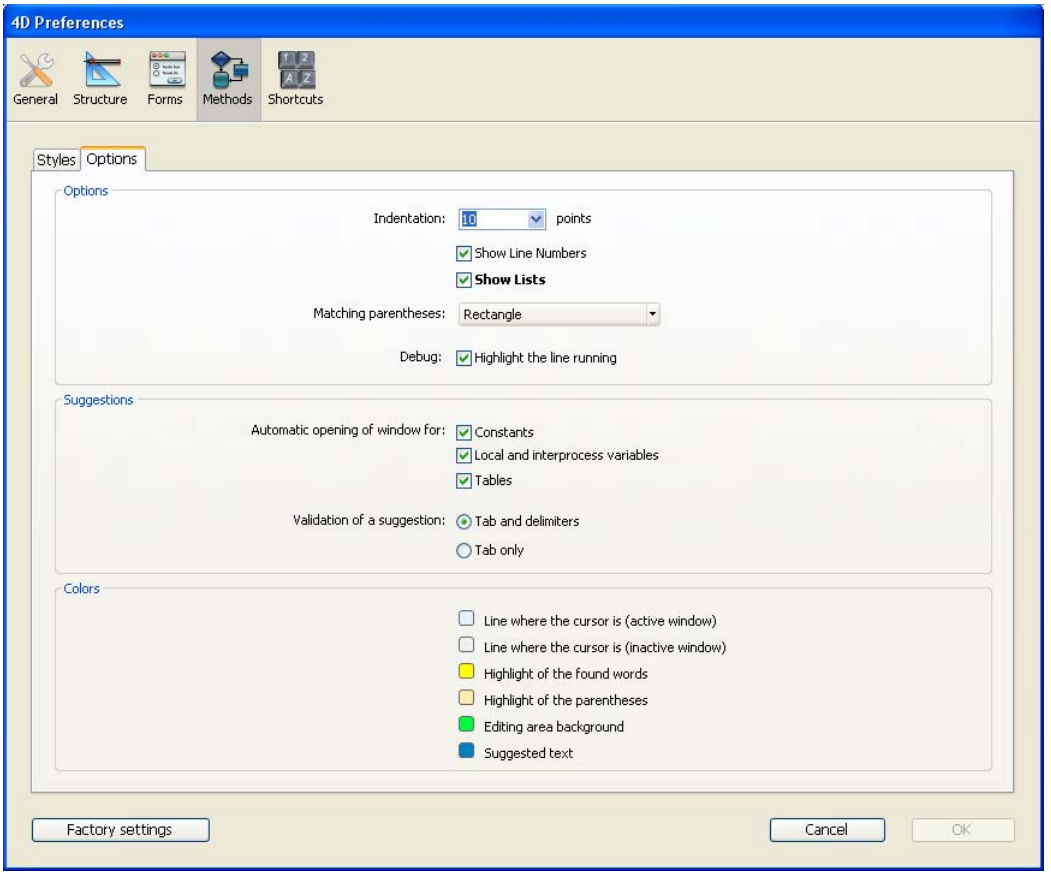
SQL シンタックス要素のスタイル

4D v12 の新しいコードエディタにおけるシンタックスの解析では、SQL ランゲージと 4D ランゲージの要素を分離します (55 ページの " 新しいコードエディタ " 参照)。環境設定のメソッドページ内、" スタイル " タブを使用して、それぞれのタイプの要素毎にカスタムスタイルを割り当てることができるようになりました :



メソッドエディタとデバッグオプション

環境設定のメソッドページ内 " オプション " タブではメソッドエディタとデバッガに関するオプションを設定できます :



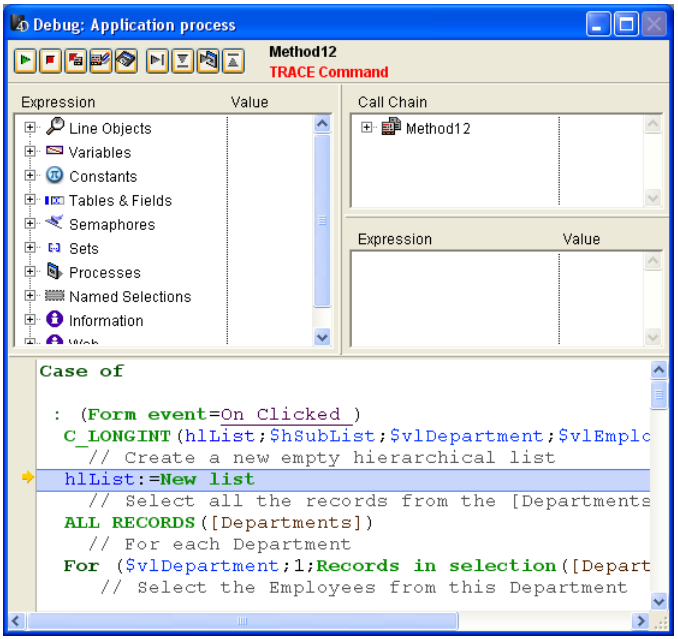
- **括弧のマッチング** : このオプションを使用して、コード中の対応する括弧を示すグラフィックサインを変更できます。このサインは括弧が一つ選択されると表示されます。以下のオプションが利用できます :
  - **なし** : サインなし
  - **四角** : 括弧を黒の線で囲む
  - **背景色**: 括弧をハイライト (色は " カラー " エリアで設定。37ページの " カラー " 参照)
  - **太字** : 括弧を太字で表示

  デフォルトでは四角オプションが選択されています。

```
INSERT MENU ITEM(main bar;-1;Get indexed string(79;1);FileMenu)
```

" 四角 " の表示

- **実行中の行をハイライト** : 通常の黄色矢印に加え、デバッガ中で、実行中の行をハイライトさせるために使用します :



このオプションの選択を解除すると、(以前のバージョンと同様に) 黄色の矢印だけが表示されます。

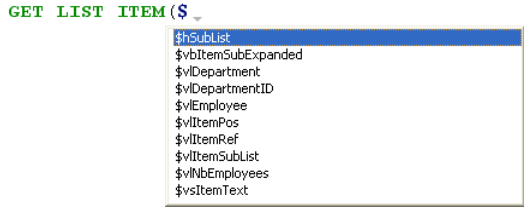
提案

自動コード補完メカニズムが 4D v12 で改善及び拡張されました (55 ページの " 自動コード補完 " 参照)。" 提案 " エリアではあなた自身の好みに応じて、このメカニズムを変更することができます。

互換性に関する注意

以前のバージョンの " タイプアヘッドを有効にする " オプションは削除され、この新しいオプションに置き換えられました。

- **自動でウィンドウを開く** :
このオプションは定数、ローカルやインタープロセス変数、およびテーブル用に自動で提案ウィンドウを表示するために使用します。
例えば " ローカルおよびインタープロセス変数 " オプションが選択されていると、$ 文字をタイプしたときに提案ウィンドウが表示されます :



対応するオプションの選択を外すことで、特定のランゲージ要素についてこの機能を無効にできます。

- **提案の受け入れ** :
このオプションは、自動コード補完ウィンドウに表示された現在の提案を、メソッドエディタが自動で受け入れる入力コンテキストを設定します。
  - **タブと区切り文字** : このオプションが選択されていると、タブキーまたはコンテキストに関連する区切り文字を押すことで、現在選択されている提案を受け入れることができます。例えば "ALE" の次に "(" と入力すると、4D は自動でエディタに "ALERT(" と書き込みます。使用することのできる区切り文字は以下の通りです :
(; : = < [{
この動作は以前のバージョンの 4D と同じです。
  - **タブのみ** : このオプションが選択されていると、現在の提案はタブキーを押したときにのみ挿入されます。この特別な動作は特に要素名に区切り文字を入力することを容易にします (例 : ${1})。

Note ウィンドウ中のダブルクリックまたはキャリッジリターンを使用することで、常に提案を受け入れることができます。

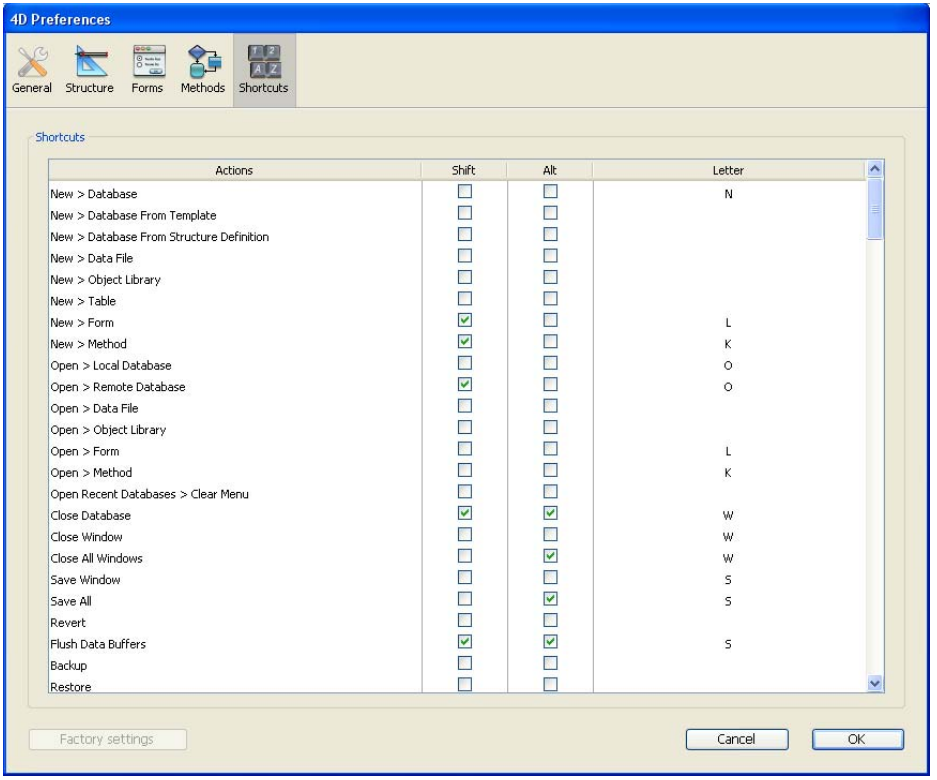
カラー

このオプショングループでは、メソッドエディタインターフェースで使用されるカラーを設定できます。

- **カーソルのある行 (アクティブウィンドウ) / カーソルのある行 (非アクティブウィンドウ)**: カーソルが置かれている行の背景色。
- **検索された語のハイライト** : 検索により見つけられた語のハイライト色。
- **括弧のハイライト** : 対応する括弧のハイライト色 (ペアの括弧がハイライトにより示される場合に使用されます。35 ページの " メソッドエディタとデバッグオプション " 参照)。
- **編集エリアの背景色** : メソッドエディタウィンドウの背景色。
- **提案テキスト**: メソッドエディタで表示される自動コード補完のテキストカラー。

ショートカット

ショートカットページでキーボードショートカットを編集できるようになりました:



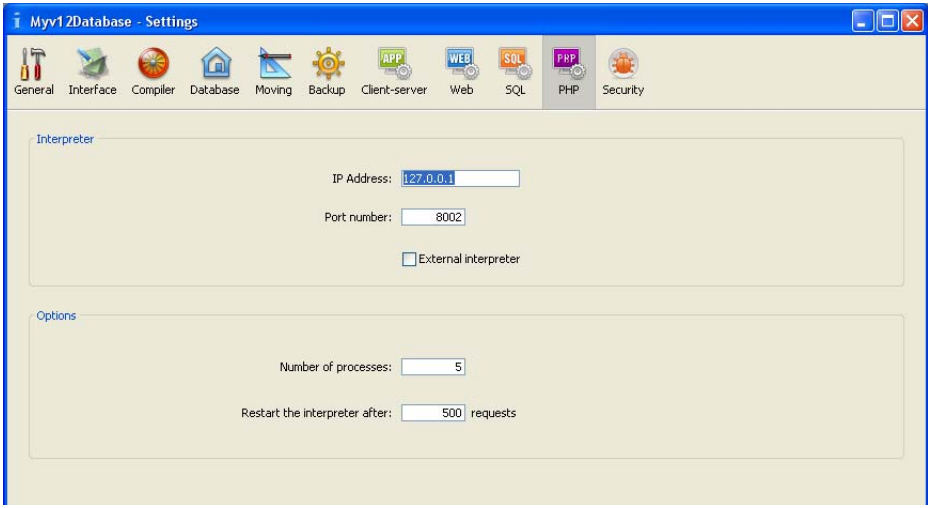
このページには 4D のデザインモードで使用されるすべてのショートカットが表示されます (ただし、例えば**コピー**コマンドに対応する Ctrl+C/Command+C のような、標準の " システム " ショートカットは除きます)。ショートカットを変更するには、ショートカットリスト中で、変更する項目の (Shift, Alt, または文字キー) を選択したり選択を外したりします。またはショートカットの行をダブルクリックして開かれるダイアログボックスで設定することもできます。それぞれのショートカットは暗黙のうちに **Ctrl** (Windows) または **Command** (Mac OS) を含むことに留意してください。

Note: このリストは [4D Extensions] サブフォルダ内の **4DShortcutsv12.xml** ファイルに基づいています。ダイアログボックスを使用してこのリストをカスタマイズすると、このファイルはユーザの Preferences フォルダに複製されて、これが標準設定の代わりに使用されます。この仕様により、4D が更新されても、カスタマイズされたキーボードショートカット設定は保持されます。

新しいデータベース設定

PHP

4D v12 では直接 PHP スクリプトを実行できます (62 ページの "4D で PHP スクリプトを実行する " 参照)。これはデータベース設定の PHP ページで設定します :



- インタプリタのアドレスとポート番号

  デフォルトで 4D が FastCGI でコンパイルされた PHP インタプリタを提供します。内部的なアーキテクチャのため、実行リクエストは指定された HTTP アドレスの PHP インタプリタに送信されます。デフォルトで 4D は IP アドレス 127.0.0.1 とポート番号 8002 を使用します。このアドレスやポートを変更できます。例えば既に他のサービスで使用されている、あるいは同じマシン上に複数のインタプリタをインストールしたい場合などです。これを行うには、データベース設定の PHP ページで **IP アドレス**と**ポート番号**パラメタを変更します。

  HTTP アドレスは 4D と同じマシンでなければなりません。

- 外部インタプリタ

  4D が提供するインタプリタとは異なる PHP インタプリタを使用できます。このインタプリタは FastCGI でコンパイルされていなければならず、4D と同じマシン上になければなりません (63 ページの " 他の PHP インタプリタや php.ini ファイルを使用する " 参照)。
  この場合、PHP リクエストが実行された際に 4D が内部インタプリタとの接続を開始しないようにするために、このオプションをチェックしな

ければなりません。この設定においては、あなた自身が実行を管理し、外部インタプリタを制御しなければならないことに留意してください。

- PHP オプション

これらのオプションは 4D が提供する PHP インタプリタの自動管理に関するものです。**外部インタプリタ**オプションがチェックされていると、これらのオプションは選択できなくなります。

- **プロセス数**: 4DのPHPインタプリタは、"子プロセス" と呼ばれる一連のシステム実行プロセスをコントロールします。最適化のために、デフォルトで最大 5 個の子プロセスを同時に起動でき、それらは PHP インタプリタにより永続的に保持されます。このオプションを使用して、必要に応じ子プロセス数を変更できます。例えば PHP インタプリタを集中的に呼び出す場合は、この値を増やす必要があるでしょう。詳細は 62 ページの " アーキテクチャ " を参照してください。

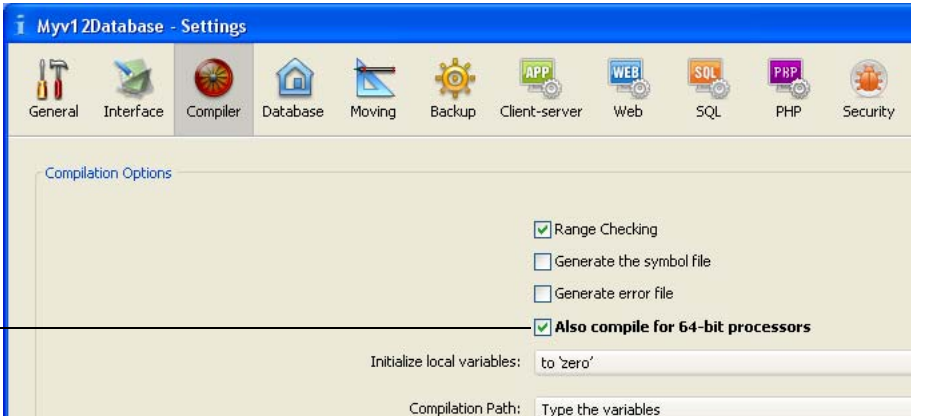
Note: Mac OS では、すべての子プロセスが同じポートを共有します。Windows では、それぞれの子プロセスが特定のポート番号を使用します。最初の番号は PHP インタプリタに設定されたもので、他の子プロセスはこの番号をインクリメントしたものを使用します。例えばデフォルトポート 8002 を使用して 5 つの子プロセスを起動すると、ポート番号 8002 から 8006 が使用されます。

- **X リクエスト後にインタプリタを再起動する**: 4DのPHPインタプリタが受け付ける最大のリクエスト数を設定するために使用します。この数値に達すると、インタプリタは再起動されます。このパラメタに関する詳細は FastCGI-PHP のドキュメントを参照してください。

Note: このダイアログボックスで、接続されたすべてのマシンおよびすべてのセッション用にデフォルトでパラメタが定義されています。マシンごとおよびセッションごとに、SET DATABASE PARAMETER, Get database parameter コマンドを使用して値を設定できます。SET DATABASE PARAMETER コマンドで変更された値はカレントセッションで優先的に使用されます。

64-bit コンパイル

4D v12 アプリケーションでは 32bit および 64bit プロセッサ用にコンパイルができるようになりました。これを行うために、データベース設定の " コンパイラー " ページで、新しい **64-bit プロセッサー用にもコンパイルする**オプションを利用できます :



新しいオプション

Note: - 64-bit バージョンの 4D Server v12 は現在ベータバージョンです。詳細情報は 303 ページの "64-bit 4D Server について " を参照してください。
- Unicode モードでない 4D データベースは、64-bit 用にコンパイルできません。

ストラクチャエディタ

ストラクチャエディタに、表示管理に関する新しい機能が追加されました。

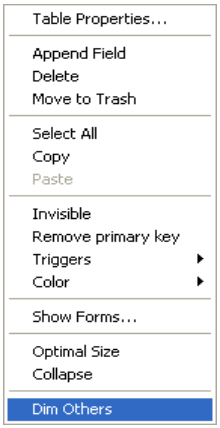
他を薄暗色にする

ストラクチャエディタ中で、選択されている以外のテーブルやリレーションを薄暗い色で表示できるようになりました。以前のバージョンの 4D ではテーブルをフォルダごとに薄暗くすることのみができました。

新しい**他を薄暗色にする**機能は以下で利用できます :

- エディタのフォルダ管理メニュー

- エディタのコンテキストメニュー (テーブルまたはリレーション上でクリック)



このコマンドを選択すると、選択されていない、または選択された要素に接続されていないテーブルやリレーションはエディタウィンドウ中で薄暗く表示されます。

すべてをハイライト

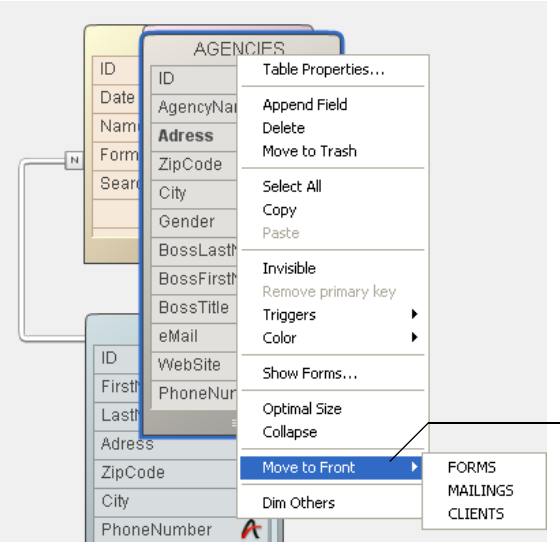
他を薄暗色にする コマンドやフォルダ管理メニューを使用して要素が薄暗く表示されているとき、新しい**すべてをハイライト**コマンドを使用してエディタのすべてのコンテンツを再表示できます。このコマンドはストラクチャエディタのコンテキストメニュー内にあります .

前面に移動

ストラクチャエディタ中で複数のテーブルが重なっていると、目的のテーブルを表示するのが困難になることがあります。

この場合、すべてのテーブルのトップにあるテーブル上で右クリックを行い、**前面に移動**メニューを表示させます。このメニュー内にはサブメ

ニューに、スタックされているすべてのテーブルが一覧されます。そして一覧から、最前面に表示したいテーブルを選択できます :

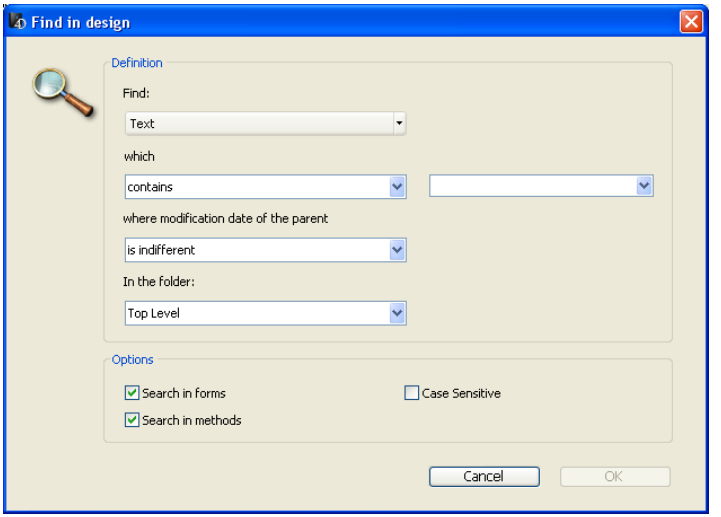


デザインモード検索の最適化

デザインモードの検索と置換のメカニズムが 4D デベロッパーのニーズによりマッチするよう、見直されました。この節では、4D v12 でこの機能に追加された新機能や変更点について説明します。

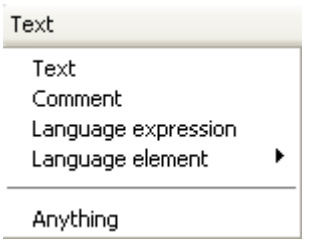
検索ダイアログボックス

検索ダイアログボックスは**編集**メニューの**デザインモードを検索**コマンドを選択することで表示されます。4D v12 では、このダイアログボックスで提供されるオプションが変更されました :



検索設定

タイプ : メニューを使用して検索する要素のタイプを指定できます。以下の選択肢が利用できます :



- **テキスト** : この場合、4D はデザインモード全体中で指定された文字列を検索します。
  検索はコンテキストを考慮しない、プレーンテキストモードで行われます。例えばテキスト "Alert("Error number:"+" や "button27" を検索できます。
  このモードではワイルドカード文字を使用できません。"@" は標準の文字として扱われます。
- **コメント** : このタイプの検索は基本的に先の検索と同じですが、検索対象がコメント内に制限されます (// で始まる行)。これにより例えば " 後で確認 " といった文字列を含むコメントを検索することができます。

Note: これら両タイプの検索の最終的な結果は、検索モード (**検索条件**) により異なります (45 ページの " 検索モード " 参照)。

- **ランゲージ式** : 有効な 4D 式を検索するために使用します。4D は検索の際に式を評価するので、有効な式を記述することが重要です。例えば式 "[clients" の検索は、式が無効なため成功しません。"[clients]" と記述する必要があります。
  このオプションは値の代入や比較の検索に対して行われます。例えば :
  - "myvar:=" (代入) の検索
  - "myvar=" (比較) の検索
- **ランゲージ要素** : 特定のランゲージ要素を検索する場合に使用します。4D は以下の要素を区別できます :
  - **プロジェクトメソッド** : "M_Add" などのプロジェクトメソッド名。(**等しい**モードを割り当てた) この検索は、コードエディタでの**参照を検索**と同じです (60 ページの " 参照検索 " 参照)。
  - **フォーム** : "Input" などのフォーム名。検索はプロジェクトフォームおよびテーブルフォームに対して行われます。
  - **フィールドまたはテーブル** : "Customers" などのフィールドやテーブル名。
  - **変数** : "$myvar" など。
  - **4D 定数** : "Is Picture" など。
  - **引用符内の文字列** : リテラルテキスト定数。つまりはコードエディタ中で引用符に囲まれた値や、フォームエディタのテキストエリア (スタティックテキストやグループボックス) に挿入された値。例えば "Martin" を検索すると以下のコードがヒットします :
    QUERY ([Customers];[Customers]Name="Martin")
  - **コマンド** : "ALERT" などの 4D コマンド。
  - **プラグイン** : "WR Find" などの、4D アプリケーションにインストールされたプラグインコマンド。
- **すべて** : このオプションを選択すると、デザインモード中のすべてのオブジェクトを対象に検索が行われます。この場合、更新日メニューのみが利用できます。特にこのオプションは、例えば " 今日更新したすべてのオブジェクト " を検索するために使用します。

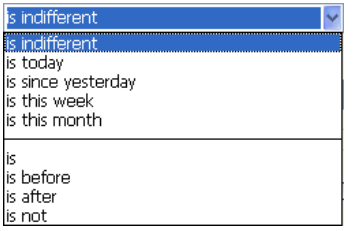
検索モード

検索モードメニュー (指定された検索タイプによって、**検索条件や名前**などに変わります) は入力された値がどのように検索されるかを指定するために使用されます。その内容はタイプメニューで選択された要素タイプに基づいて変わります。

- **テキスト**や**コメント**を検索 : このタイプの検索が選択されていると、メニューには以下のオプションが含まれます :
  - **含む**: デザインモードのテキスト中でその文字列を含むオブジェクトを検索します。"var" を検索すると "myvar" や "variable1"、"aVariable" がヒットします。
  - **語全体を含む** : デザインモードのテキスト中でその文字列を検索します。"var" を検索するとその名前のオカレンスのみがヒットします。"myvar" は検索されません。他方 "var:=10" や "ID+var" は、記号 : や + が語の区切り文字であるため、検索されます。
  - **前方一致 / 後方一致**: 単語 (テキスト検索) やコメント (コメント検索) がその文字列で始まるまたは終わるものを検索します。テキストの後方一致検索の場合、"var" を検索すると "myvar" はヒットします。
- **ランゲージ要素**の検索 : このタイプの検索が選択されていると、メニューには標準のオプション (**等しい**、**含む**、**前方一致**、**後方一致**) が提供されます。これらのオプションは以前のバージョンの 4D で提供されていたものと同じです。
  等しいオプションではワイルドカード "@" を使用することができます (指定したタイプのオブジェクトすべてが返されます)。

親オブジェクトの更新日

このメニューには、検索の期間を素早く指定できる、いくつかの新しいオプションが含まれます :

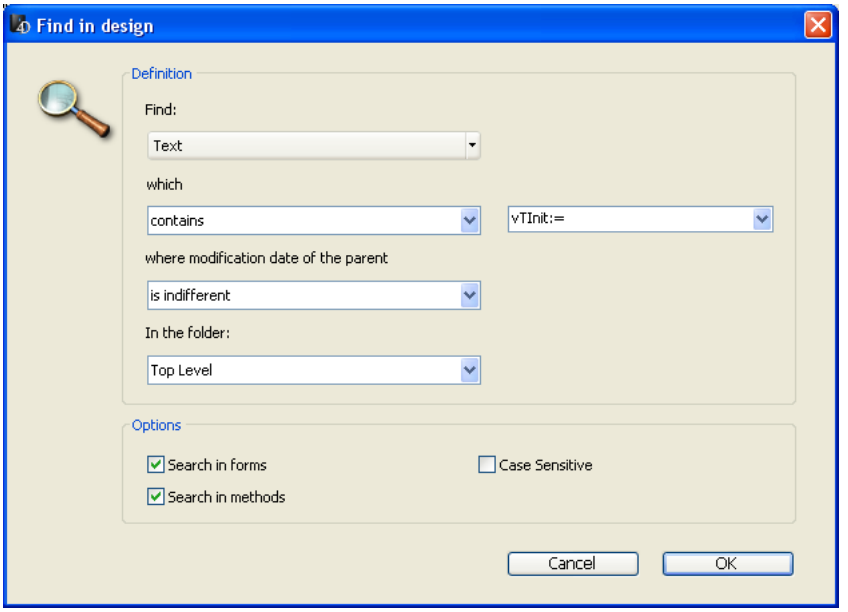


- **本日** : 本日の深夜 00:00 以降。
- **昨日以降** : 昨日と本日。
- **今週** : 月曜日から始まる今週。
- **今月** : 1 日から始まる今月。

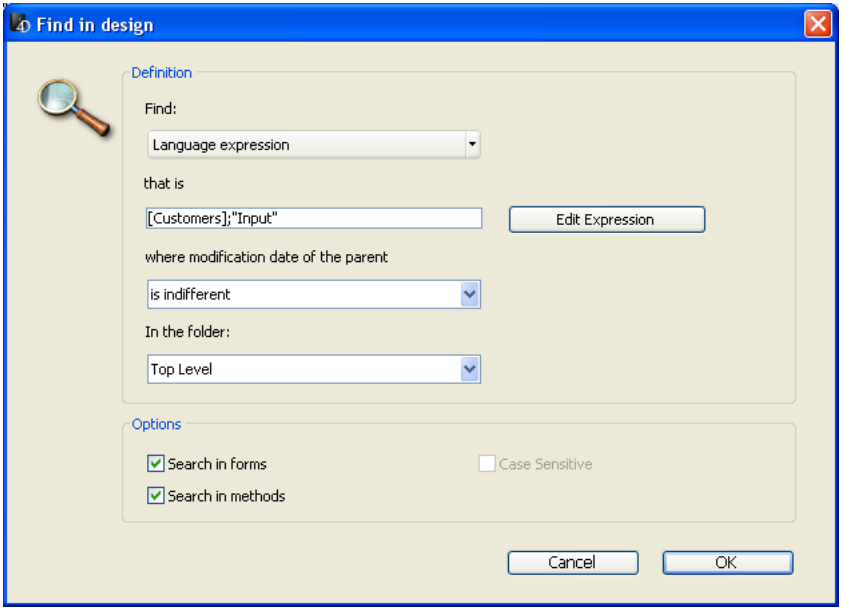
検索例

効果的に検索を行うには、正しい**タイプ**オプションおよび検索モード (**検索条件**) メニューの組み合わせを選択することが重要です。4D v12 での検索の動作を説明するために、典型的な検索の例とそれをどのように設定するかを示します。

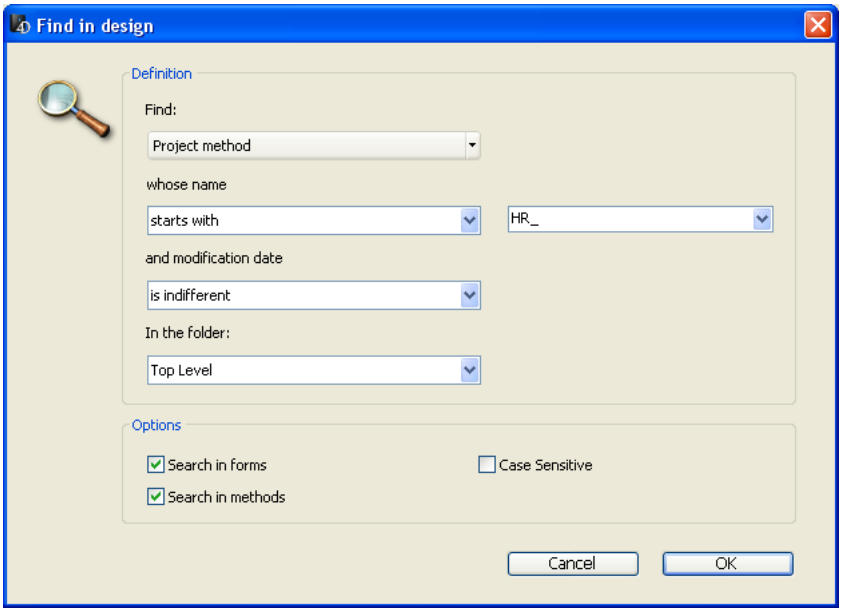
- 値が vTInit 変数に直接代入されている箇所をすべて検索する例 :



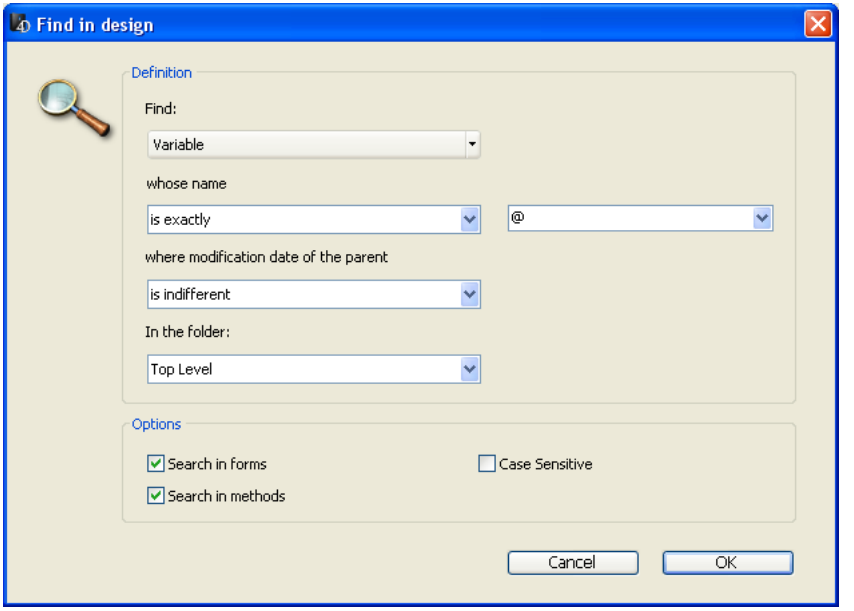
- [Customers] テーブルの "Input" フォームが参照されている場所をすべて検索する例 :



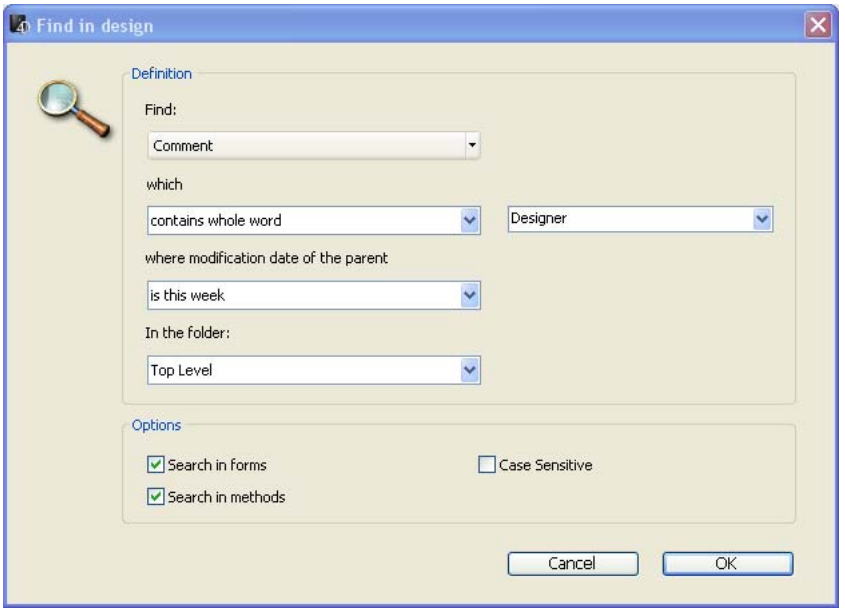
- メソッド名が "HR_" で始まるメソッドの呼び出しをすべて検索する例



- データベースのすべての変数をリストする例 :



- 今週書かれたコメント中で、"Designer" キーワードを検索する例 :



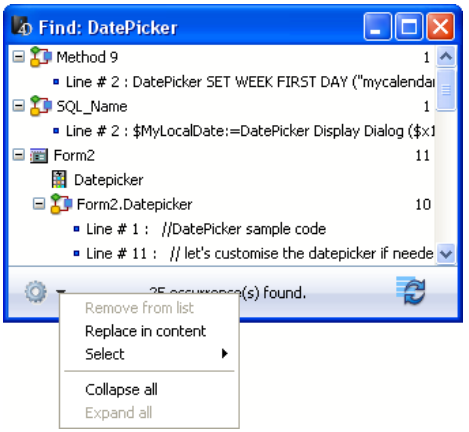
置換と名称変更

名称変更と置換機能が 4D v12 でより簡易になりました。本バージョンからは２つのコマンドを使用できます :

- **内容を置換**は検索結果ウィンドウにあります。このコマンドはリストされたオブジェクトの内容を別の文字列に置換するために使用されます。
- **名称変更**はエクスプローラと新しいコードエディタにあります。このコマンドはプロジェクトメソッドや変数の名称を変更し、すべての呼び出し元を新しい名前で置換するために使用されます。

内容を置換

すべての置換は検索結果ウィンドウのオプションメニュー内にある**内容を置換**を使用して行われます :

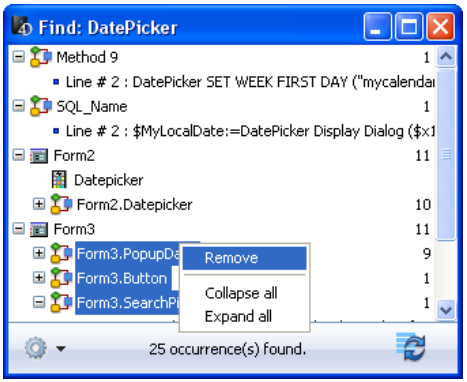


このコマンドを選択すると文字列を入力するためのダイアログが表示され、検索の結果見つかったオカレンスがその文字列で置き換えられます :

置換は以下の原則に基づいて動作します :

- 置換は選択された項目にだけでなく、常に検索されたすべての項目に対して実行されます。置換対象を絞り込むために、コンテキストメニュー

から**削除**コマンドを使用することができます。:



一覧中に表示されたオカレンスのみが置換され、また最初の検索から置換操作までの間にオブジェクトが変更される場合に備え、最初の検索条件を検証したのちに置換を行います。

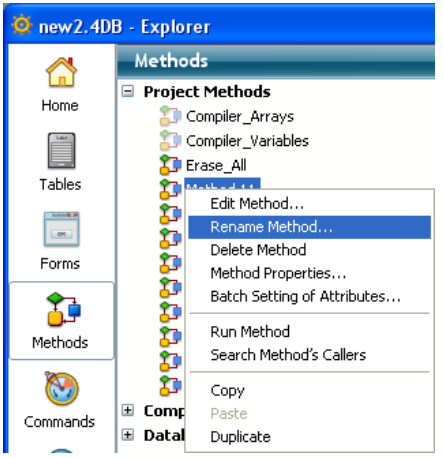
- 置換は以下に対して実行されます :
  - メソッドの内容
  - フォームオブジェクトのプロパティ
  - ヘルプメッセージの内容
  - 入力フィルタの内容
  - メニュー項目の内容 (項目テキストとメソッド呼び出し)
  - 選択リストの内容
  - エクスプローラ中のメソッド、フォーム、テーブル、そしてフィールドのコメントの内容
- 変更するオブジェクト毎に、4D はそれが他のマシンや他のウィンドウにロードされているかを検証します。コンフリクトが発生している場合は、オブジェクトがロックされている旨を知らせる標準のダイアログが表示されます。オブジェクトを閉じて再試行するか、置換をキャンセルできます。置換処理はその後リスト中の他のオブジェクトに対し続行されます。
- " 内容を置換 " 処理に関連するメソッドやフォームが同じ 4D アプリケーション内で既に編集されている場合、それは開かれたエディタ内で直接更新されます (警告は表示されません)。このように更新されたフォームやメソッドは、自動では保存されません。この変更を有効にするために、明示的に**保存**や**すべてを保存**コマンドを使用する必要があります。
- 置換がリストの項目に対して行われた場合、この項目はイタリックで表示されます。置換数はリアルタイムでウィンドウの下部に表示されます。

- フォームオブジェクトを除き、オブジェクトが**内容を置換**機能で名称変更されることはありません。結果、リスト中の特定の項目が置換処理の対象とならないことがあります。これは、項目の名前のみが最初の検索条件に対応する場合に発生します。この場合、リストの項目はイタリックで表示される必要がなく、置換された項目数は最初の検索で見つかった項目数より少なくなります。

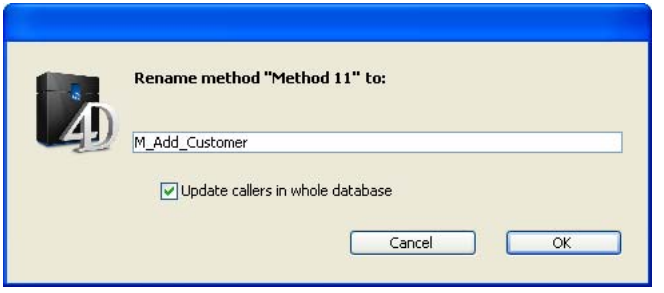
名称変更

データベース全体にわたって行われる名称変更は、**プロジェクトメソッド**と**変数**に対して行われます :

- メソッドエディタ内のコンテキストメニューから**名称変更 ...** コマンドを使用する (プロジェクトメソッドと変数)。
- エクスプローラのコンテキストメニューから**メソッド名称変更 ...** コマンドを使用する (プロジェクトメソッド).



コマンドを選択するとダイアログボックスが表示され、オブジェクトの新しい名前を入力できます :



この新しい名称は命名規則に沿っていなければなりません。そうでなければダイアログボックスを受け入れる際に警告が表示されます。例えば

"ALERT" など、コマンド名と同じ名称をメソッドに与えることはできません。

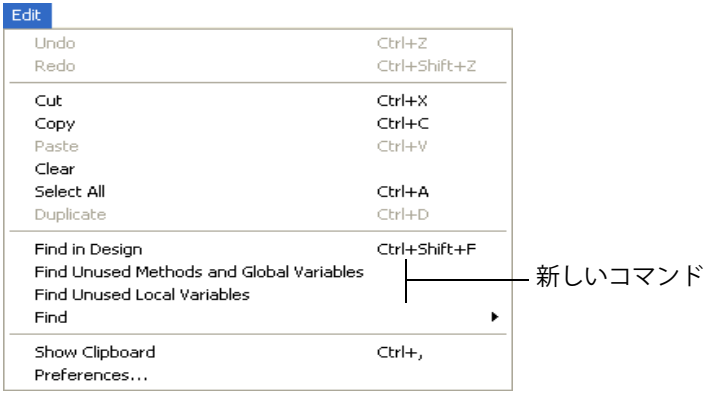
名称変更するオブジェクトのタイプに基づき (プロジェクトメソッドまたは変数)、名称変更ダイアログボックスには追加のオプションが表示されます :

- **プロジェクトメソッド** : **データベース全体で呼び出し元を更新**オプションを使用して、そのメソッドを参照するすべてのデータベースオブジェクトに対し、名称変更を適用できます。このオプションのチェックを外せば、例えばエクスプローラ内でのみメソッド名を変更できます。
- **プロセスおよびインタープロセス変数** : **データベース全体で変数を名称変更**オプションを使用して、その変数を参照するすべてのデータベースオブジェクトに対し、名称変更を適用できます。このオプションのチェックを外せば、カレントメソッド内でのみメソッド名を変更できます。
- **ローカル変数** : このオブジェクトに対しては追加のオプションは表示されません。ローカル変数はカレントメソッド内でのみ名称変更されます。

未使用の要素を検索する

新しい 2 つの検索コマンドを使用して、コード中で使用されていない変数やメソッドを検知することができます。これにより、例えばメモリを解放する目的で、それらを削除できます。

これらのコマンドはデザインモードの**編集**メニューにあります :



未使用のメソッドおよびグローバル変数を検索

未使用のメソッドおよびグローバル変数を検索コマンドは定義されているが使用されていないプロジェクトメソッドとグローバル変数 (プロセスおよびインタープロセス変数) を検索します。検索結果は標準の結果ウィンドウに表示されます。

プロジェクトメソッドは以下の場合に未使用と判断されます :

- ゴミ箱の中に入っていない
- いずれの 4D コードからも呼ばれていない
- メニューコマンドから呼ばれていない
- 4D コード中で文字定数として呼ばれていない (括弧つきの引数リストが後に続いていても、4D は文字列中のメソッド名を検知します。)

プロセスおよびインタープロセス変数は以下の場合に未使用と判断されます :

- 4D コード中で C_XXX や ARRAY XXX などのコマンドで定義されている
- いずれの 4D コード中でも使用されていない
- いずれの 4D フォーム中でも使用されていない

この機能で検知することのできない使用ケースがある点に留意してください。実際使用されているにもかかわらず、未使用と判断されることもあります。例えば以下のケースです :

```
v:="method"
EXECUTE FORMULA("my"+v+String(42))
```

このコードはメソッド名を動的に構築しています。プロジェクトメソッド mymethod42 は、実際に呼び出されますが、未使用と判断されます。そのため、実際に削除を行う前には、未使用と指定された要素が本当に必要のないものであるかどうかを検証することをお勧めします。

未使用のローカル変数を検索

未使用のローカル変数を検索コマンドは、定義されているのに使用されていないローカル変数を探します。検索結果は標準の結果ウィンドウに表示されます。

ローカル変数は以下の場合に未使用と判断されます :

- 4D コード中で C_XXX や ARRAY XXX などのコマンドで定義されている
- 同じメソッド内のいずれの場所でもそれが使用されていない

新しいコードエディタ

4D v12 で、開発者の生産性を向上させる新しい機能を追加することにより、コードエディタが拡張されました。

特に、4D コードでも SQL コードでも、自動コード補完やデバッグ等で、同様の使いやすさが提供されます。

```
Begin SQL
  SELECT Activity.ActivityCode
  FROM Activity
  WHERE (ActivityManager="Johnson")
   OR (ActivityManager="Johnson" AND ActivityDesignation="Tennis");
End SQL
```

4D v12 のエディタは、4D v11 SQL のエディタの特徴やツールをほとんど継承しつつ、さらに機能が追加されています。このマニュアルでは新しい機能と相違点のみを説明します。エディタの基本機能については 4D v11 SQL の Design Reference マニュアルを参照してください。

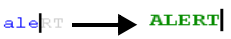
入力アシスト

4D v12 でいくつか新しい入力アシスト機能とコード編集機能が追加されました。

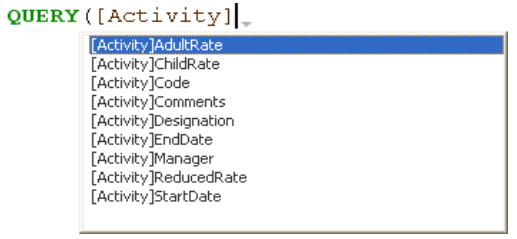
自動コード補完

自動コード補完が SQL コードにも適用されるようになり、また既に入力されたオプションの "*" などの引数が考慮されるようになりました。2 つのあたらしい機能が提供されます :

- 入力された文字が、可能性のある値 1 つに対応する場合、それがグレイアウトされた状態で表示されます (そして **Tab** キーを押すと挿入されます):



- 引数の自動コード補完は変数さらにはテーブルやフィールド名に拡張され、既に入力されたオプションの "*" などの引数が考慮されるようになりました。



Note: 自動コード補完のメカニズムはアプリケーションの環境設定で設定できます (36 ページの " 提案 " 参照)。

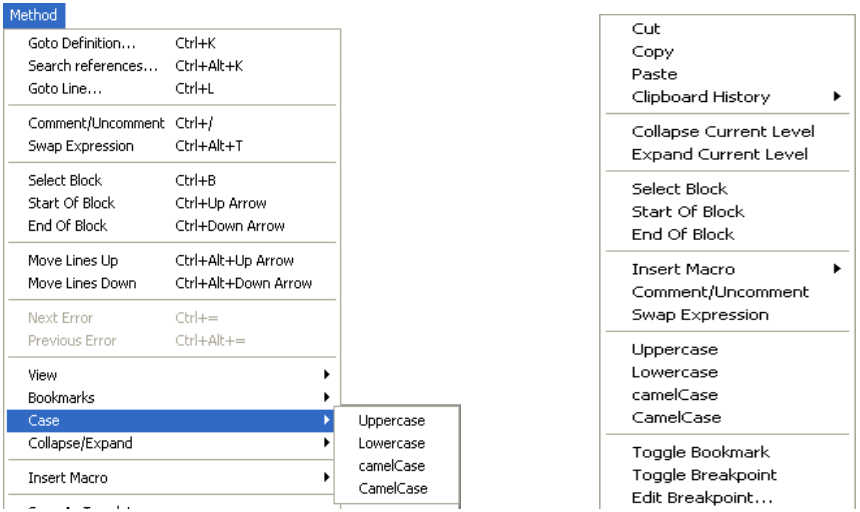
選択

コードエディタに新しい選択機能が追加されました :

- " 語句 " を選択するためにダブルクリックを利用できます。参照される要素の名前 (コマンド、定数、メソッド等) がスペースを含む場合、それ全体を選択するためには **Alt/Option** + **Double-click** のコンビネーションを使用します。
- 最初に選択する必要なく、直接カーソルを含む行を移動できます。これを行うには **Alt/Option** + **Up Arrow** または **Down Arrow** のコンビネーションを使用します。この機能は**メソッド**メニュー内の新しい**行を上に移動**と**行を下に移動**コマンドでも利用できます。
- ブロック選択機能が 2 つの新しいコマンド、**ブロックの開始**と**ブロックの終了** (**メソッド**メニュー) で拡張されました。これらを使用して、カーソルを (**If...End if** タイプの論理構造で分割される) コードブロックの開始または終了位置に置くことができます。.
さらに、**ブロックを選択**コマンドは括弧やブラケットなどの囲み文字を除くブロックを選択するようになりました。

大文字小文字の変更

選択された文字の大文字や小文字を変更するコマンドが複数追加されました。これらのコマンドは**メソッド** メニューの**大文字 / 小文字** サブメニューまたはエディタのコンテキストメニュー内にあります :



- **大文字 / 小文字** : これらのコマンドは選択された文字列の大文字と小文字をスイッチします。
- **キャメルケース先頭小文字 / キャメルケース先頭大文字** : これらのコマンドを使用して、選択された文字列をキャメルケースにスイッチできます。これは付加された単語の先頭文字を大文字にします。このタイプの記法はしばしば変数の命名規則として使用されます。hireDate と

PurchaseDate はキャメルケース記法の 2 つのバリエーションです。選択したテキストにこれらのうちいずれかのコマンドを適用すると、スペースと "_" 文字は取り除かれ、単語の先頭が大文字に変換されます。

複数行への入力

1 つの文を複数行に記述できるようになりました。これを行うには各行の終わりをバックスラッシュ "\" (Windows で日本語フォントを使用した場合 円マーク) で終了させます。4D はすべての行を一文として扱います。例えば以下の 2 つの文は完全に同じです :

```
[DOCS]Plural:=Uppercase([DOCS]Plural[[1]])+Lowercase(Substring([DOCS]Plural;2))

[DOCS]Plural:=Uppercase([DOCS]Plural[[1]])+\
Lowercase(Substring([DOCS]Plural;2))
```

折りたたみ / 展開

メソッドメニューの新しいコマンドを使用してコードの一部を折りたたんだり展開したりできます。

Note: これらのアクションは以前のバージョンの 4D で折りたたみ / 展開と呼ばれていました。

- **選択範囲を折りたたむ / 展開する** : 選択されたテキストに含まれるすべてのコードストラクチャを折りたたんだり展開したりするために使用します。
- **現レベルを折りたたむ / 展開する** : カーソルがあるレベルのコードストラクチャを折りたたんだり展開したりするために使用します。これらのコマンドはエディタのコンテキストメニューでも利用できます。

IME (Windows)

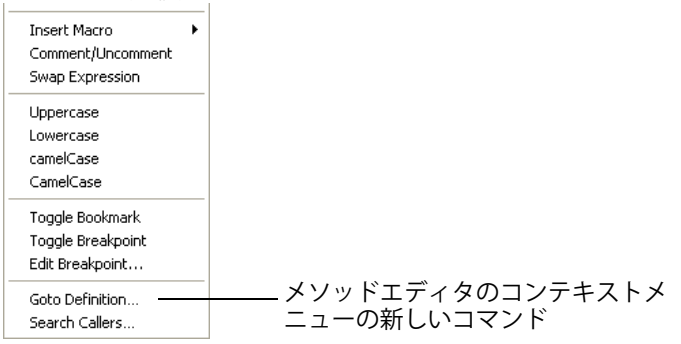
Windows で、新しいコードエディタは日本語および中国語を編集する際の Input Method Editor (IME) をサポートしました。

ブラウズ

エディタで開く

新しい**エディタで開く**コマンドを使用して、コードエディタ内で参照されるオブジェクト (メソッド、フィールド、テーブル、またはフォーム) の定義を、直接新しいウィンドウ内で開くことができます。これを行うにはオブジェクト名にカーソルを置き (または選択して)、**メソッド** メ

ニュー内またはエディタのコンテキストメニュー内の**エディタで開く ...** コマンドを選択します :



このコマンドは以下のオブジェクトに対して動作します :

- プロジェクトメソッド : メソッドの内容を新しく開いたメソッドエディタ内に表示します。

Note: この機能は、プロジェクトメソッド名が選択されているとき、キーボードショートカット **Ctrl+K** (Windows) または **Command+K** (Mac OS) でも使用できます。

- フィールド : ストラクチャウィンドウのインスペクタにフィールドのプロパティを表示します。
- テーブル : ストラクチャウィンドウのインスペクタにテーブルのプロパティを表示します。
- フォーム : フォームエディタに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): 現在のメソッドあるいはコンパイラメソッド中で変数定義行を表示します。

ブロックの先頭位置や終了位置への移動

2 つの新しいコマンドでコード構造 (If...Else...End if 等) 内のブラウズが容易になります :

- **ブロックの先頭** : このコマンドを選択するとカーソルは現在の構造の開始キーワードの前に移動します。
- **ブロックの終了** : このコマンドを選択するとカーソルは現在の構造の終了キーワードの後ろに移動します。

これらのコマンドは**メソッド**メニューおよびエディタのコンテキストメニューから利用できます。また以下のショートカットも使用できます :

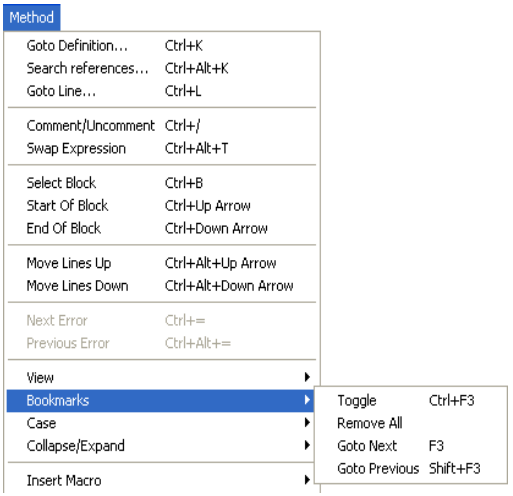
- Windows: Ctrl + ↑ または Ctrl + ↓
- Mac OS: Command + ↑ または Command + ↓

ブックマーク

4D v12 では特定の行にブックマークを割り当てることができます。特定のコマンドを使用して、あるブックマークから他のブックマークへと、コード内を素早くブラウズできます。

一度ブックマークが設定されると、追加の行がその前に挿入されても、そのブックマークは設定された行とともに移動します。ブックマークはメソッドとともに保存されます。

ブックマークは**メソッド**メニューの**ブックマーク**サブメニューで管理します:



- **切り替え** : (カーソルのある) 現在の行に、まだブックマークされていなければブックマークを割り当てます。すでにブックマークされていればそれを取り除きます。
この機能はエディタのコンテキストメニュー内、**ブックマークをトグル**コマンドから、あるいは **Ctrl+F3** (Windows) または **Command+F3** (Mac OS) キーボードショートカットからも利用できます。
- **すべて取り除く** : カレントのウィンドウからすべてのブックマークを取り除きます。
- **次に移動** / **前に移動** : これらのコマンドを使用して、ウィンドウ内に設定されたブックマーク間を移動できます。これらのコマンドを選択すると、カーソルはブックマークされた行の先頭文字位置におかれます。キーボードショートカット **F3** (次に移動) または **Shift+F3** (前に移動) も使用できます。

参照検索

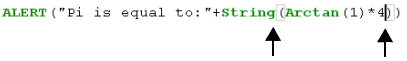
メソッドメニューやメソッドエディタのコンテキストメニュー内の**参照検索 ...** コマンドは、以前のバージョンの 4D の**呼び出し元を検索**コマンドを置き換えます。**呼び出し元を検索**は選択されたサブメソッドを参照するオブジェクト一覧を、検索結果ウィンドウに表示しました。

参照検索 ... コマンドはこの機能を保持しつつ、検索対象を変数やフィールドに広げます。これにより、選択された変数やフィールドが参照されているすべてのメソッドを検索できます。

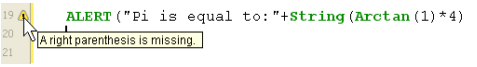
表示

新しい表示方法により、コードの可読性とメンテナンスが向上します。

- 囲み文字 (括弧やブラケット) の次をクリックすると、(開始 / 終端文字とともに) 灰色の四角で囲まれます :



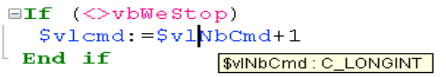
- エラーは編集エリアの左側にアイコンとヘルプ TIp 付きで示されます :



- コードブロックが折りたたまれているとき、最初の一行だけが表示され、展開ボタンが自動で表示されます。:

展開ボタン上にマウスポインタを重ねると、隠されたコードの一行目がヘルプ Tip に表示されます。

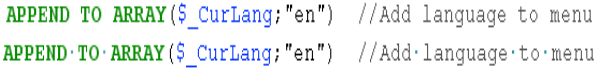
- ヘルプ Tip には変数の型が表示されるようになりました :



- 単語間のスペースを、" 空白スペース " でなく、記号を使用して表示できるようになりました。この機能はすべてのコード要素 (コマンド名、変数、コメント等) に適用され、**メソッド**メニューの**表示 > 空白**コマンドで利用できます :

空白スペース

記号のスペース



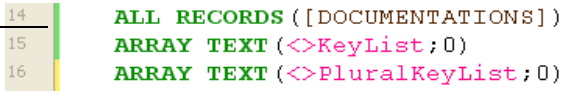
- エディタウィンドウの右下にあるステータスバーにカーソル位置が表示されます :

  - **行** : 行番号
  - **階層** : プログラムストラクチャの階層間レベル。最初の階層は 0 です。階層番号はデバッグの際にコード中でエラーが発生したとき、インタプリタによって提供される情報となります。
  - **文字** : 行内の文字の位置

- **メソッド > 表示**サブメニュー内の新しい**大きなフォント**と**小さなフォント**コマンドを使用して最前面のウィンドウのフォントサイズを一時的に変更できます。この変更はウィンドウを閉じた後は保持されません。永続的にメソッドエディタのフォントサイズを変更するには、アプリケーション環境設定の " メソッド " ページを使用します。

- **変更バー** : 色つきのバーにより、メソッドが開かれてから変更された行が一目でわかるようになります :

変更バー



変更バーの色は更新が保存されたかされていないかで変わります :

- 黄 : 行が変更され、メソッドはまだ保存されていない。
- 緑 : 行が変更され、その後メソッドが保存された。

コメント

コードのコメントはダウブルスラッシュ (//) で記述するように変更されました。行またはブロックのコメント / アンコメントはショートカット **Ctrl+/** (Windows) または **Command+/** (Mac OS) を使用して行います。

Note: 互換性のため、` 文字も行をコメント化するために使用できます。

4D で PHP スクリプトを実行する

4D v12 で PHP スクリプトを直接実行できます。これにより PHP で利用できる豊富なユーティリティライブラリを使用できるようになります。特に以下のような機能がライブラリに含まれます :

- 暗号化 (MD5) およびハッシュ
- ZIP ファイルの処理
- ピクチャの処理
- LDAP アクセス ,
- COM アクセス (MS Office ドキュメントのコントロール)...

このリストに挙げるものだけではありません。デフォルトで 4D で利用可能な PHP モジュールに関する完全なリストは 307 ページの appendix A、"PHP モジュール " を参照してください。また追加のカスタムモジュールのインストールも可能です (313 ページの " 追加モジュールのインストール " 参照)。

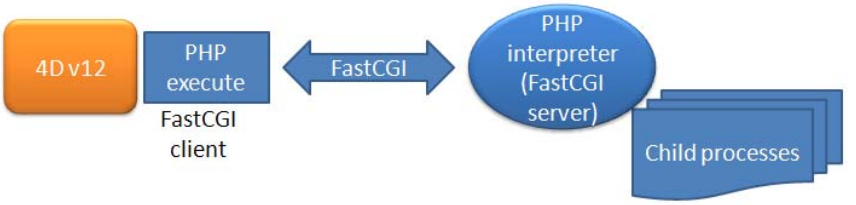
アーキテクチャ

4D は、アプリケーションと PHP インタプリタ間のクライアント - サーバタイプのコミュニケーションプロトコルである、FastCGI でコンパイルされた PHP インタプリタを提供します。

PHP インタプリタは " 子プロセス " と呼ばれる一連のシステム実行プロセスをコントロールします。これらのプロセスは 4D から送信されたリクエストを処理するために使用されます。リクエスト実行は同期的に行われます。最適化のため、デフォルトでは最大 5 つまでの子プロセスを同時に実行できます。この数値はデータベース設定や SET DATABASE PARAMETER コマンドで変更できます。Mac OS では、これらのプロセスは最初のリクエストで起動され、PHP インタプリタにより永続的に保持されます。Windows では、必要に応じて 4D がプロセスを作成し、必要な時にリサイクルします。4D はデフォルトで、起動と終了の PHP インタプリタのプロセスの処理を自動的にサポートします。

Note: PHP 子プロセスが動作中である間に 4D プログラムが予期せず終了した場合、システムプロセス管理ウィンドウから手動でそれらを削除しなければなりません。

下図は 4D v12 の 4D/PHP アーキテクチャを示しています :



このアーキテクチャは、4D から定義済みの TCP アドレス (IP アドレスとポート番号) に送信される内部リクエストのシステムで動作します。同じマシン上で複数の PHP インタプリタが動作しているなどの場合は必要に応じて、データベース設定 (39 ページの "PHP" 参照) またはプログラム (SET DATABASE PARAMETER, Get database parameter コマンド参照) を使用してこのアドレスを変更できます。

Note: 同じマシン上で 4D アプリケーションを 2 つ起動し、それぞれで (PHP Execute コマンドを使用して) PHP ステートメントを実行する場合、FastCGI インタープリタの待ち受けポートを変更して、それぞれのアプリケーションが異なるポートを使用するようにしなければなりません。そうでなければ PHP ステートメントの実行は予期せず失敗することがあります。

他の PHP インタプリタや php.ini ファイルを使用する

4D が提供するものとは異なる PHP インタプリタを使用することができます。カスタム PHP インタプリタは 2 つの条件を満たさなければなりません :

- FastCGI でコンパイルされていなければなりません。
- 4D と同じマシン上になければなりません。

カスタム PHP インタプリタを使用するには、指定されたアドレスと TCP ポートで待ち受けを行うよう、また 4D の内部インタプリタを有効にしないよう設定します。これらのパラメタはデータベース設定 (39 ページの "PHP" 参照) あるいはセッションごとに SET DATABASE PARAMETER コマンド (SET DATABASE PARAMETER, Get database parameter 参照) で設定できます。もちろんこの場合、インタプリタの起動や動作はあなたが管理しなければなりません。

追加のモジュールを使用するなどのため、カスタマイズした php.ini 設定ファイルを使用することもできます。php.ini ファイルは特に PHP エクステンションの場所を定義するために使用できます。
これを行うには、カスタマイズした php.ini ファイルをカレントデータベースの Preferences フォルダ (ストラクチャファイルと同階層の

"Preferences" フォルダ) に配置します。インタプリタの起動時に 4D がそのファイルの存在を検知し、この設定が 4D のデフォルト php.ini ファイルの代わりに使用されます。php.ini ファイルの内容が有効であることを開発者は確認しなければなりません。

警告 : PHP コードの実行時に発生するエラーが 4D に通知されるためには、php.ini ファイルで "display_errors" が "stderr" に設定されていなければなりません。例えば :

```
; stderr - Display the errors to STDERR (only affects the CGI/CLI)
; To direct the errors to STDERR for the CGI/CLI:
display_errors = "stderr"
```

カスタム php.ini ファイルの設定に関する情報は、4D が提供する php.ini ファイル中のコメントを参照してください。

PHP モジュール

4D v12 で提供される PHP インタプリタには、統合されたモジュールの適切さと妥当性を保持するセレクションが含まれます。

4D が提供するモジュールの詳細なリストと例題は 307 ページの appendix A、"PHP モジュール " を参照してください。

必要であれば、モジュールを追加インストールして、アプリケーションに機能を追加できます。PHP モジュールの追加に関する情報は、313 ページの " 追加モジュールのインストール " を参照してください。

PHP スクリプトの実行

PHP スクリプトや関数を実行するには、新しい PHP Execute コマンドを使用します。このコマンドは 179 ページの "PHP Execute" で説明しています。

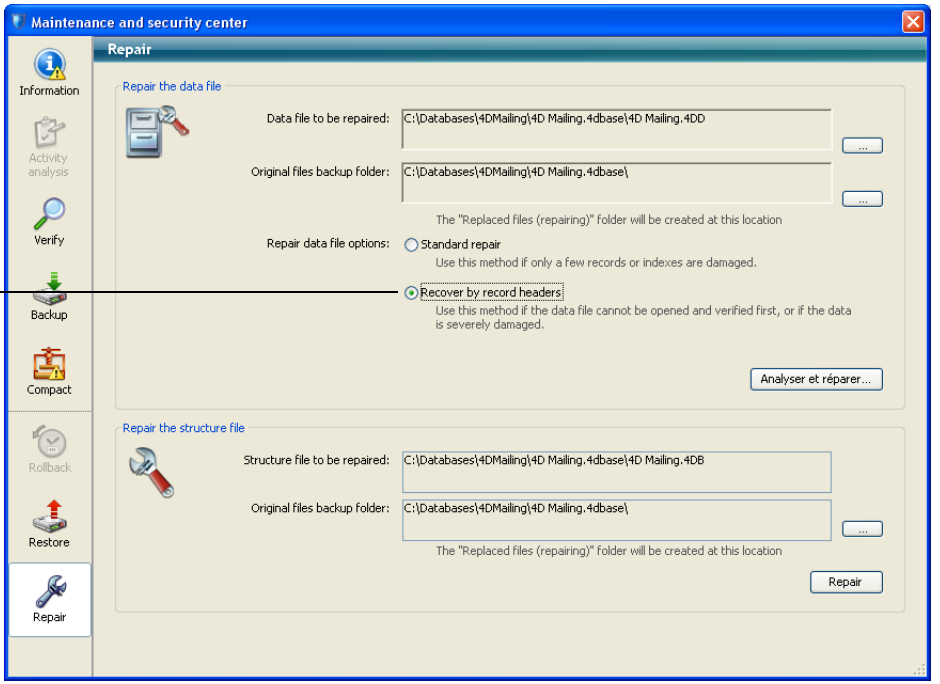
デフォルトで、4D v12 はバージョン 5.3.2 の PHP を含んでいます。実行されるスクリプトはこのバージョンおよびインストールされるモジュールと互換でなければなりません。4D が提供する PHP で利用できるモジュールのリストは 307 ページの appendix A、"PHP モジュール " を参照してください。

PHP コマンドとシンタックスの説明は、インターネット上で豊富に公開されている PHP のドキュメントを参照してください。以下は参照情報を見ることのできるサイトの例です :

http://us.php.net/manual/en/
http://phpdeveloper.org/
http://php.start4all.com/
http://php.resourceindex.com/Documentation/

レコードヘッダーによる再生

Maintenance & Security Center (MSC) の**修復**ページに新しい**レコードヘッダーによる再生**オプションが追加され、レコードヘッダーマーカーを使用してレコードを再生できるようになりました :



新しい再生オプション

この低レベルな修復オプションは、データファイルが激しく破損していて、他のすべての解決法 (バックアップからの復元、標準の修復) が全く効果がない場合にのみ使用します。

ヘッダーマーカーによる再生について

4D レコードはサイズが可変であるため、レコードを探すためには、ディスク中の格納場所を保持する必要があります。そしてプログラムはインデックスやアドレステーブルを使用して、レコードのアドレスにアクセスします。

レコードまたはインデックスのみが壊れている場合、問題の解決には通常、標準の修復オプションが適しています。しかしアドレステーブルが関連する場合、アドレステーブルを再構築するために、より上級の修復が必要になります。この処理を実行するために、MSC はレコードごとのヘッダにあるマーカーを使用します。マーカーをレコードのサマリー情報と比較し、アドレステーブルを再構築することが可能となります。

Note: データベースストラクチャのテーブルプロパティオプションで、**レコードを完全に削除**オプションを選択していない場合、ヘッダーマーカーによる再生を行うと、普通に削除したレコードが復活することがあります。

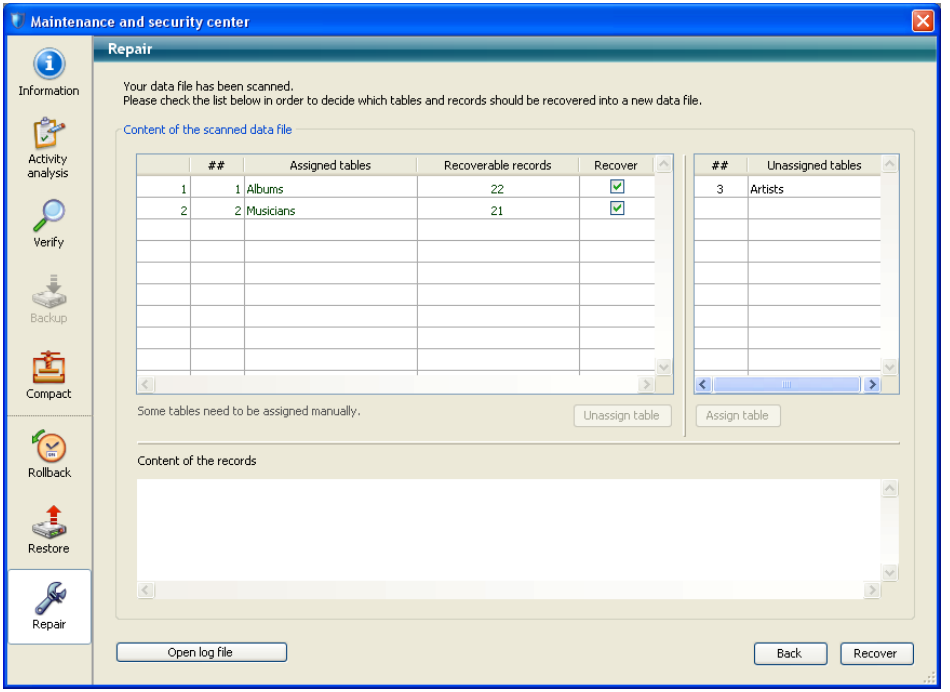
ヘッダによる修復は一切の整合性制約を考慮しません。特にこの処理を行った後は重複付加フィールドに緒副下値が現れたり、NULL 値を許可しない設定にされたフィールドに NULL 値が入っていることがあります。

再生手順

ヘッダーによる再生を行うには、MSC の " 修復 " ページを開き、**レコードヘッダーによる再生**オプションを選択して、**スキャンおよび修復 ...** をクリックします。

Note: この処理を行うにはデータベースをメンテナンスモードで開く必要があります。まだこのモードで開かれていない場合、警告ダイアログが表示され、データベースが閉じられメンテナンスモードで開かれます。

再生手順の最初のフェーズで、4D はデータファイルを完全にスキャンします。スキャンが終了すると、結果が以下のウィンドウに表示されます :



Note: すべてのレコードとすべてのテーブルがすでに再生対象になっている場合、メインエリアのみが表示されます。

" スキャンしたデータファイルの内容 " エリアには 、データファイルのスキャン結果の情報が 2 つの表に統合されています。

最初の表にはデータファイルのスキャン結果が一覧されます。それぞれの行にはデータファイル中で再生可能なレコードのグループが表示されます :

- 一番目の列はレコードグループの再生順を示します。
- ## 列はグループに割り当てられるテーブルの番号を示しています。
- **割当先テーブル**列は、識別されたレコードグループに自動で割り当て可能なテーブル名です。自動で割り当てられたテーブル名は緑で表示されます。
- **再生可能レコード** 列はグループ中の再生可能なレコード数を表します。
- **再生**列では、グループごとにレコードを再生するかどうか指定できます。デフォルトですべてのグループにチェックが付けられています。

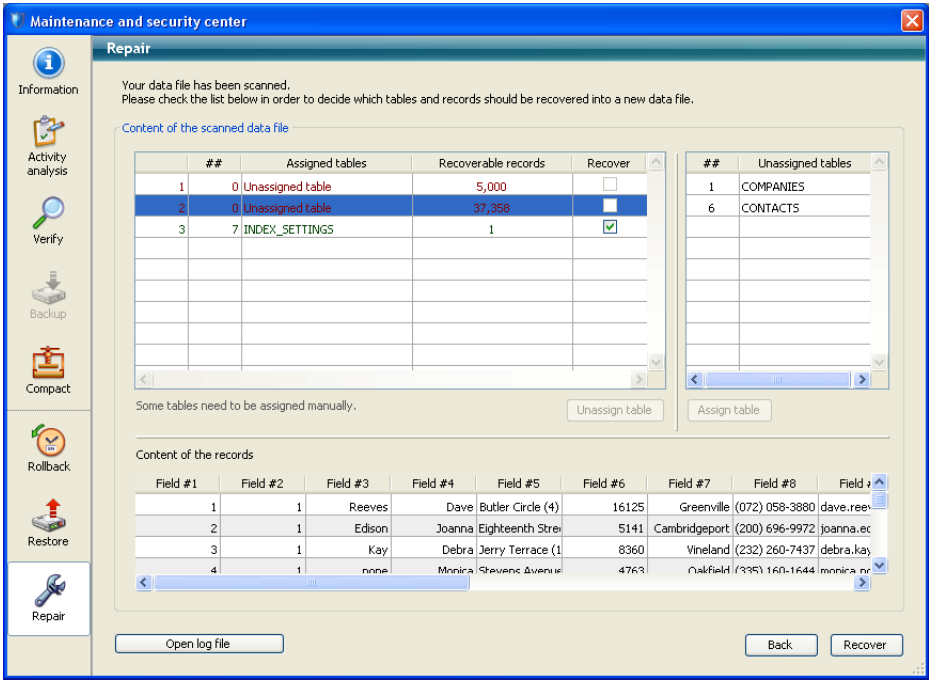
2 番目の表には割り当てられていないテーブルが一覧されます。言い換えると、対応するレコードがないテーブルです。

手動による割り当て

アドレステーブルの損傷のため、レコードグループをテーブルに割り当てることができない場合、手動でそれを行うことができます。

これを行うには、まず最初のテーブル中で未割当のレコードグループを選択します。すると " レコードの内容 " エリアにグループの最初のレコー

ドの内容がプレビューされます。これによりグループの割り当てが容易になります :



次にグループに割り当てるテーブルを " 割り当てられていないテーブル " から選択し、**テーブルを割り当て**ボタンをクリックします。ドラッグ&ドロップでもテーブルを割り当てられます。

これでレコードグループにテーブルが割り当てられ、このテーブルに再生されます。手動で割り当てられたテーブルは黒で表示されます。

テーブルを割り当てないボタンを使用して手動で設定したテーブルとレコードグループ間の割り当てを取り除くこともできます。

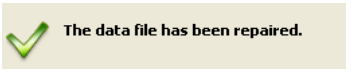
再生の起動

再生する要素を設定したら、**再生**ボタンをクリックして修復を起動できます。

Note: **戻る**ボタンをクリックして処理をキャンセルし、前の状態に戻ることもできます。

4D は新しい空のデータファイルを元のファイルの場所に作成します。元のファイルは MSC の " 修復 " ページで指定された場所 (デフォルトでデータベースフォルダ) の "Replaced Files (Repairing) date time" フォルダに移動されます。空のファイルにデータが再生されます。

処理が終了すると、MSC の " 修復 " ページが表示され、修復が完了した旨のメッセージが表示されます :



この場合、すぐにデータベースを開くことができます。

MSC のすべての検証機能のように、**ログファイルを開く** ボタンはブラウザで Web ページを開き、処理の結果を表示します。このページには実行されたすべての検証と修復、および発生したエラーが一覧されます (検証が正しければ [OK] が表示されます)。ログファイルはデータベースの **Logs** フォルダに作成されます。これは XML と HTML フォーマットで作成され、"DatabaseName_Repair_log" という名称です。

4 フォームとオブジェクト

4D v12 はフォームとフォームオブジェクトに関して多くの新機能を提供します。これらの新機能によりアプリケーションインタフェースの開発が容易になり、また高速化されます。

サブフォームの能力の拡張

4D v12 ではサブフォーム (組み込みフォーム) の能力が拡張されました。より自由に使用でき、プログラム可能、コンポーネントで利用可能で、この新世代のオブジェクトは先進で容易に移植が可能な機能を開発するためのパワフルなツールとなります

用語

サブフォームに実装された概念を明確に定義するために、使用される用語をいくつか説明します :

- **サブフォーム** : 他のフォームに含まれる用途のフォーム。
- **親フォーム** : ひとつあるいは複数のサブフォームを含むフォーム。
- **サブフォームコンテナ** : 親フォームに含まれるオブジェクトで、サブフォームのインスタンスを表示します。
- **サブフォームインスタンス** : 親フォーム内に表示されたサブフォームの実体。この概念は重要です。ひとつの親フォーム内に同じサブフォームの複数のインスタンスを表示できるからです。

新しいプロパティ

親フォームにインストールされたサブフォームオブジェクトに、プロパティリストを使用して追加のプロパティを設定できます :

- **変数名** : サブフォームオブジェクトに変数をバインドできるようになりました。デフォルトでこの変数には、オブジェクト自身と同様 "Subform" という名称がつけられます。変数にはタイプがあり (後述参照)、親フォーム内で標準変数の形で表すことができます。変数を変更すると

フォームイベントが生成され、親フォームとサブフォームを同期させることができます :

- On Data Change フォームイベントを使用してサブフォームコンテナに対し、変数の値がサブフォーム内で変更されたことを通知できます。
- 新しい On bound variable change フォームイベントを使用してサブフォーム (サブフォームのフォームメソッド) に対し、変数が親フォーム内で変更されたことを通知できます (74 ページの " サブフォームの内容の更新 " 参照)。

- **変数タイプ** : サブフォームオブジェクトにバインドされる変数のタイプを設定するために使用します。デフォルトで文字タイプが使用されます。変数タイプは、バインドされた変数を経由して親フォームとサブフォーム間で交換される値の動作を決定します。
タイプを " なし " に設定することもできます。この場合、4D は自動で変数のタイプを実行時に決定します (239 ページの " ダイナミック変数 " 参照)。

- **ソース** : 以前のバージョンの 4D では、このプロパティは " ソーステーブル " という名称でした。このバージョンからは異なるタイプのソースが選択できるようになりました :
  - **< なし >**: サブフォームとしてプロジェクトフォームやコンポーネントフォームを使用したい場合にこのソースタイプを使用します。これらのサブフォームはページフォーム内でのみ利用できます。このフォームが動作するためには " リストサブフォーム " オプションの選択が外されていなければなりません。
    " 詳細フォーム " リストにコンポーネントフォームが表示されるようにするには、コンポーネント内で公開されていなければなりません (78 ページの " サブフォームの公開 (コンポーネント)" 参照)。
  - **テーブル名** : テーブルフォームを使用したい場合に、このタイプのソースを使用します (以前の 4D と同じ動作)。

Note: " プラットフォーム " アピアランスプロパティはもう表示されません。アピアランスはサブフォーム自身により設定され、変更できません。

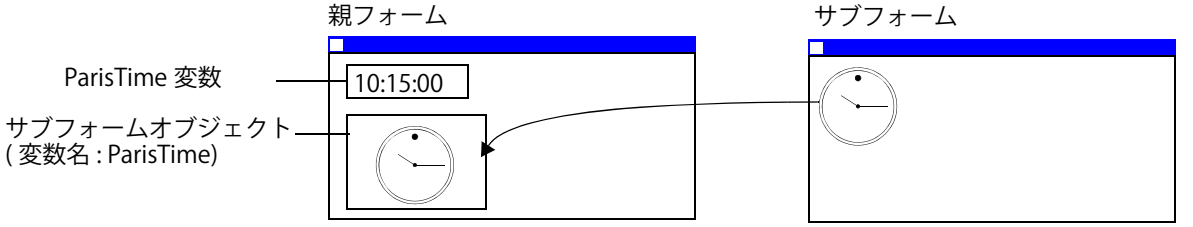
- **メソッド** : 特にイベントがトリガされたとき (後述参照)、サブフォームの動作をコントロールするために、サブフォームオブジェクトがオブジェクトメソッドを持つことができるようになりました。サブフォームオブジェクトのコンテンツと親フォーム間との相互作用は特定のメカニズムを使用して管理されなければなりません (76 ページの " 上級フォーム間通信プログラミング " 参照)。

- **フォームイベント** : サブフォームオブジェクトは以下のイベントを受け付けます :
  - On Load と On Unload: それぞれサブフォームが開かれるときと閉じられるときに生成されます (これらのイベントを使用するためには、親フォームレベルでもイベントが有効にされていなければなりません)。これらのイベントは親フォームのそれぞれのイベントの前に実行される点に留意してください。
    またフォームイベントの処理の原則に基づき、サブフォームが 0 や 1 ページ以外におかれていると、これらのイベントはそのページが表示され / 閉じられるときにのみ生成されます (フォームが開かれ / 閉じられるときではありません)。
  - On Validate: サブフォーム中でのデータ入力検証。
  - On Data Change: サブフォームオブジェクトの変数値が変更された。
  - On Getting Focus と On Losing Focus: サブフォームコンテナがフォーカスを得た、あるいは失ったときに生成されます。これらのイベントはサブフォームオブジェクトのメソッド中で、イベントがチェックされているときに生成されます。イベントはサブフォームのフォームメソッドに送信されます。つまり、例えば、フォーカスに基づき、サブフォーム内にナビゲーションボタンを表示するといったことが可能です。
    サブフォームオブジェクト自身がフォーカスを持つことに留意してください。

Note: - 新しい CALL SUBFORM CONTAINER コマンドを使用して、サブフォーム内で生成できるカスタムイベントタイプを指定できます。このコマンドを使用してコンテナオブジェクトメソッドを呼び出したり、イベントコードをコンテナオブジェクトメソッドに渡したりできます。
- 新しいフォームイベント、On bound variable change はサブフォームレベルで管理できます (74 ページの " サブフォームの内容の更新 " 参照)。
- サブフォームのコンテキスト (サブフォームのフォームメソッド) で SET TIMER コマンドが実行されるとき、On Timer イベントは、親フォームレベル (以前のバージョンの 4D での動作) ではなく、サブフォーム内で生成されます。これはとくに、それぞれのサブフォームおよび親フォームで異なる間隔の On Timer イベントを生成することを可能にします。

バインドした変数の管理

サブフォームにバインドした変数を使用して、2 つのコンテキスト (フォームとサブフォーム) をリンクでき、洗練されたインタフェースを作成できるようになります。例えばサブフォーム上に時計があるとしましょう。このフォームは、入力可能な時間タイプの変数がおかれた親フォームに表示されます :



両オブジェクト (時間変数とサブフォームコンテナ) は同じ変数名を持っています。この場合、親フォームが開かれるとき、両方の値は 4D により自動で同期されます。変数の値が複数の箇所で設定される場合、使用される値は最後にロードされたものです。以下のロード順が適用されます :

1- サブフォームのオブジェクトメソッド
2- サブフォームのフォームメソッド
3- 親フォームのオブジェクトメソッド
4- 親フォームのフォームメソッド

親フォームが実行されるとき、変数の同期は適切なフォームイベントを使用して開発者が行わなければなりません。2 タイプの相互作用、フォームからサブフォームとその逆があり得ます。

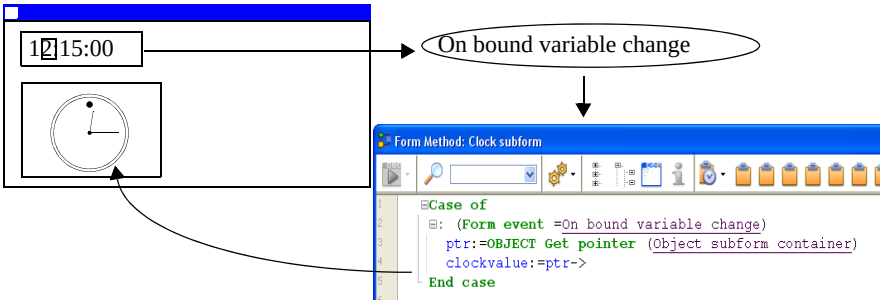
サブフォームの内容の更新

親フォームの変数値が更新され、この変更をサブフォームに反映させる必要があります。先の例でいえば、ParisTime の時刻がユーザの入力や Current time などのコマンドにより動的に 12:15:00 に変更されたとします。

この場合、新しい **On bound variable change** フォームイベントを使用しなければなりません。このイベントはサブフォームのプロパティで

チェックされていなければなりません。イベントはサブフォームのフォームメソッドで生成されます。

親フォーム



On bound variable change フォームイベントは以下の時に生成されます :

- 親フォームの変数に値が割り当てられるとすぐに。
- サブフォームがカレントフォームページまたは 0 ページに属しているとき。

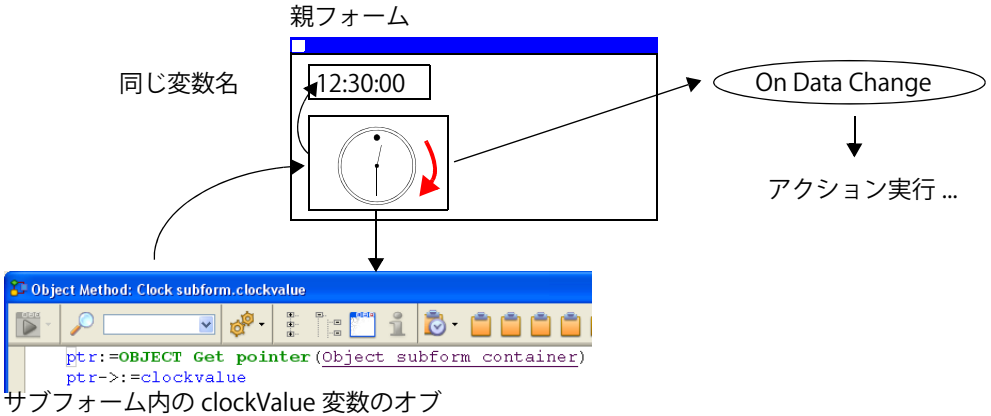
上の例の通り、サブフォームの変数を使用するのではなく、サブフォームコンテナへのポインタを返す OBJECT Get pointer コマンドを使用することを強くお勧めします。なぜなら同じ親フォームに複数のサブフォームを配置できるからです (例えば異なるタイムゾーンを表示するウィンドウには複数の時計が表示されます)。この場合、ポインタを使用してのみ、どのサブフォームコンテナがイベントの生成元かを知ることができます。

親フォームの内容の変更

サブフォームの内容が変更され、この変更を親フォームに反映させる必要があるとします。例えば先の例で、時計の針がユーザにより移動されたとします。

この場合、サブフォームから、オブジェクトの値を親サブフォームコンテナの変数に代入しなければなりません。先の例でいえば、OBJECT Get pointer コマンドを Object subform container 定数で使用して、サブフォームコンテナへのポインタを取得することを推奨します。
変数に値を代入すると、親サブフォームコンテナのオブジェクトメソッドで **On Data Change** フォームイベントが生成され、開発者はどのような

タイプのアクションも実行できます。このイベントはサブフォームコンテナのプロパティで有効にされなければなりません。



サブフォーム内の clockValue 変数のオブジェクトメソッド

Note: " 手入力 " で時計の針を動かした場合も、サブフォーム内の clockValue のオブジェクトメソッドで **On Data Change** フォームイベントが生成されます。

上級フォーム間通信プログラミング

親フォームとサブフォームのインスタンスとの間の通信には、バインドされた変数を経由した値の交換以上のことが必要となることがあります。実際、親フォームで実行されたアクションに基づきサブフォームの変数を更新したい場合や、その逆を行いたい場合があるでしょう。先の時計の例でいえば、時計ごとにアラームの時間を設定したいといった場合です。

このニーズを満たすために、新しいメカニズムが 4D に実装されました。これは以下のように動作します :

- OBJECT Get pointer コマンドにサブフォーム引数を使用してサブフォームオブジェクトを特定し、新しい OBJECT Get name コマンドを使用する。
- 新しい CALL SUBFORM CONTAINER コマンドを使用してサブフォームからコンテナオブジェクトを呼び出す。
- 新しい EXECUTE METHOD IN SUBFORM メソッドを使用してサブフォームのコンテキストでメソッドを実行する。

Object get pointer と Object get name コマンド

"Object subform container" セレクタ (74 ページの " サブフォームの内容の更新 " 参照) に加え、Object get pointer コマンドは、二番目の引数に渡されたオブジェクトを検索するサブフォームを指定する引数を受け入れます。このシンタックスは "Object named" セレクタが渡されたときにのみ使用できます。

例えば以下のステートメントは :
$ptr:=**Object get pointer**(Object named;"MyButton";"MySubForm")

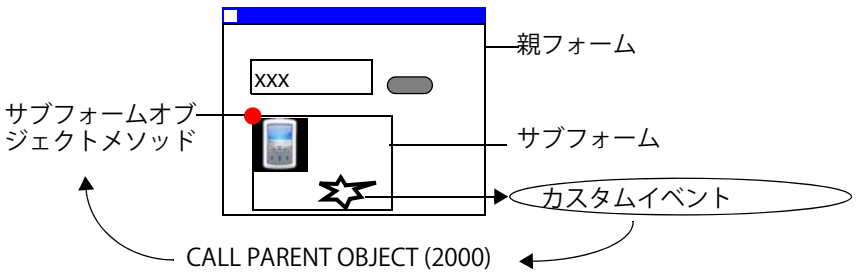
"MyButton" サブフォームオブジェクト内に置かれた "MyButton" 変数へのポインタを取得するために使用できます。このシンタックスは親フォームからサブフォーム内のオブジェクトにアクセスするために使用できます。このコマンドに関する説明はコマンド OBJECT Get pointer、ページ 155 を参照してください。

Note: フォーカスのあるオブジェクトの名前を取得するために使用する新しい OBJECT SET STYLED TEXT ATTRIBUTES コマンドにも留意してください。

新しい CALL SUBFORM CONTAINER コマンド

新しい CALL SUBFORM CONTAINER コマンドを使用すると、サブフォームインスタンスからサブフォームコンテナオブジェクトにイベントが送信され、親フォームのコンテキストでそれを処理することができます。イベントはコンテナオブジェクトメソッドで受信されます。それはサブフォームで検知されたイベント (クリックやドラッグ & ドロップなど) の源泉かもしれません。このメカニズムを図示すると以下のようになります :

CALL PARENT OBJECT コマンドの実行例



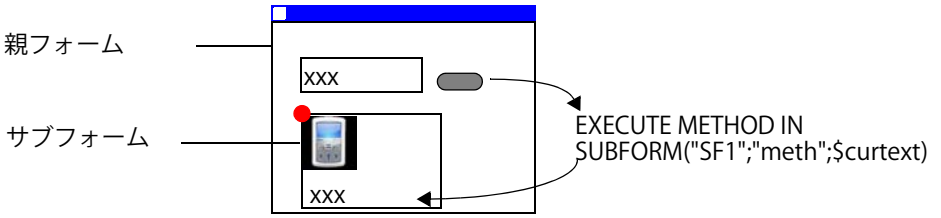
イベントコードに制限はありません (例えば 20000 や -100 など)。論理的には、既存のフォームイベントと重複しないようにしなければなりません。将来のバージョンでも 4D がこのコードを使用しないことを保証するために、負数の利用をお勧めします。
既存のイベントに対応するコードを使用することもできます (例えば On Validate (3))、しかしこの場合、サブフォームコンテナでイベントが有効にされていなければなりません。

このコマンドに関する説明は、136 ページの "CALL SUBFORM CONTAINER" を参照してください。

新しい EXECUTE METHOD IN SUBFORM コマンド

この新しいコマンドを使用して、サブフォームのコンテキストでのメソッド実行をフォームやオブジェクトからリクエストでき、これによりサブフォームの変数、オブジェクト、その他にアクセスできます。このメソッドは引数を受け取ることもできます。このメカニズムを図示すると以下のようになります :

EXECUTE METHOD IN SUBFORM コマンド実行の例



Note: サブフォームがコンポーネントに属している場合、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する " プロパティが選択されていなければなりません。

このコマンドに関する説明は 162 ページの "EXECUTE METHOD IN SUBFORM" を参照してください。

GOTO OBJECT コマンド (以前の GOTO AREA)

GOTO AREA コマンドは GOTO OBJECT に名称変更され、サブフォームの中から実行された場合でも親フォームの移動先オブジェクトを探すことができるようになりました (138 ページの "GOTO OBJECT" 参照)。

コンポーネント中のサブフォーム

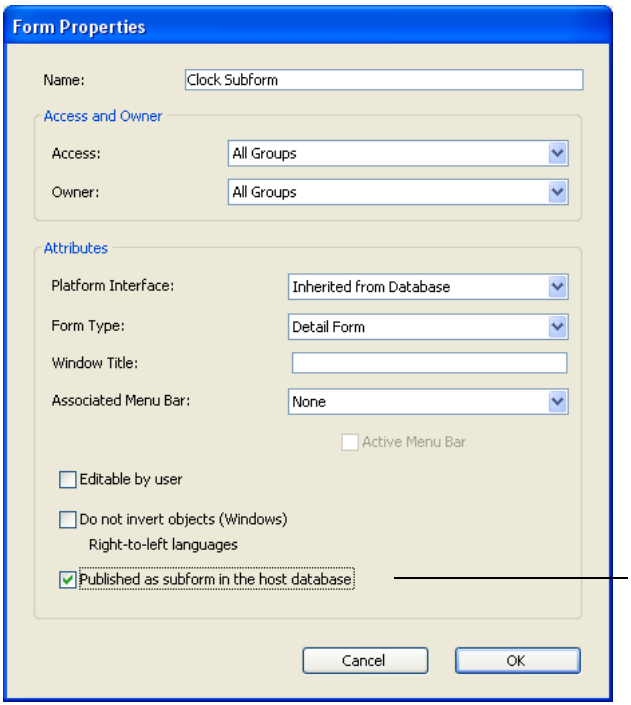
4D v12 ではホストデータベース中でコンポーネントフォームをサブフォームとして公開できるます。これは特に、グラフィックオブジェクトを提供するコンポーネントの開発ができることを意味します。4D 社が提供する新しいウィジェットオブジェクトはコンポーネントのサブフォームを使用しています (82 ページの " ウィジェット " 参照)。

サブフォームの公開 (コンポーネント)

コンポーネント (マトリクスデータベース) の側では、プロジェクトサブフォームのみを公開サブフォームとして定義できます。

ホストデータベース中でサブフォームとして選択されるコンポーネントフォームは、新しい**ホストデータベースでサブフォームとして公開**オプ

ションにより、フォームのプロパティダイアログボックスで " 公開フォーム " と明示的に指定しなければなりません。:



Note: このダイアログボックスはコンテキストメニューやエクスプローラのアクションメニューから**フォームプロパティ ...** コマンドを選択することでアクセスできます。

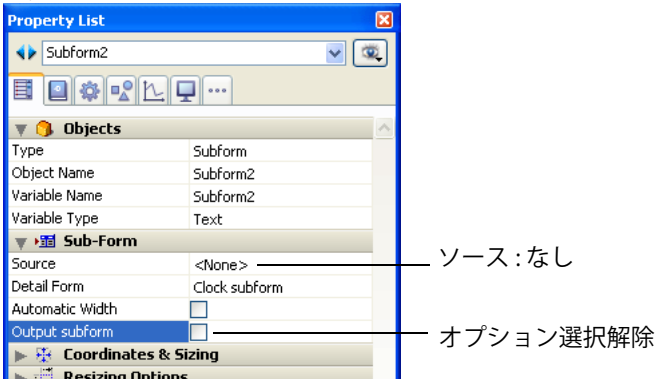
先に説明したメカニズムとツールを使用して、ホストデータベースの親フォームとサブフォーム間との相互作用を管理しなければなりません。

コンポーネントサブフォームを使用する (ホストデータベース)

ホストデータベース側では、コンポーネントに所属するサブフォームをページモードで利用することが必須です。これを行うにはフォームエディタで親フォーム中のサブフォームオブジェクトを選択し、プロパティリストの " サブフォーム " テーマ中、**出力サブフォーム**オプションの選択を解除します。

そして " テーブル " メニューから **< なし >** を選択します。これでコンポーネントから公開されたフォームは " 詳細フォーム " にリストされます。フォーム名の後ろには括弧の中にコンポーネント名が表示されます。

" 詳細フォーム " のリストに使用されるフォームのリストから選択できます。



選択されたオブジェクトの新しいインスタンスが即座にフォーム上に作成されます。

設定済みオブジェクトライブラリ

4D v12 の設定済みオブジェクトライブラリは 4D フォームにオブジェクトを追加することを簡単にするためにデザインされた新しいツールです。既に設定されたオブジェクトのコレクションが提供され、ドラッグ & ドロップやコピー / ペーストするだけで、フォーム中で使用できます。

ライブラリオブジェクト

このライブラリの利用目的は、特定のプロパティを事前に定義することで、標準の 4D オブジェクト (ボタン、テキストエリアなど) の利用を手助けすることにあります。例えば " パスワード入力エリア " オブジェクトは特定のスタイルシートが割り当てられたテキスト変数です。またライブラリでは日付ピッカーや時刻ピッカーウィジェットなどのハイレベルなオブジェクトも提供されます (82 ページの " ウィジェット " 参照)。

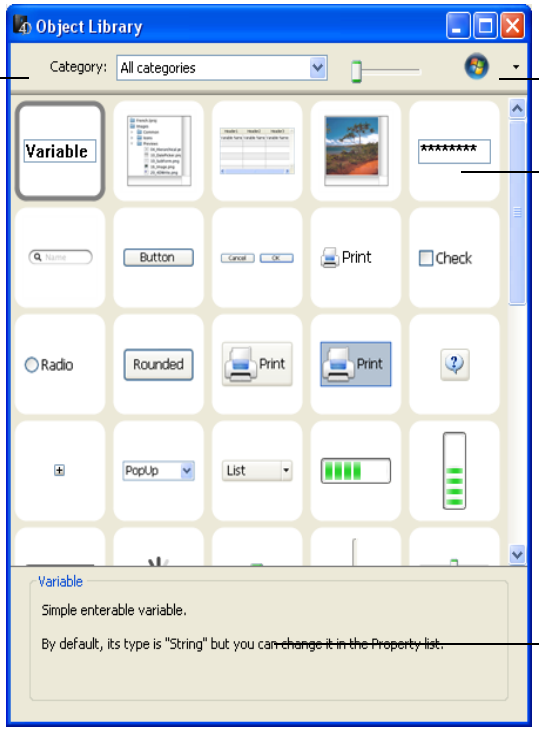
Note: ユーザオブジェクトライブラリと異なり、4D v12 の設定済みオブジェクトライブラリは変更できません。このライブラリにオブジェクトを足したり、削除したりすることはできません。

ライブラリの利用

設定済みオブジェクトライブラリは別ウィンドウに表示されます。オブジェクトはフォーム上にドラッグ＆ドロップで挿入できます。

設定済みオブジェクトライブラリウィンドウを表示するには、4D フォームエディタツールバーの最初のボタンをクリックします :

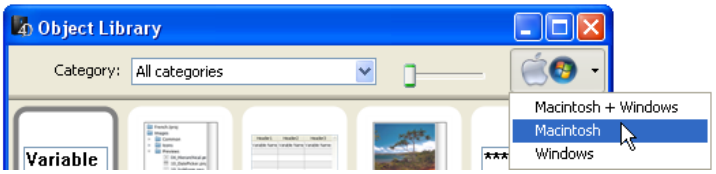
設定済みオブジェクトライブラリが新しいウィンドウに表示されます。この中にはポップアップメニュー、プレビューエリア、コメントエリアがあります :



オブジェクトはいくつかのカテゴリに分けられています (ボタン、入力エリアなど)。カテゴリに属するオブジェクトのみを表示するにはポップアップメニューから選択します。**すべてのカテゴリ**を選択するとすべてのオブジェクトが表示されます。

いくつかのオブジェクトは使えるプラットフォームが限定されます (Windows または Macintosh)。ウィンドウの右上に配置されたボタンを

使用してプラットフォームに基づき表示されるオブジェクトをフィルタできます。



中央のエリアにはプレビューおよびオブジェクトの名前が表示されます。オブジェクトをクリックしてオブジェクトに関する情報を得ることができます。説明はウィンドウの下部に表示されます。

ウィンドウ中央部からのドラッグ & ドロップまたはコピー / ペーストを使用してフォームにオブジェクトを挿入できます。オブジェクトは定義済みプロパティとともに配置されます。必要に応じてプロパティを変更することが可能です。

ウィジェット

4D v12 では定義済みの複合オブジェクトを使用することができます。プログラムあり、またはなしで使用できるこれらのウィジェットを使用すればとても簡単に標準の機能を実装できます。以下の新しいオブジェクトが利用できます :

- **SearchPicker**: 標準のアピアランスを持った検索エリア
- **DatePicker**: 日付セレクタ
- **TimePicker**: 時間セレクタ

Note: ウィジェットは 4D に新しく統合されたオブジェクトライブラリからもアクセスできます (80 ページの " 設定済みオブジェクトライブラリ " 参照)。

SearchPicker

SearchPicker ウィジェットを使用して、ブラウザやツールバーに見られるような標準の検索エリアを簡単に作成できます。エリアのアピアランスはプラットフォームに依存します。

Windows

Mac OS

デフォルトでエリアに表示されるテキストは、251 ページの "SearchPicker" で説明するコンポーネントメソッドを使用して、プログラムでコントロールできます。

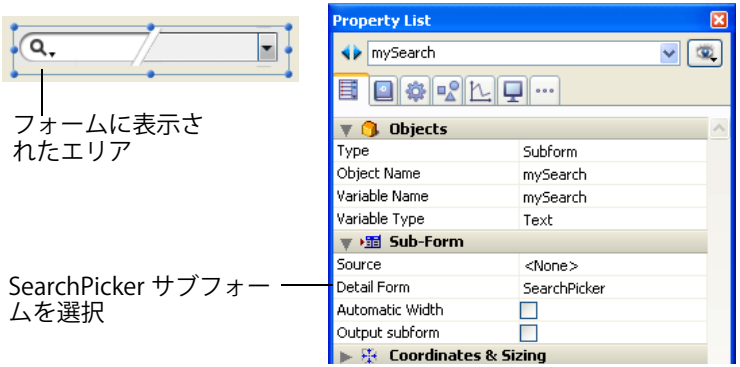
動作

アピアランスの他、SearchPicker 検索エリアは以下の要素による特徴を持っています : グレイアウトされたテキスト、入力エリア、削除アイコン。

- 入力エリアには検索する値を入力します。この値はプロパティリストの**変数名**プロパティを使用してエリアにバインドした変数に自動および動的に代入されます。この変数を使用して検索する値を取り出し、検索メソッドに渡すことができます。
- グレイアウトされたテキストは、検索が実行されるフィールドをユーザに補助的に提示するものです。エリアにフォーカスが当たるとテキストは消えます。このテキストは SearchPicker SET HELP TEXT コマンドで設定できます。
- 削除ボタンはエリアの内容を消去するために使用します。このボタンは自動で動作します。
- 実行中、ボタンのクリックやフォームイベントを使用して検索メソッドを起動できます。エリアは特に On Data Change や On Losing Focus イベントを生成します。On Data Chang イベントで検索メソッドを呼び出すことで、ユーザが他の文字を入力するたびに再評価を行うような動的検索を提供できます。ユーザが **Enter** キーを押したときに検索を起動することもできます。この場合、On Losing Focus イベントで検索メソッドを呼び出します。

作成

SearchPicker エリアは新しく 4D に統合されたオブジェクトライブラリから挿入できます (80 ページの " 設定済みオブジェクトライブラリ " 参照)。またサブフォームを作成し、SearchPicker コンポーネントの詳細フォーム割り当てることもできます。この点については 79 ページの " コンポーネントサブフォームを使用する (ホストデータベース)" を参照してください。.



そしてサブフォームにバインドする変数名を指定します (プロパティリストの**変数名**プロパティ)。フォームが実行されると、この変数には自動でユーザの検索値が代入されます。この値を検索メソッドに渡すことができます。

DatePicker

DatePicker ウィジェットは直感的に簡単に使用できるオブジェクトで、日付の入力を要求するエリアを作成するために使用できます。

ウィジェットは 2 つの形式で提供されます :

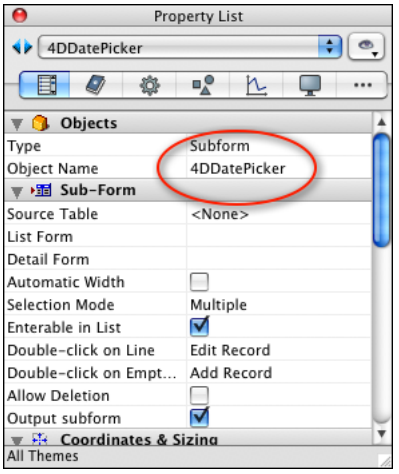
- **DatePicker カレンダ** : このオブジェクトはサブフォームおよびボタンクリックで表示されるプルダウンカレンダとして利用できます。
- **DateEntry エリア** : コントロールボタンが割り当てられた日付エリア。このオブジェクトはサブフォームでのみ利用できます。

DatePicker カレンダ

実行中、ユーザは矢印ボタンをクリックして次月・前月と移動できます。キーボードの矢印キーも使用できます。

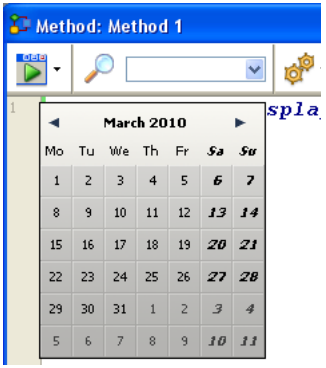
DatePicker をサブフォームに挿入すると、バインド変数により提供されるメカニズム (後述) のおかげで、DatePicker オブジェクトをプログラムなしで使用できます。しかし DatePicker の動作をカスタマイズしたい場合やポップアップメニューとして表示したい場合などは、提供されるコンポーネントメソッドを使用しなければなりません。これらのコンポーネントメソッドは 243 ページの "DatePicker" で説明しています。

- **サブフォームで利用**
  4D に新しく統合されたオブジェクトライブラリ (80 ページの " 設定済みオブジェクトライブラリ " 参照) を使用して、またはサブフォームエリアを作成して DatePicker コンポーネントのフォームを割り当てることにより DatePicker カレンダをフォームに挿入できます (79 ページの " コンポーネントサブフォームを使用する (ホストデータベース)" 参照)。



  そしてサブフォームにバインドする変数の名前を指定します (プロパティリストの**変数名**プロパティ)。
  フォームが実行されると、この日付変数には自動でユーザが選択した日付が代入されます。逆にこの変数の値をプログラムで変更すると、サブフォーム内でその日付が表示されます。

- **ポップアップで利用**
  DatePicker カレンダをポップアップウィンドウで利用できます。これを行うには DatePicker Display Dialog コンポーネントメソッドを呼び出します。このメソッドはユーザが選択した日付を返します。



4DDatePicker コンポーネントのメソッドは 243 ページの "DatePicker" で説明しています。

DateEntry エリア

DateEntry タイプのエリアは標準的な DD/MM/YY 形式による日付入力を容易にします。

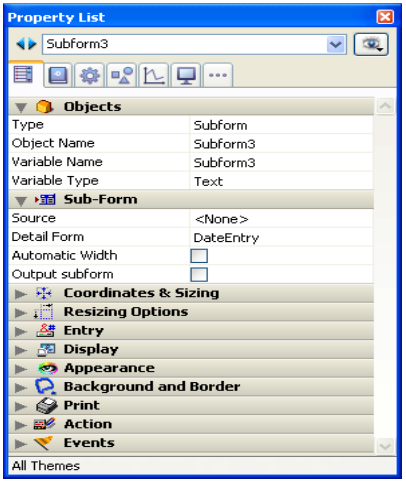
エリアはボタンが割り当てられた日付タイプとして表示されます :

実行中入力エリアの右側にあるボタンはオブジェクトにフォーカスがあるときにのみ表示されます。クリックやタブキーを使用して、ユーザは日付の要素 (年 / 月 / 日) を個別に選択できます。また数値ステッパーボタンやキーボードの上下キーを使用して数値をスクロールできます。右側のカレンダアイコンを使用すると、DatePicker ポップアップカレンダを使用して日付を選択できます。

バインド変数により提供されるメカニズム (後述) のおかげで、DateEntry オブジェクトをプログラムなしで使用できます。しかし DateEntry の動作をカスタマイズしたい場合は、提供されるコンポーネントメソッドを使用しなければなりません。これらのコンポーネントメソッドは DatePicker オブジェクトと同じです。詳細は 243 ページの "DatePicker" を参照してください。

- **サブフォームで利用**
  4D に新しく統合されたオブジェクトライブラリ (80 ページの " 設定済みオブジェクトライブラリ " 参照) を使用して、DateEntry エリアをフォームに挿入できます。またサブフォームエリアを作成して DateEntry コンポーネントのフォームを割り当てることにより DateEntry カレンダを

フォームに割り当てることもできます (79 ページの " コンポーネントサブフォームを使用する (ホストデータベース)" 参照)。

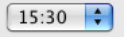


そしてサブフォームにバインドする変数の名前を指定します (プロパティリストの**変数名**プロパティ)。
フォームが実行されると、この日付変数には自動でユーザが選択した日付が代入されます。逆にこの変数の値をプログラムで変更すると、サブフォーム内でその日付が表示されます。

TimePicker

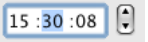
TimePicker ウィジェットは時間の入力や表示をさらにアトラクティブに行えるフィールドを作成するための簡単に使用できるオブジェクトを提供します。

1 つ、あるいは 2 つのポップアップメニュー、または時 / 分 / 秒を増減させるボタンを伴った "hh:mm:ss" 形式の時間エリアとして使用できます。



1 つのポップアップ

2 つのポップアップ



時間エリア (サブフォーム)

さらにそれぞれのタイプの TimePicker は 12 時間 (AM-PM) または 24 時間形式で時間を表示できます。

TimePicker オブジェクトは、バインド変数により提供されるメカニズム (後述) のおかげで、プログラムなしで使用できます。しかし TimePicker オブジェクトの動作をカスタマイズしたい場合は、提供されるコンポーネントメソッドを使用しなければなりません。これらのコンポーネントメソッドについては 251 ページの "TimePicker" を参照してください。

- **利用**
  TimePicker エリアは、4D に新しく統合されたオブジェクトライブラリ (80 ページの " 設定済みオブジェクトライブラリ " 参照) を使用してフォームに挿入できるサブフォームです。またサブフォームエリアを作成して TimePicker コンポーネントのフォームを割り当てることもできます (79 ページの " コンポーネントサブフォームを使用する (ホストデータベース)" 参照)。そしてサブフォームにバインドする変数の名前を指定します (プロパティリストの**変数名**プロパティ)。
  フォームが実行されると、この変数には自動でユーザが選択した時刻が代入されます。逆にこの変数の値をプログラムで変更すると、サブフォーム内でその時刻が表示されます。

  4DTimePicker コンポーネントメソッドは 251 ページの "TimePicker" で説明しています。

新しいフォームオブジェクト

この節では 4D フォームで使用できる標準オブジェクトについて説明します。

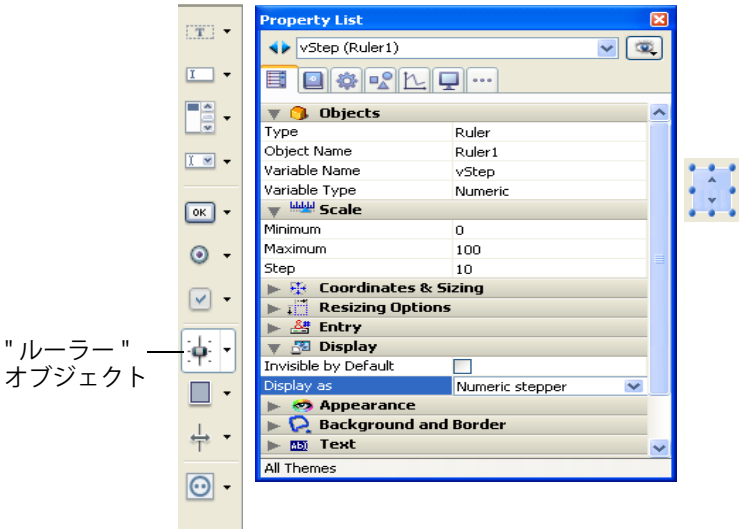
新しいステッパーオブジェクト

4D v12 に新しいフォームオブジェクト、" ステッパー " が追加されました。この標準オブジェクトを使用すれば、三角ボタンをクリックさせることでユーザに数値、時間、日付を、事前に定義した通りにスクロールさせることができます。

この機能の動作は既にあるルーラオブジェクトに似ています。オブジェクトに割り当てられた変数をフィールドや変数などの入力エリアに代入て、オブジェクトの現在地を格納したり変更したりできます。

ステッパーオブジェクトはルーラオブジェクトの派生形です。

ステッパーをフォームに配置するには、ルーラオブジェクトを描画し、プロパティリストの表示メニューから**数値ステッパー**を選択します :



ステッパーの最大値、最小値、ステッププロパティを設定できます。他のプロパティ、特にフォームイベントはすべてのインジケータオブジェクトと同様に動作します。

ステッパーには直接数値、日付、時間変数を割り当てることができます。

- 時間タイプの場合、最大値、最小値、ステッププロパティは秒を表します。例えばステッパーを 8:00 から 18:00 までの 10 分ステップで設定するには以下のようにします :
  - 最小 = 28 800 (8*60*60)
  - 最大 = 64 800 (18*60*60)
  - ステップ = 600 (10*60)
- 日付タイプの場合、ステッププロパティに入力された値は日付を表します。最大と最小プロパティは使用されません。

ステッパーを日付や時間変数で使用できるようにするには、プロパティリストでタイプを指定し、かつ明示的に C_TIME や C_DATE コマンドで宣言しなければなりません。

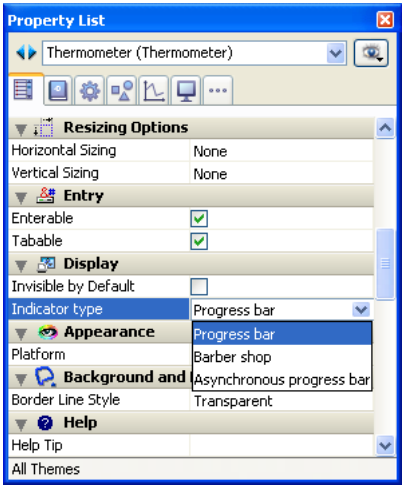
Note: OBJECT SET FORMAT (SET FORMAT コマンドの新しい名称) コマンドを使用して " ステッパー " 表示を有効にできます。

非同期進捗インジケータ

注意事項: インタフェース標準に適合させるために、4D v12 において、" サーモメータ " オブジェクトは " 進捗インジケータ " に名称が変更されました。

4D v12 では進捗インジケータに複数のバリエーションが加えられました。これらの異なるバリエーションを使用して、ネットワーク接続の検索や計算などの現在の処理状況を表示する、標準のインタフェースオブジェクトを表示できます。

これらのバリエーションを使用するには進捗インジケータをフォームに描画し、プロパティリスト中で新しい " インジケータータイプ " メニューを使用します。このメニューには利用可能なインジケータが含まれています:



- **進捗バー**: デフォルトのインジケータ。この変数は以前のバージョンの " サーモメータ " に対応します。" スケール " テーマのオプションを設定できます。
- **バーバーショップ**: インジケータにはバーバーショップタイプのアニメーションが継続的に表示されます。このバリエーションは以前のバージョンの " 未定義 " 設定に対応します。これが選択されると、" スケール " テーマのオプションは隠されます。
- **非同期進捗バー**: 継続的に回転するインジケータのアニメーションが表示されます。これが選択されると、" スケール " テーマのオプションは隠されます。

Note: 実行中インジケータがアニメーションするためには、オブジェクトに割り当てられた変数に 0 以外の値を設定します (例えば 1)。値が 0 に設定されるとアニメーションは停止します。

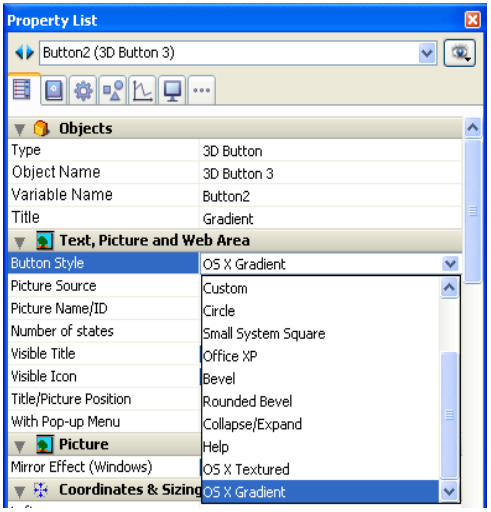
Note: これらのバリエーションは OBJECT SET FORMAT コマンド (SET FORMAT コマンドの新しい名称) を使用して指定できます。

新しい 3D ボタン

4D v12 には、3D ボタングループに 4 つの新しいスタイルが加えられ、ネイティブボタンをインタフェース中で使用できるようになりました :

- 折りたたみ / 展開
- ヘルプ
- OS X テクスチャ
- OS X グラデーション

これらのバリエーションには 3D ボタンや 3D チェックボックス、3D ラジオボタンのプロパティリスト中ボタンスタイルポップアップメニューからアクセスできます :



折りたたみ / 展開

このボタンスタイルは標準の折りたたみ / 展開アイコンを追加するために使用します。これらのボタンはネイティブに階層リスト中で使用されています。Windows 上でボタンは [+] や [-]、Mac OS 上では右または下を指す三角として表示されます。このスタイルは特に 3D チェックボックスでの利用を意図しています。2 つのボタン状態がチェックボックスの選択 / 非選択状態に対応します :

Windows　　Mac OS

ヘルプ

このボタンスタイルを使用してシステム標準のヘルプボタンを表示できます。このスタイルを使用してシステムヘルプボタンをフォームに追加できます :

Windows　　Mac OS

OS X テクスチャ

Mac OS X において、" テクスチャ " ボタンは標準のシステムボタンで、灰色のグラデーションで表示されます。高さは固定されていて、増減することはできません。このボタンスタイルではすべての 3D ボタンオプションを使用できます。
Windows で、このスタイルはプッシュボタンと同じですが、ポップアップメニューを持つことができ、また Vista 上では透過になることのできる特別な機能があります :

Windows　　Mac OS

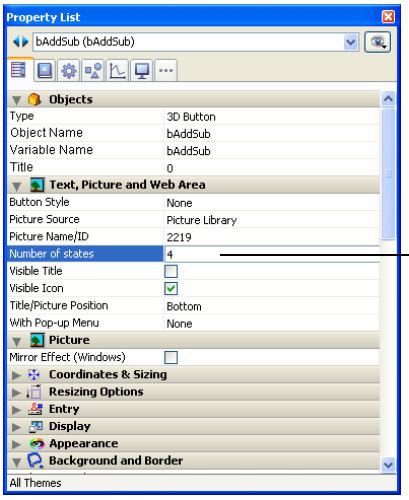
OS X グラデーション

Mac OS X において、" グラデーション " ボタンは 2 トーンのシステムボタンです。このボタンスタイルではすべての 3D ボタンオプションを使用できます。
Windows で、このスタイルはプッシュボタンと同じですが、ポップアップメニューを持つことができます :

Windows　　Mac OS

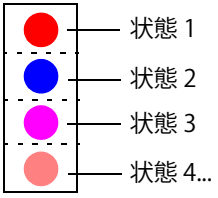
状態の数

3D ボタンで新しいプロパティ**状態の数**が使用できるようになりました。



このプロパティは、3D ボタンのアイコンとして使用するピクチャが表す状態の数を設定するために使用します。ソース画像では、状態を表す画像が縦に並んでいなければなりません:

ソース画像

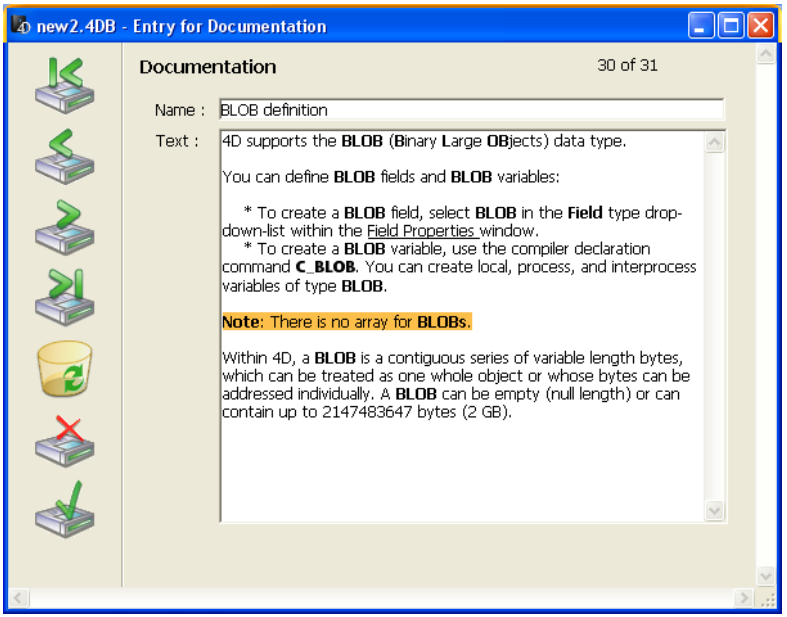


一般的にボタンアイコンは 4 つの状態、有効 / クリック中 / マウスオーバー / 無効を含みます。

リッチテキスト

概要

4D v12 では文字ごとににスタイルを設定できるリッチテキストエリアを使用できます。例えばテキストに太字、イタリック、文字色を持たせることができます :



この新しい機能は文字およびテキストタイプのフィールドや変数およびリストボックスのセルに適用されます。ページおよびリストフォームで、表示および印刷がサポートされます。

このリッチテキスト機能を設定するために新しいオプションがプロパティリストに追加されています。

利用可能な属性は**フォント**、**サイズ**、**スタイル**、**文字色**、そして (Windows のみ) **背景色**です。リッチテキストエリアでスタイルを更新するには 2 つの方法があります :

- 実行中に自動ポップアップメニューを使用する (このメニューを利用可能にするにはプロパティリストで設定します)。
- 新しい OBJECT SET STYLED TEXT ATTRIBUTES コマンドをプログラムで使用する。

リッチテキストエリアにて、スタイル属性は標準 HTML タグとして格納されます。テキストエリアが表示されると、これらのタグを 4D が解釈します。これは開発者がプログラムを使用してスタイル属性を指定したり

変更したりできることを意味します。4D がサポートするタグは 317 ページの appendix B、" スタイルタグ " で示されています。

新しい OBJECT Get plain text コマンドを使用してスタイルタグなしの生テキストを取得できます。

Note: 以下のコンテキストではリッチテキストエリアを使用できません : 入力フィルタ、クイックレポート、ラベルエディタ

リッチテキスト管理プロパティ

フォームエディ中で、リッチテキストのサポートを有効にして設定するための新しいプロパティを使用できます。これらのプロパティは文字またはテキスト型の入力可変数、フィールド、リストボックスセルで利用できます。

マルチスタイル

このオプション (" テキスト " テーマ) は、選択されたエリアで指定されたスタイルを利用可能にするかどうかを指定するものです。このオプションがチェックされていると、4D はエリア中の HTML スタイル管理タグを解釈します。

デフォルトでこのオプションはチェックされていません。

デフォルトスタイルタグを格納

このオプションは " マルチスタイルオプション " がチェックされているときにのみ " テキスト " テーマに表示されます。
このオプションがチェックされていると、エリアはたとえスタイルの変更が行われていなくても、テキストとともにスタイルタグを格納します。この場合、タグはデフォルトスタイルに対応します。このオプションがチェックされていないと、更新されたスタイルタグのみが格納されます。

以下のようにスタイルが変更されたテキストがあります :

What a beautiful day!

- " デフォルトスタイルタグを格納 " オプションがチェックされていない場合、エリアは更新されたスタイルのみを格納します。格納される内容は以下のようになります :

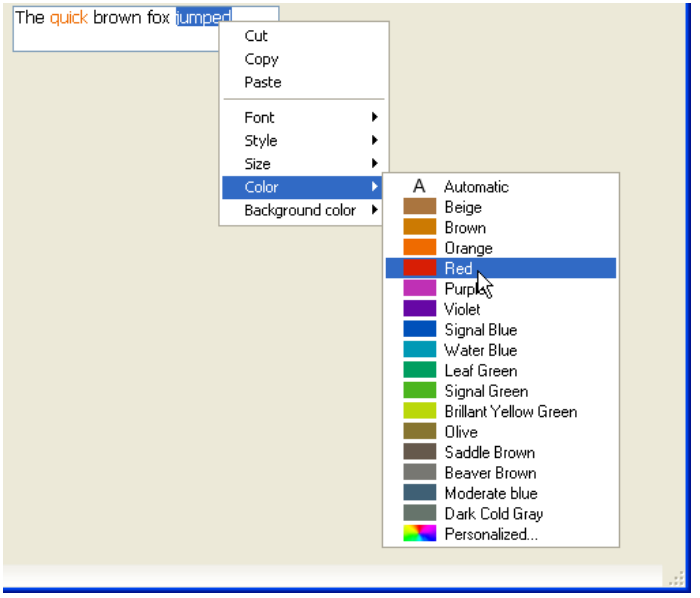
  What a<SPAN STYLE="font-size:13.5pt">beautiful</SPAN> day !

- オプションがチェックされていると、エリアはすべてのフォーマット情報を格納します。先頭の汎用タグはデフォルトスタイルを定義し、変更されたスタイルはネストされたタグに書き込まれます。格納される内容は以下のようになります :

<SPAN STYLE="font-family:'Arial';font-size:9pt;text-align:left;font-weight:normal;font-style:normal;text-decoration:none;color:#000000;background-color:#FFFFFF">What a <SPAN STYLE="font-size:13.5pt">beautiful</SPAN> day!</SPAN>

コンテキストメニュー

このオプション (" 入力 " テーマ) は " マルチスタイル " オプションがチェックされているときにのみ表示されます。
このオプションが選択されていると、ユーザがエリアに入力中、右クリックでポップアップメニューが表示されるようになります。このポップアップメニューでは標準のテキスト編集コマンド (カット、コピー、ペースト) の他、スタイルの変更を行うためのフォント、サイズ、スタイル、カラー、そして Windows のみ背景色コマンドが提供されます .



ユーザがこのポップアップメニューを使用してスタイル属性を編集すると、4D は On After Edit フォームイベントを生成します。

Note: 新しい OBJECT SET STYLED TEXT ATTRIBUTES コマンドを使用してスタイルを編集することもできます。この場合フォームイベントは生成されないことに留意してください。

Note: Mac OS では "strikethrough (取り消し線)" スタイルをサポートしていません。しかし対応するタグをプログラムで使用することはできます。

リッチテキストの処理

コピー / ペーストとドラッグ & ドロップ

サポートされるスタイル属性 (フォント、サイズスタイルそしてカラー) は以下のケースでドラッグ & ドロップやコピー / ペースト時にも保持されます :

- 4D 内での異なるリッチテキストエリア間 (テキスト変数 / フィールド、そしてリストボックス)
- 4D Write エリアと 4D リッチテキストエリア間
- 外部スタイル付きテキストエリアと 4D リッチテキストエリア間

その他の場合、スタイルはコンテキストに基づいて保持されます。

テキストオブジェクト管理コマンド

プログラムでテキストオブジェクトを操作するために使用することのできるコマンドは、テキストに統合されたスタイルタグを無視します。つまりこれらのコマンドは以前の 4D と同様に動作します。以下のコマンドが関連します :

- **ユーザインタフェース**テーマ
  HIGHLIGHT TEXT
  GET HIGHLIGHT

  文字列を操作するコマンドとともにこれらのコマンドを使用する際、新しいコマンド OBJECT Get plain text を使用してフォーマット文字をフィルタしなければならないことに留意してください :

  HIGHLIGHT TEXT([Products]Notes;1;**Length**(**OBJECT Get plain text**([Products]Notes))+1)
- **オブジェクトプロパティ**テーマ

Note: このテーマのコマンドは 4D v12 で名称変更され、また新しいコマンドが追加されました。詳細は 187 ページの " オブジェクトプロパティ " を参照してください。

オブジェクトのスタイルを変更するために使用するコマンド (例えば OBJECT SET FONT) は以前のバージョンの 4D と同様に動作します。これらのコマンドは選択された文字列ではなく、指定されたオブジェクト全体を対象とします。
コマンドが実行されたときオブジェクトにフォーカスがない場合、変更はオブジェクト (テキストエリア) とそれに割り当てられた変数に同時に適用されます。オブジェクトにフォーカスがある場合、変更はそのオブジェクトに適用され、割り当てられた変数には適用されません。この場合変更は変数に対してオブジェクトがフォーカスを失ったときにのみ適

用されます。テキストエリアに対するプログラムを行う際はこの原則を忘れないでください。

" デフォルトスタイルタグを格納 " がそのオブジェクトに対して選択されている場合、これらのコマンドを使用すると、オブジェクトに保存されているタグが更新されます。

Get edited text コマンド

Get edited text (" フォームイベント " テーマ) がリッチテキストエリアで使用されると、コマンドはすべてのスタイルタグを含む現在のエリアのすべてのテキストを返します。

編集された生テキスト (タグなしのテキスト) を取り出すには、新しい OBJECT Get plain text コマンドを使用しなければなりません :

OBJECT Get plain text(Get edited text)

Note: このコマンドに関する詳細は 191 ページの "OBJECT Get plain text" を参照してください。

クエリおよび並び替えコマンド

マルチスタイルオブジェクトに対して行われるクエリや並び替えはオブジェクトに保存されたスタイルタグを考慮に入れます。単語中でスタイルの変更が行われた場合、その単語の検索は失敗します。

有効な検索や並び替えを行うには、新しい OBJECT Get plain text コマンドを使用します。例えば :

QUERY BY FORMULA([MyTable];**Get plain text**([MyTable]MyFieldStyle)="very well")

新しいコマンド

リッチテキストエリアのスタイル属性に対して使用できる新しいコマンドがいくつか追加されました : OBJECT SET STYLED TEXT ATTRIBUTES, OBJECT GET STYLED TEXT ATTRIBUTES そして OBJECT Get plain text。これらのコマンドは 187 ページの " オブジェクトプロパティ " で説明しています。

Note: 新しい " 常に選択を表示 " プロパティを使用して、オブジェクトにフォーカスがない時でも、テキストのハイライトを表示させることができます。これはメニューやボタンを使用したスタイル変更インタフェースを作成できることを意味します。詳細は 102 ページの " 常に選択を表示 " を参照してください。

フィールドや変数の新しいプロパティ

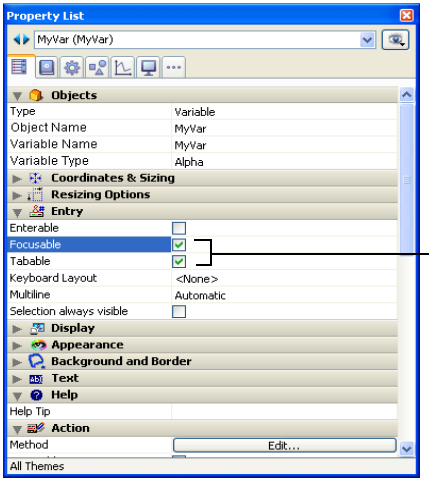
4D フォームエディタでフィールドや変数に対する新しいプロパティが使用できます。これらの新しい機能は 4D デベロッパがフォームオブジェクトをよりコントロールできるようにするために追加されました。

入力できないオブジェクト

以前のバージョンの 4D では入力不可に設定された (**入力可**オプションが選択されていない) フィールドや変数の内容を選択したりコピーしたりすることはできませんでした。今バージョン以降、**フォーカス可**、**タブ有効**そして**ドロップ可プロパティ**を使用して新しい相互作用が可能となりました。

フォーカス可とタブ有効

フォーカス可と**タブ有効**オプションは入力不可オブジェクトに対しても有効です :



入力不可変数やフィールドの新しいオプション

オブジェクトに**フォーカス可**オプションが選択されていると、ユーザはエリアの内容を選択、コピー、さらにはドラッグ & ドロップで取り出すことができます。

タブ有効オプションは**フォーカス可**オプションが選択されると有効になります。このオプションをチェックすると**タブ**キーを使用してオブジェクトを選択できるようになります。しかしオブジェクトの入力順に含めることは依然としてできないことに留意してください。

ドロップ可

ドロップ可プロパティを入力不可オブジェクトに設定できるようになりました。開発者は入力不可の変数やフィールドに対してドロップされたオブジェクトのレスポンスとしていかなるタイプのアクションもプログラムできるようになります。

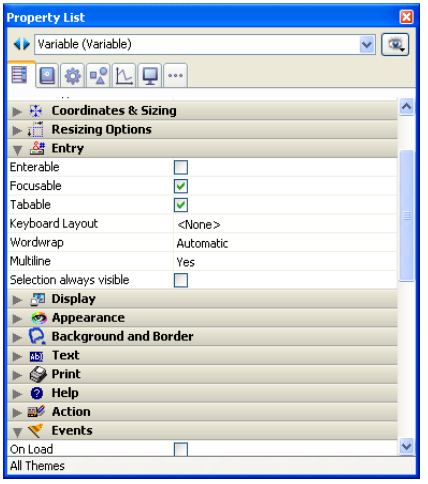
複数行オブジェクト

以前のバージョンの 4D では、フォーム中のフィールドや変数に格納されたテキストの表示や印刷は制御が困難でした。テキストタイプのオブジェクトにおいて、表示はエリアのサイズとプラットフォームに左右されました。

Note: 改行を文字タイプオブジェクトに使用することはできません。

4D v12 ではテキストエリアの表示と印刷がプラットフォームとの間で調和され、また 2 つの新しいオプションを使用して行送りをコントロールできます :

- **複数行** : 文字やテキストタイプの、入力可不可両方の変数およびフィールドで利用可能なオプション。3 つの選択肢から選択できます : **はい** , **いいえ** , **自動**
- **ワードラップ** : **複数行**オプションが**はい**に設定された場合にのみ利用可能なオプション。3 つの選択肢から選択できます : **はい** , **いいえ** , **自動**

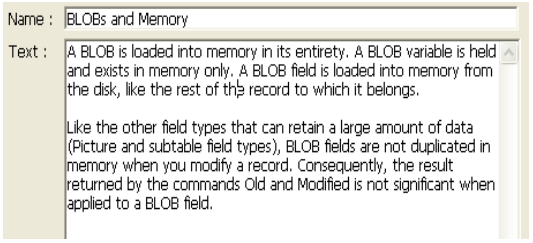


これらのオプションを使用してテキストエリアの 2 つの表示パラメタを設定できます :

- 一行のエリアにおける、行の最後の単語の表示
- テキストエリアにおける行送りの自動挿入

複数行 = 自動

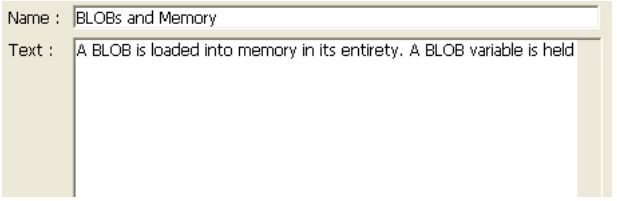
一行のエリアでは、行の終わりにある単語は切り取られ、行送りはありません。
複数行エリアでは、4D が自動で行送りを行います :



この動作は以前のバージョンの 4D に対応するものです。

複数行 = いいえ

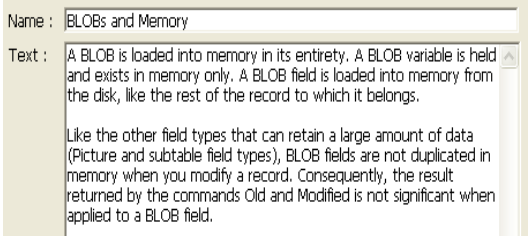
一行のエリアでは、行の終わりにある単語は切り取られ、行送りはありません。
行送りはありません。テキストは常に一行で表示されます。文字やテキストフィールドあるいは変数が改行を含む場合、最初の改行より後ろのテキストは、エリアが更新されると過ぎに取り除かれます :



複数行 = はい

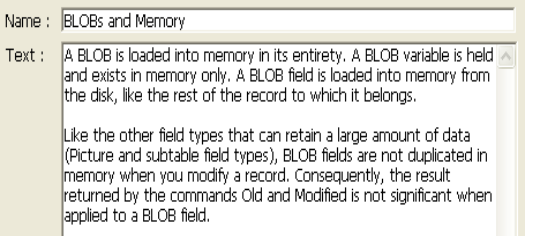
この場合、追加のオプションワードラップを設定する必要があります。

- **ワードラップ = 自動**
一行のエリアでは、テキストは最初の改行まで、または全体が表示可能な最後の単語までが表示されます。4D は行送りを挿入します。下矢印キーを押すことでエリアの内容をスクロールできます :

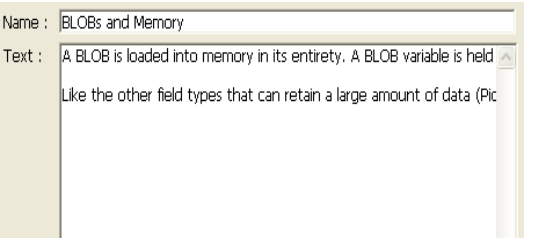


- **ワードラップ = はい**
一行のエリアでは、テキストは全体が表示可能な最後の単語までが表示されます。4D は行送りを挿入します。下矢印キーを押すことでエリアの

内容をスクロールできます。
複数行エリアでは、4D は自動で行送りを実行します :



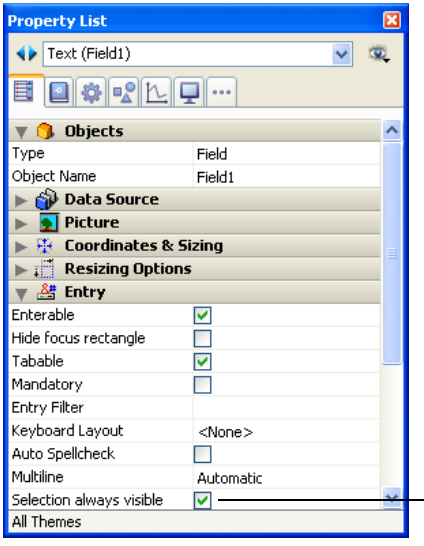
- **ワードラップ = いいえ**
4D は自動の行送りを行わず、表示可能な最後の単語は切り取られるかもしれません。テキストタイプのエリアでは改行がサポートされます :



Note: これらの設定が正しく動作するためには、テキストオブジェクトにスクロールバーをつけてはなりません。

常に選択を表示

文字やテキストタイプのフィールドや変数で新しいプロパティ**常に選択を表示**が利用できます :



新しいオプション

このプロパティを使用すると、オブジェクトがフォーカスを失った後もテキストの選択状態を保持することが可能になります。これにより、特にテキストのスタイルを変更するインタフェースの実装が容易になります (94 ページの " リッチテキスト " 参照)。

リストボックス

4D v12 ではリストボックスに関する新機能がいくつか追加されています :

- 階層リストボックスが指定可能
- リストボックスが印刷可能
- SELECT により追加された列のデータへのアクセス

階層リストボックス

4D v12 では階層リストボックスを指定できるようになりました。階層リストボックスとは、**一列目の内容**が階層形式で表示されるリストボックスのことです。このタイプの表現は繰り返される値や、(国、地域、都市など) 階層的に表現される値を含む情報の表示に使用されます。

配列タイプのリストボックスのみが階層になることができます。

階層リストボックスはデータを表示する特別な方法ですが、データの構造 (配列) を変更することはありません。階層リストボックスは通常のリストボックスとまったく同様に管理されます。詳細は 107 ページの " 階層リストボックスの 動作 " を参照してください。

階層リストボックスを指定するには、3 つの異なる方法があります :

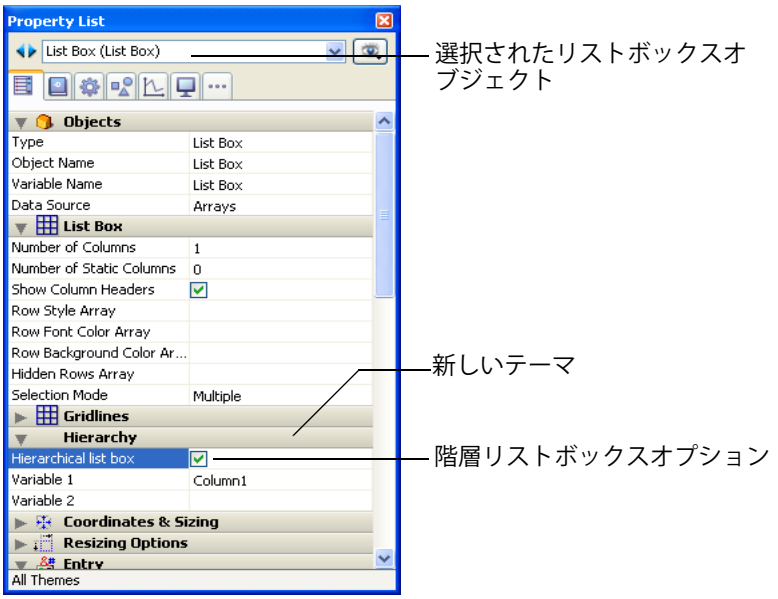
- フォームエディタのプロパティリストで階層要素を指定する。この方法は 103 ページの " 新しいリストボックスプロパティ " で説明します。
- フォームエディタで、リストボックス管理ポップアップメニューを使用して、視覚的に階層を生成する。これは 106 ページの " コンテキストメニューで階層を管理 " で説明します。
- 新しいランゲージコマンド LISTBOX SET HIERARCHY と LISTBOX GET HIERARCHY を使用する。詳細は 164 ページの " リストボックス " を参照してください。

新しいリストボックスプロパティ

新しいプロパティを使用して階層リストボックスを管理できます。リストボックスオブジェクトのポップアップメニューを使用して階層を指定すると、これらのプロパティも自動で変更されることに留意してください。

- **階層リストボックス**

階層リストボックスを指定するには、新しい**階層リストボックス**オプションにチェックします。このオプションはリストボックスオブジェクト選択時に利用できるプロパティリストの新しい " 階層 " テーマにあります :

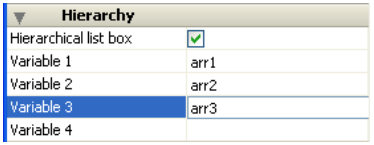


このオプションはデータソースが配列にリストボックスにのみ表示されます。

- **変数 1 ... 10**

階層リストボックスオプションがチェックされると、同テーマに入力可能フィールド " 変数 1", " 変数 2", 等が追加されます : 。値が入力されると新しい行が追加されます。10 個までの変数を指定できます。

これらの変数は最初の列に表示される階層レベルを設定します。



最初の変数は常にリストボックスの最初の列の変数名に対応します (2 つの値が自動でバインドされます)。この最初の変数は常に行事され、入力可能です。例 : 国

二番目の変数も常に表示され、入力可能です。この変数は二番目の階層レベルを指定します。例 : 地域

三番目のフィールド以降、それぞれの変数はその前の変数に基づき指定できるようになります。例 : 都道府県 , 市区 , など。最大 10 レベルの階層を指定できます。

値を取り除くと、後ろの階層レベルが上がります。

最後の変数は、このレベルに複数の同じ値があっても階層にはなりません。例えば以下のような例で、arr1 には A A A B B B、arr2 には 1 1 1 2 2 2、そして arr3 には X X Y Y Y Z が格納されているとします。この場合、A、B、1、そして 2 を折りたたまれた形式で表示 できますが、X と Y はできません :

▼ A
  ▼ 1
    X
    X
    Y
▼ B
  ▼ 2
    Y
    Y
    Z

この原則は、階層に変数が一つだけ指定されたときには適用されません。この場合、同じ値はグループ化されます。

Note: 既存のリストボックスで一番目の列をもとに階層を指定した場合、you must then remote or hide these columns (except for the first), otherwise they will appear in duplicate in the list box. If you specify the hierarchy via the pop-up menu of the editor (see the 106 ページの " コンテキストメニューで階層を管理 "), the unnecessary columns are automatically removed from the list box.

- **変更されたプロパティ**

階層モードが有効になる (" 階層リストボックス " オプションがチェックされる) と、いくつかの他のプロパティが無効になるか取り除かれます :

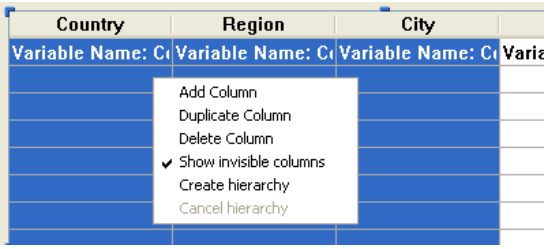
- **固定列の数** : 階層リストボックスの場合、このオプションは常に少なくても 1 です。
- **行の移動可** : 階層リストボックスでこのオプションは利用できません。
- 一番目の列の " 表示 " テーマのすべてのプロパティは無効になります。

コンテキストメニューで階層を管理

リストボックスオブジェクト上でクリックをしたさに表示されるコンテキストメニューで、2 つの新しいコマンドを利用できます : **階層を作成**と**階層をキャンセル**

- **階層を作成**

フォームエディタで配列タイプのリストボックスオブジェクトで最初の列に加えさらに列を選択すると、**階層を作成**コマンドがコンテキストメニュー中で利用できるようになります :

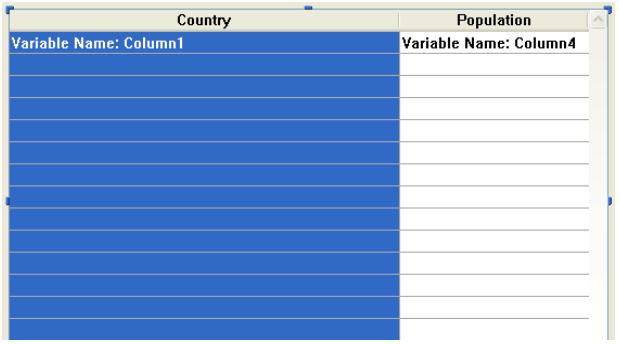


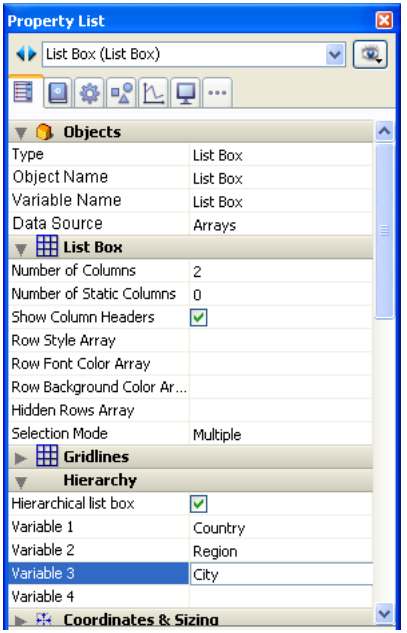
このコマンドを選択すると、以下のアクションが実行されます :

- そのオブジェクトの " 階層リストボックス " オプションがチェックされます (104 ページの " 階層リストボックス " 参照)。
- 階層を指定するために列変数が使用されます (104 ページの " 変数 1 ... 10" 参照)。既に指定済みの変数は置き換えられます。
- 選択された列はリストボックスに表示されなくなります (最初の列のタイトルを除く)。

例 : リストボックスがあり、先頭から国、地域、市区、人口が列に割り当てられています。国、地域、市区が選択されているとき (上図参照)、コンテキストメニューから**階層を作成**メニューを選択すると、階層が最初

の列に作成され、二番目と三番目の列が取り除かれます。人口は二番目の列になります。:





- **階層をキャンセル**

先頭の列が選択され既に階層が指定されているとき、**階層をキャンセル**コマンドを使用できます。このコマンドを選択すると、以下のアクションが実行されます:

- オブジェクトの"階層リストボックス"オプションの選択が外されます。
- 階層レベルが削除され、列としてリストボックスに追加されます。

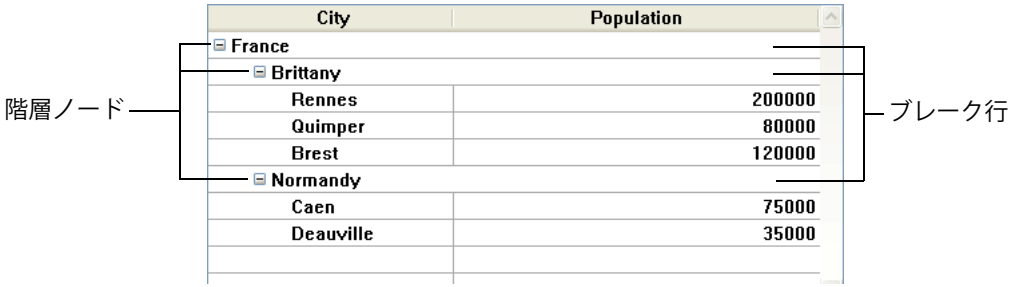
階層リストボックスの動作

階層リストボックスを含むフォームが最初に開かれるとき、デフォルトですべての行が展開されて表示されます。

配列中で値が繰り返されているとき、ブレーク行と階層ノードが自動でリストボックスに追加されます。例えば国、地域、名前、そして人口データを持つ市区が、リストボックスに 4 つの配列で含まれているとします:

Country	Region	City	Population
France	Brittany	Rennes	200000
France	Brittany	Quimper	80000
France	Brittany	Brest	120000
France	Normandy	Caen	75000
France	Normandy	Deauville	35000

このリストボックスが階層形式で表示されると、最初の 3 つの配列は階層に含まれ、以下のように表示されます :



階層が構築される前に配列はソートされていません。例えばもし配列がデータ AAABBAACC で構成されていると、取得される階層は以下のようになります :

- A
- B
- A
- C

階層ノードを展開あるいは折りたたむにはノードをクリックします。**Alt+click** (Windows) あるいは **Option+click** (Mac OS) を行うとすべてのサブ要素が展開されたり折りたたまれたりします。

選択や位置の管理

階層リストボックスはノードの展開 / 折りたたみ状態により、スクリーン上に表示される行数が変わります。しかし配列の行数が変わるわけではありません。表示が変わるだけでデータに変更はありません。

この原則を理解することは重要です。階層リストボックスに対するプログラムによる管理は常に配列データに対して行われるのであり、表示されたデータに対して行われるわけではないからです。特に、自動で追加されるブレーク行は、表示オプション配列では考慮されません (110 ページの " ブレーク行の管理 " 参照)。

例として以下の配列を見てみましょう :

France	Brittany	Brest
France	Brittany	Quimper
France	Brittany	Rennes

これらの配列が階層的に表示されると、2 つのブレーク行が追加されるため、"Quimper" 行は 2 行目ではなく 4 行目に表示されます :

France
Brittany
Brest
Quimper
Rennes

階層であってもなくても、リストボックスにどのようにデータが表示されているかにかかわらず、"Quimper" が含まれる行を太字にしたい場合はステートメント Style{2} = bold を使用しなければなりません。配列中の行の位置のみが考慮されます。

この原則は以下のものを管理する内部的な配列に適用されます :

- カラー
- 背景色
- スタイル
- 非表示行
- 選択行

例えば Rennes を含む行を選択したい場合、以下のように書きます :

```
-> MyListbox{3}:=True
```

非階層表示 :

France	Brittany	Brest
France	Brittany	Quimper
France	Brittany	Rennes

階層表示 :

France
Brittany
Brest
Quimper
Rennes

Note: 親が折りたたまれているために行が隠されていると、それらは選択されなくなります。(直接あるいはスクロールで) 表示されている行のみを選択できます。言い換えれば行を選択かつ隠された状態にすることはできません。

選択と同様、LISTBOX GET CELL POSITION コマンドは階層リストボックスと非階層リストボックス同じ値を返します。つまり以下の両例題で、LISTBOX GET CELL POSITION は同じ位置 (3;2) を返します。

非階層表示 :

France	Brittany	Brest	120000
France	Brittany	Quimper	80000
France	Brittany	Rennes	200000
France	Normandy	Caen	75000

階層表示 :

France		
Brittany		
	Brest	120000
	Quimper	80000
	Rennes	200000
Normandy		
	Caen	75000

ブレーク行の管理

ユーザがブレーク行を選択すると、LISTBOX GET CELL POSITION は対応する配列の最初のオカレンスを返します。以下のケースで :

France		
Brittany		
	Brest	120000
	Quimper	80000
	Rennes	200000
Normandy		
	Caen	75000

LISTBOX GET CELL POSITION は (2;4) を返します。プログラムでブレーク行を選択するには新しいコマンド LISTBOX SELECT BREAK、ページ 171 を使用しなければなりません。

ブレーク行はリストボックスのグラフィカルな表示 (スタイルやカラー) を管理する内部的な配列では考慮されません。しかしオブジェクトのグラフィックを管理する " オブジェクトプロパティ " テーマのコマンドを使用してブレーク行の表示を変更できます。階層を構成する配列に対して、適切なコマンドを実行します。

以下のリストボックスを例題とします (割り当てた配列名は括弧内に記載しています):

非階層表示 :

(T1)	(T2)	(T3)	(T4)	(tStyle)	(tColor)
France	Brittany	Brest	120000	*Normal*	*0*
France	Brittany	Quimper	80000	*Underline*	*0*

非階層表示 :

France	Brittany	Rennes	200000	*Normal*	*0xFF0000*
France	Normandy	Caen	220000	*Normal*	*0*
France	Normandy	Deauville	4000	*Normal*	*0*

階層表示 :

France	
Brittany	
Brest	120000
Quimper	80000
Rennes	200000
Normandy	
Caen	75000
Deauville	4000

階層モードでは tStyle や tColors 配列で変更されたスタイルは、ブレーク行に適用されません。ブレークレベルでカラーやスタイルを変更するには、以下のステートメントを実行します :

```
OBJECT SET RGB COLORS(T1;0x0000FF;0xB0B0B0)
OBJECT SET FONT STYLE(T2;Bold)
```

このコンテキストでは、配列は割り当てられたオブジェクトがないため、配列変数を使用したシンタックスのみがオブジェクトプロパティコマンドで動作します。

結果 :

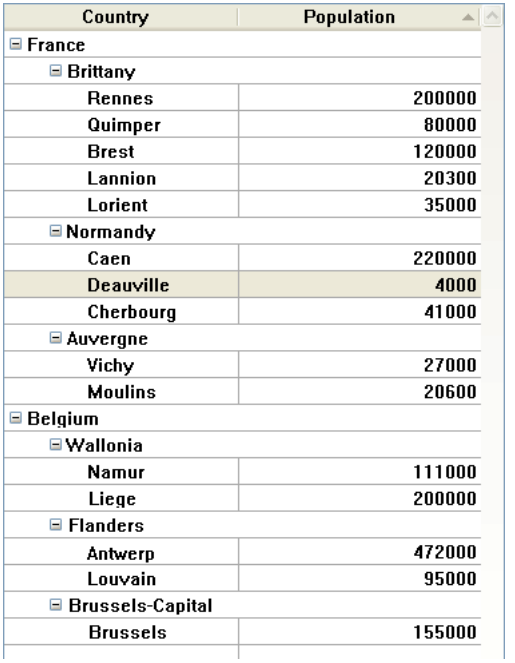
France	
Brittany	
Brest	120000
Quimper	80000
Rennes	200000
Normandy	
Caen	75000
Deauville	4000

ソートの管理

階層モードのリストボックスにおいて、(リストボックス列のヘッダをクリックして実行される) 標準のソートは綱に以下のように行われます :

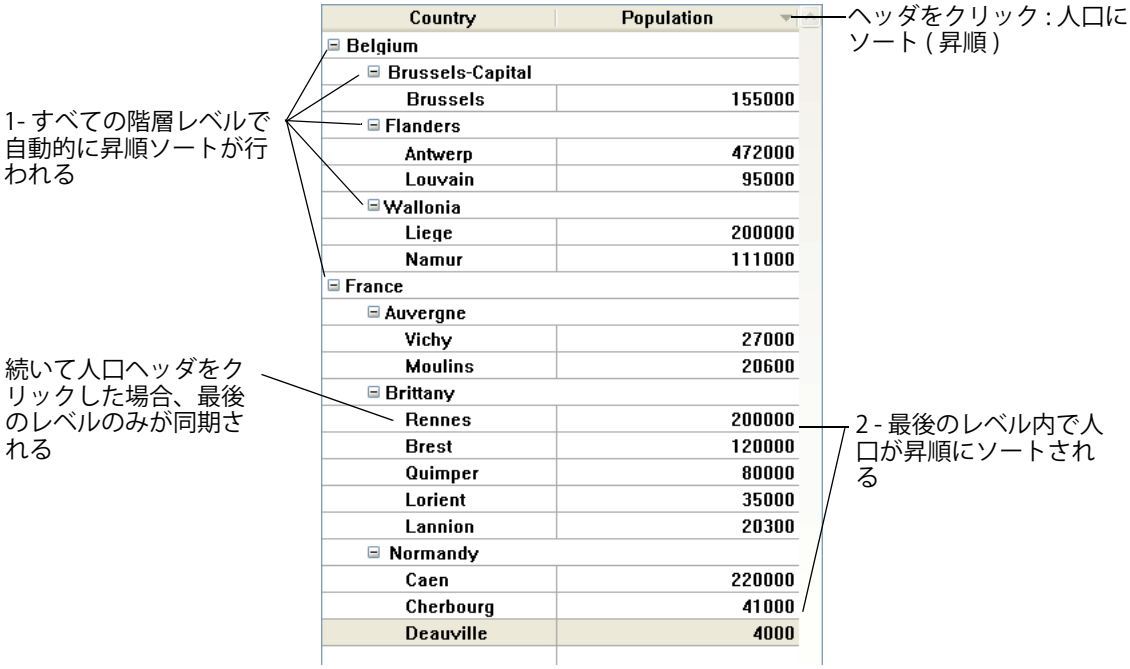
- まず階層列 (最初の列) のすべてのレベルで自動的に昇順にソートされます。
- そしてクリックされた列の値に対して、(ユーザアクションに応じ) 昇順または降順にソートされます。

- すべての列が同期されます。
- リストボックスの非階層列に対して続いて実行されるソートの間、最初の列の最後のレベルのみがソートされます。この列のソートをヘッダのクリックにより変更することができます。
- 例として以下のリストボックスを考えます。ソートはされていません:



Country	Population
France	
Brittany	
Rennes	200000
Quimper	80000
Brest	120000
Lannion	20300
Lorient	35000
Normandy	
Caen	220000
Deauville	4000
Cherbourg	41000
Auvergne	
Vichy	27000
Moulins	20600
Belgium	
Wallonia	
Namur	111000
Liege	200000
Flanders	
Antwerp	472000
Louvain	95000
Brussels-Capital	
Brussels	155000

人口に基づき昇順 (または降順) でソートするために "Population" ヘッダをクリックすると、データは以下のように表示されます :



すべてのリストボックスのように、リストボックスの " ソート可 " オプションを選択解除して、標準のソートメカニズムを無効にし、プログラムでソートを管理できます。

日付と時間の表示

階層リストボックスに日付や時間タイプの値が含まれるとき、これらは標準フォーマットで表示されます :

- 日付はシステム短表記フォーマットで表示されます (例えば 2009 年 5 月 30 日の場合、アメリカのシステムでは "05/30/09"、日本のシステムでは "09/05/30")。
- 時間もシステム短表記フォーマットで表示されます (システム設定により、"12:15:30" や "12:15")。

非表示行

サブ階層のすべての行が隠されているとき、ブレーク行は自動で隠されます。先の例題で 1 から 3 行目までが隠されていると、"Brittany" ブレーク行は表示されません。

スクロールと編集

OBJECT SET SCROLL POSITION (以前の SCROLL LINES) と EDIT ITEM コマンドは 階層モードで表示されるリストボックスに適合されました。対象

の行が折りたたまれた階層レベルに属する場合、このレベルおよびすべての親レベルは自動で展開され、その行が表示されます。

リストボックスの印刷

4D v12 ではリストボックスの印刷が可能になりました。2 つの印刷モデルが利用できます。**プレビュー**モードはリストボックスをフォームオブジェクトのように印刷し、**詳細**モードではフォーム内でのリストボックスオブジェクトの印刷をコントロールすることが可能です。リストボックスオブジェクトの " 印刷 " アピアランスを指定するための新しいプロパティを使用できます。

プレビューモード

プレビューモードでリストボックスを印刷することは、標準の印刷コマンドあるいは**プリント**メニューを使用して、リストボックスを含むフォームを直接印刷することです。リストボックスはフォーム上にあるとおりに印刷されます。

このモードでは緻密なオブジェクト印刷のコントロールができません。特にリストボックスが表示できる以上の行を持つリストボックスの、すべての行を印刷することはできません。

詳細モード

このモードではリストボックスの印刷は、新しい Print object を使用してプログラミングで実行されます。そのため、プロジェクトフォーム上のリストボックスのみを詳細モードで印刷できます。新しい LISTBOX GET PRINT INFORMATION コマンドを使用して、オブジェクトの印刷をコントロールできます。

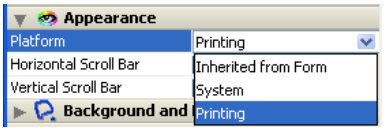
このモードでは :

- 印刷される行数がオブジェクトのもとの高さより小さい場合、リストボックスオブジェクトの高さは自動で縮められます。空の行は印刷されません。他方、オブジェクトの内容により、高さが自動で増やされることはありません。実際に印刷されるオブジェクトのサイズは LISTBOX GET PRINT INFORMATION コマンドで取得できます。
- リストボックスオブジェクトは " 現状のまま " 印刷されます。言い換えれば現在の表示パラメタ (ヘッダやグリッドラインの表示、非表示行など) が考慮されます。
  印刷される最初の行も考慮されます。印刷を実行する前に OBJECT SET SCROLL POSITION コマンドを呼び出すと、印刷される最初の行はコマンドで指定された行となります。
- リストボックスに表示される以上の行を印刷する自動的なメカニズムを使用できます。連続した Print object の呼び出しを使用して、新しい行を印刷できます。LISTBOX GET PRINT INFORMATION コマンドを使用して、

実行中の印刷のステータスをチェックできます。詳細はこれらのコマンドの説明を参照してください。

印刷プロパティ

プロパティリストのアピアランステーマで、新しいプラットフォームオプション**印刷**を使用できます。



このオプションを選択すると、リストボックスのアピアランスは印刷用に設定されます。ヘッダは白黒になり、チェックボックスは [x] で印刷されるなどします。印刷アピアランスはそれが実行されるプラットフォームの影響を受けません。
このオプションを選択しないと、リストボックスは現在のプラットフォームのアピアランスで印刷されます。

Note: 詳細モードで、選択されたアピアランスにかかわらず、リストボックスのスクロールバーが印刷されることはありません。

SELECT で追加された列のデータへのアクセス

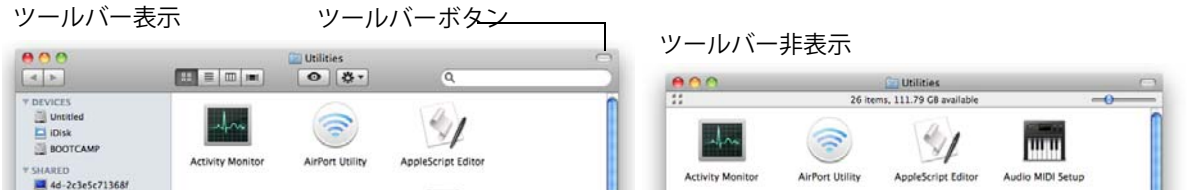
リストボックスに定義された数以上の列を生成する SELECT クエリの結果、4D がリストボックスに自動で追加した列の内容を取り出すことが可能になりました。

以前のバージョンでは、これらの追加された列は内部的な配列にリンクされ、4D ランゲージからアクセスすることはできませんでした。4D v12 では、これらの列は一時的な配列タイプにバインドされます。これらの一時的な配列の寿命はフォームと同じです。一時的な変数もヘッダ用に作成されます。GET LISTBOX ARRAYS コマンドが呼び出されると、arrColVars 引数には一時的な配列へのポインタが、また arrHeaderVars パラメタには一時的なヘッダ変数へのポインタが返されます。

これらの列に対しデータ入力やソートが可能になる点に留意してください。

ツールバーボタン (Mac OS)

Mac OS 上で、4D v12 はツールバー管理ボタンを表示できます。この標準のボタンはウィンドウのツールバーを表示したり隠したりするために使用できます。



この機能の 4D によるサポートは 2 つのレベルで実行されます :

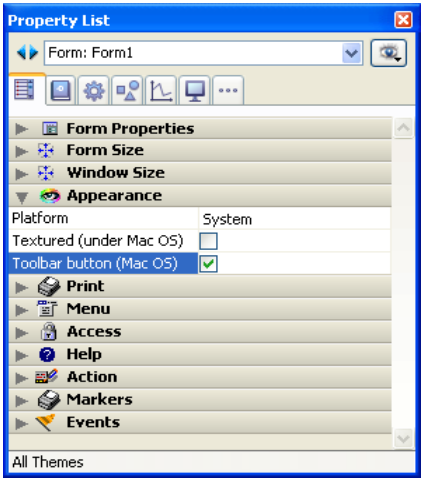
- ウィンドウにボタンを描画するために使用する新しいプロパティ。
- ユーザがボタンをクリックすると生成される新しいフォームイベント。

ボタンがクリックされたときに実行されるアクションの指定はデベロッパが行います。

ツールバープロパティ

4D にツールバー管理ボタンを表示させるには、2 つの方法があります :

- プロパティリストのアピアランステーマに追加された新しい**ツールバーボタン (Mac OS)** プロパティをチェックする :



このオプションは例えば Open form window や DIALOG コマンドでウィンドウを作成したときに使用されます。

- 新しい Has toolbar button Mac OS 定数を使用する。このオプションは Open window や Open form window コマンドで使用されます (241 ページの "Open window" 参照)。

On Mac toolbar button フォームイベント

新しい On Mac toolbar button フォームイベントは、Mac OS 上でユーザがウィンドウのツールバー管理ボタンをクリックしたときにフォームメソッドで生成されます。もちろん、対応するプロパティがフォームイベントプロパティでチェックされていなければなりません。

イベントだけが生成されます。4D はウィンドウ中で他のアクションをなにも行いません。ウィンドウのサイズを変更したり、インタフェース要素を表示したり隠したりするのは開発者の役割です。

5 ランゲージ

この章では、4D v12 のプログラム言語に関する、新しい機能や変更点について説明します。

文字列

Convert to text

Convert to text(BLOB ; charSet) → テキスト

Convert to text コマンドは BOM (Byte Order Marks) をサポートするようになりました。

指定された文字セットが Unicode タイプ (UTF-8、UTF-16 または UTF-32) の場合、4D は受け取った先頭バイトの BOM の識別を試みます。BOM 検知されるとそれは結果から取り除かれ、4D は指定された文字セットの代わりに検知した文字セットを使用します。

通信

RECEIVE PACKET

RECEIVE PACKET({docRef ;} receiveVar ; stopChar | numBytes)

RECEIVE PACKET コマンドは、receiveVar が文字型変数のとき、BOM (Byte Order Marks) をサポートするようになりました。

この場合、現在の文字セットが Unicode タイプ (UTF-8、UTF-16 または UTF-32) の場合、4D は受け取った先頭バイトの BOM の識別を試みます。BOM 検知されるとそれはデータを受信する変数から取り除かれ、4D は現在の文字セットの代わりに BOM で検知した文字セットを使用します。

ペーストボード

SET FILE TO PASTEBOARD

SET FILE TO PASTEBOARD(file{; *})

引数	型		説明
file	テキスト	←	ファイル名、または ファイルの完全パス名
*	*	→	指定時 = 追加 省略時 = 置き換え

自由度を増すために、SET FILE TO PASTEBOARD コマンドに直接 (完全パス名でなく) ファイル名を渡せるようになりました。

さらにコマンドはオプションの引数 * を受け入れるようになりました。この引数が省略された場合デフォルトで、コマンドはペーストボードの内容を file で指定された最新のパス名で置き換えます (コマンドの従来の動作)。
この引数を渡すと、コマンドは file をペーストボードに追加します。この方法で、スタックされたファイルのパス名を含むことができます。

両方のケースで、ペーストボードにパス名以外のデータが存在する場合、そのデータは消去されます。

ストラクチャアクセス

データベース中の " 実体のない " データを取り出すために使用できる、新しい 2 つのユーティリティコマンドがこのテーマに追加されました。

GET MISSING TABLE NAMES

GET MISSING TABLE NAMES(missingTables)

引数	型		説明
missingTables	テキスト配列	←	データベース中で失われたテーブルの名前

新しい GET MISSING TABLE NAMES コマンドはカレントデータベース中ですべての失われたテーブルの名前を missingTables 配列に返します。

失われたテーブルとは、データファイル中にデータがあるにもかかわらず、カレントストラクチャレベルに存在しないテーブルです。これはデータファイルが異なるバージョンのストラクチャで開かれたときに発生します。

典型的なシナリオは以下の通りです :

- 開発者はテーブル A、B、C を含むストラクチャを提供する。
- ユーザが (例えば統合された SQL を使用して) カスタムテーブル D と E を追加し、これらのテーブルにデータを格納する。
- デベロッパが新しいバージョンのストラクチャを提供する。このストラクチャにはテーブル D と E が含まれていません。

  この場合、ユーザバージョンのデータファイルにはテーブル D と E のデータが含まれていますが、アクセスすることはできません。GET MISSING TABLE NAMES コマンドはテーブル名 "D" と "E" を返します。

  データベースで失われているテーブルを識別したら、新しい REGENERATE MISSING TABLE コマンドを使用して、それらを再アクティブにすることができます。

Note: 失われたテーブルのデータは、テーブルが再生成されていないと、データファイル圧縮時に失われます。

REGENERATE MISSING TABLE

REGENERATE MISSING TABLE (tableName)

引数	型		説明
tableName	テキスト	→	再生成する、失われたテーブルの名称

新しい REGENERATE MISSING TABLE コマンドは、tableName 引数に渡された名前の失われたテーブルを再構築します。失われたテーブルが再構築されると、ストラクチャエディタにそれらが現れ、データに再びアクセスできるようになります。

失われたテーブルとは、データファイル中にデータがあるにもかかわらず、ストラクチャレベルに存在しないテーブルのことです。新しい GET MISSING TABLE NAMES コマンドを使用して、アプリケーション中に存在するかもしれない失われたテーブルを識別できます。

tableName で指定されたテーブルがデータベースの失われたテーブルでない場合、コマンドはなにも行いません。

- 以下のメソッドはデータベース中に存在するかもしれないすべての失われたテーブルを再生成します :

```
ARRAY TEXT($arrMissingTables;0)
GET MISSING TABLE NAMES($arrMissingTables)
$SizeArray:=Size of array($arrMissingTables)
If ($SizeArray#0)
  // 配列をデータベース中のテーブル名で埋める
  ARRAY TEXT(arrTables;Get last table number)
  If (Get last table number>0)  // テーブルが実際に存在すれば
    For ($vlTables;Size of array(arrTables);1;-1)
      If (Is table number valid($vlTables))
        arrTables{$vlTables}:=Table name($vlTables)
      Else
        DELETE FROM ARRAY(arrTables;$vlTables)
      End if
    End for
  End if
  For ($i;1;$SizeArray)
    If (Find in array(arrTables;$arrMissingTables{$i})=-1)
      CONFIRM(" テーブルを再生しますか : "+$arrMissingTables{$i}+"?")
      If (OK=1)
        REGENERATE MISSING TABLE($arrMissingTables{$i})
      End if
    Else
      ALERT(" テーブル "+$arrMissingTables{$i}+" を再生できません。
データベース中に同じ名前のテーブルがあります。")
    End if
  End for
Else
  ALERT(" 再生するテーブルがありません。")
End if
```

参照 : GET MISSING TABLE NAMES

システムドキュメント

XLIFF アーキテクチャに基づく複数言語インタフェース内のドキュメント管理が 4D v12 で拡張されました。新しいコマンドを使用して、カレントの言語に基づき、このアーキテクチャに統合されたドキュメントのパス名を取得できます。さらに 2 つの新しいコマンドはシステムと POSIX パス名の変換を容易にします。Select folder コマンドに新しい引数が追加され、追加のオプションが使用できるようになりました。

Note: "4D 環境 " テーマの新しい SET DATABASE LOCALIZATION コマンドを使用して、データベースのカレント言語を設定できるようになりました。

Convert path POSIX to system

Convert path POSIX to system(posixPath{; *}) → テキスト

引数	型		説明
posixPath	テキスト	→	POSIX パス名
*	*	→	エンコーディングオプション
戻り値	テキスト	←	システムシンタックス表現のパス名

新しい Convert path POSIX to system コマンドは POSIX (Unix) シンタックスで表現されたパス名をシステムシンタックスで表現されたパス名に変換します。

posixPath 引数に POSIX シンタックスで表現されたファイルやフォルダの完全パス名を渡します。これは ("/" から始まる) 絶対パスでなければなりません。ディスクパスを渡さなければならず、(ftp://ftp.mysite.fr などから始まる) ネットワークパスを渡すことはできません。

コマンドは現在のシステムシンタックスで表現された、ファイルやフォルダの完全パス名を返します。

オプションの * 引数を使用して posixPath 引数がエンコードされているかどうかを指定できます。posixPath 引数がエンコードされている場合、この引数を渡さなければなりません。そうでなければ正しく変換されません。このコマンドはエンコードなしのパス名を返します。

▶ Mac OS での例題

```
$path:=Convert path POSIX to system("/Volumes/machd/
                                    file 2.txt")
  // "machd:file 2.txt" を返す
$path:=Convert path POSIX to system("/Volumes/machd/
                                    file%202.txt";*)
  // "machd:file 2.txt" を返す
$path:=Convert path POSIX to system("/file 2.txt")
  // "machd:file 2.txt" を返す (machd は起動ディスク )
```

▶ Windows での例題

```
$path:=Convert path POSIX to system("c:/docs/file 2.txt")
  // "c:\docs\truc 2.txt" を返す

$path:=Convert path POSIX to system("c:/docs/
                                    file%202.txt";*)
  // "c:\docs\truc 2.txt" を返す
```

参照 : Convert path system to POSIX

Convert path system to POSIX

Convert path system to POSIX(systemPath{; *}) → テキスト

引数	型		説明
systemPath	テキスト	→	システムシンタックスで表現された、相対または絶対パス名
*	*	→	エンコーディングオプション
戻り値	テキスト	←	POSIX シンタックスで表現された絶対パス名

新しい Convert path system to POSIX コマンドはシステムシンタックスで表現されたパス名を POSIX (Unix) シンタックスで表現されたパス名に変換します。

systemPath 引数には、システムシンタックスで表現した、ファイルやフォルダのパス名を渡します (Mac OS や Windows)。このパスは絶対パス、あるいはデータベースフォルダ (データベースストラクチャファイルを含むフォルダ) からの相対パスです。コマンド実行時に、パスの要素が実際にディスク上に存在する必要はありません。コマンドはパス名の妥当性をテストしません。

コマンドはファイルやフォルダの POSIX シンタックスで表現した完全パス名を返します。systemPath に渡されたパスのタイプにかかわらず、コマンドは常に絶対パス名を返します。systemPath に相対パス名を渡すと、4D はデータベースフォルダのパス名を追加して返します。

オプションの * 引数を使用して POSIX のエンコーディングを指定できます。デフォルトで、Convert path system to POSIX は POSIX パスの特別文字をエンコードしません。* 引数を渡すと、特別文字は変換されます (例えば "My folder" は "My%20folder" になります)。

- Mac OS での例題

```
$path:=Convert path system to POSIX("machd:file 2.txt")
  // "/Volumes/machd/file 2.txt" を返す (machd が起動ディスクでも )

$path:=Convert path system to POSIX("machd:file 2.txt";*)
  // "/Volumes/machd/file%202.txt" を返す

$path:=Convert path system to POSIX("resources:images")
  // "/Volumes/machd/bases/basevideo/resources/images" を返す
```

- Windows での例題

```
$path:=Convert path system to POSIX("c:\docs\file 2.txt")
  // "c:/docs/file 2.txt" を返す

$path:=Convert path system to POSIX("\\srv\tempo\file.txt")
  // "//srv/tempo/file.txt" を返す
```

参照 : Convert path POSIX to system

Get localized document path

Get localized document path (relativePath) → テキスト

引数	型		説明
relativePath	テキスト	→	ローカライズされたバージョンを取得したいドキュメントの相対パス名
戻り値	テキスト	←	ローカライズされたドキュメントの絶対パス名

Get localized document path コマンドは、相対パスで指定された、xxx.lproj フォルダ内に存在するドキュメントの完全 (絶対) パス名を返します。

このコマンドは Resources フォルダおよび xxx.lproj (xxx は言語コード) サブフォルダが存在するマルチ言語アプリケーションアーキテクチャで使用しなければなりません。このアーキテクチャにおいて、4D はローカライズされた .xliff タイプや画像などのファイルを自動でサポートします。しかし開発者は他のタイプのファイルについても同じメカニズムを使用する必要があるかもしれません。

relativePath に検索するドキュメントの相対パス名を渡します。渡すパス名はデータベースの "xxx.lproj" フォルダの第一階層から相対でなければなりません。このコマンドはデータベースのカレント言語に対応する "xxx.lproj" フォルダを使用した完全パス名を返します。

Note: カレント言語は Resources の内容に基づき 4D が自動で (Get database localization コマンド参照)、あるいは新しい SET DATABASE LOCALIZATION コマンドで設定されます。

relativePath 引数はシステムまたは POSIX シンタックスを使用して表現できます。例えば :

- xsl/log.xsl (POSIX シンタックス : Mac OS および Windows で利用可)
- xsl\log.xsl (Windows のみ)
- xsl:log.xsl (Mac OS のみ)

コマンドから返される絶対パス名は常にシステムシンタックスで表されます。

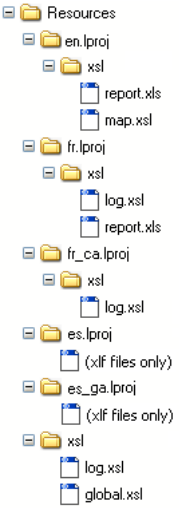
4D Server リモートモードで、コマンドがクライアントプロセスから呼び出された場合、クライアントマシンの Resources フォルダのパスが返されます。

4D は処理されるマルチ言語アプリケーションの、ありうるすべてのケースのシーケンスを試しながら、ファイルを探します。ステップごとに 4D は言語に対応するフォルダ内で relativePath の存在を検証し、ファイルを見つければその完全パスを返します。relativePath が見つからないかフォルダが存在しない場合、4D 次のステップを試みます。検索ステージ毎のフォルダは以下のようになります :

カレント言語 (例 : fr-ca)
地域なしのカレント言語 (例 : fr)
開始時にデフォルトでロードされる言語 (例 : ja-jp)
開始時にデフォルトでロードされる地域なし言語 (例 : ja)
最初に見つかった .lproj フォルダ (例 : it.lproj)
Resources フォルダの第一レベル

relativePath がこれらのパスのどこにも存在しない場合、コマンドは空の文字列を返します。

- XML や HTML ファイルを変換する目的で、"log.xsl" 変換ファイルを使用したいとします。このファイルはカレント言語により異なるため、どの "log.xsl" ファイルを使用するか決定する必要があります。
Resources フォルダの中身は以下のようになっています :



カレント言語に適用する .xsl ファイルを決定するには、以下のコードを使用します :

```
$myxsl:=Get localized document path ("xsl/log.xsl")
```

カレントの言語が日本語である場合 (ja)、コマンドは以下を返します :

- Windows: C:\users\...\...\...\resources\ja.lproj\xsl\log.xsl"
- Mac OS: "HardDisk:users:...:...:...:resources:ja.lproj:xsl:log.xsl"

Select folder

Select folder{(message{; defaultPath{; options}})} → テキスト

引数	型		説明
message	テキスト	→	ウィンドウのタイトル
defaultPath	テキスト | 倍長整数	→	パス名または記憶されたパス名の番号
options	倍長整数	→	選択オプション
戻り値	テキスト	←	選択されたフォルダへのアクセスパス

Select folder コマンドはオプションの引数を受け入れるようになりました。この引数により Mac OS 下で追加の機能を利用できるようになります。

options に、"System Documents" テーマの以下の定義済み定数を渡すことができます :

- Package open (2): パッケージをフォルダとして開くことを許可し、結果その内容を表示 / 選択できるようにします。この定数が使用されない場合、デフォルトでパッケージは開かれません。
- Use Sheet Window (16): 選択ダイアログボックスをシートウィンドウとして開きます。シートウィンドウは Mac OS X インタフェース特有のものであり、グラフィックアニメーションが使用されます。この定数が使用されない場合、デフォルトでコマンドは標準のダイアログボックスを表示します。

ひとつあるいは両方を組み合わせた定数を渡せます。

これらのオプションは Mac OS 下でのみ有効となります。Windows では、これらのオプションを渡されても無視されます。

4D 環境

新しい SET DATABASE LOCALIZATION コマンドを使用して、特にアプリケーションインタフェース (XLIFF アーキテクチャ) で使用されるデータベースのカレント言語を変更できます。Version type は 64-bit 4D Server を識別するための新しいコマンドを受け入れます。さらに SET DATABASE PARAMETER と Get database parameter コマンドに新しいセレクタが追加されました。

SET DATABASE LOCALIZATION

SET DATABASE LOCALIZATION(language{; *})

引数	型		説明
languageCode	テキスト	→	言語セレクタ
*	*	→	コマンドのスコープ

新しい SET DATABASE LOCALIZATION コマンドを使用して、カレントセッションのデータベースカレント言語を変更できます。

データベースのカレント言語は、アプリケーションのローカライズされた要素 (テキストおよびピクチャ) をプログラムが検索する場所である .lproj フォルダを指定します。デフォルトで 4D は、Resources の内容およびシステム環境に基づき自動でカレント言語を決定します (Get database localization コマンドの説明参照)。SET DATABASE LOCALIZATION を使用して、デフォルトのカレント言語を変更できます。

コマンドは既にロードされたフォームの言語を変更しません。コマンドが呼び出された後に表示された要素のみが、新しい設定の効果を得ます。

languageCode にはアプリケーションで使用する、RFC 3066、ISO639 そして ISO3166 標準で規定された言語コードを渡します。例えば日本語であれば "ja"、フランス語なら "fr"、アメリカ英語なら "en-us" を渡します。この標準に関する詳細や、渡すことが可能な値については Design Reference マニュアルを参照してください。

デフォルトで、コマンドは開かれたすべてのデータベースとコンポーネント、およびすべてのプロセスに適用されます。オプションの * 引数が渡されると、このコマンドを呼び出したデータベースにのみ適用されます。この引数は特に、データベースとコンポーネントで別々に言語を指定するために使用されます。

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

Note: RFC に従い、コマンドは言語コードと地域コードを分けるために "-" (ハイフン) を使用します (例えば "fr-ca" や "en-us")。他方、Resources フォルダ内の "lproj" フォルダは "_" (アンダースコア) を使用します (例えば "fr_ca.lproj" や "en_us.lproj")。

4D Server 4D Server では、コマンドを呼び出したリモートマシン上に存在する言語を利用できます。そのため、Resources フォルダが 同期されているか確かめなければなりません。

- 日本語をインタフェース言語として設定する場合 :

```
SET DATABASE LOCALIZATION("ja")
```

- アプリケーションで文字列参照 ":xliff:shopping" が使用されていて、XLIFF ファイルには以下のような情報が含まれています :
  - JA フォルダ :

```
<trans-unit id="15" resname="Shopping">
    <source>Shopping</source>
    <target> ショッピング </target>
</trans-unit>
```

  - FR フォルダ :

```
<trans-unit id="15" resname="Shopping">
    <source>Shopping</source>
    <target>Faire les courses</target>
</trans-unit>
```

```
SET DATABASE LOCALIZATION("fr")
  // 文字列 ":xliff:shopping" は "Faire les courses" を表示する
SET DATABASE LOCALIZATION("ja")
  // 文字列 ":xliff:shopping" は " ショッピング " を表示する
```

参照 : Get database localization

Get table fragmentation

Get table fragmentation (aTable) → 実数

引数	型		説明
aTable	テーブル	→	フラグメンテーション率を取得するテーブル
戻り値	実数	←	フラグメンテーションの割合

新しい Get table fragmentation コマンドは、aTable で指定されたテーブルの、レコードの論理的なフラグメンテーションの割合をパーセンテージで返します。

レコードの論理的なフラグメンテーション率は、データファイルにレコードが並んで格納されているかどうかを示します。フラグメント率が高くなると、テーブルに対する並び替えやシーケンシャル検索が大幅に遅くなります。フラグメント率が 0 はフラグメント状態にないことを示します。率が 20% を超えるようだと、データファイルの圧縮を考慮したほうがよいでしょう。

- このメンテナンスメソッドは、少なくとも 1 つのテーブルに大きなフラグメンテーションが発生したとき、圧縮を要求するようにするものです :

```
ToBeCompacted:=False
For ($i ;1;Get last table number)
  If(Is table number valid($i))
    If(Get table fragmentation(Table($i)->)>20)
      ToBeCompacted:=True
    End if
  End if
End for
If(ToBeCompacted)
  // 圧縮をリクエストするマーカーを置く
End if
```

Get database localization

Get database localization {(languageType)} → テキスト

引数	型		説明
languageType	倍長整数	→	言語タイプ
戻り値	テキスト	←	使用されるランゲージコード

Note: Get database localization コマンドは以前のバージョンでは Get current database localization という名称でした。

Get database localization コマンドは新しい引数を受け入れ、取得する言語のタイプを指定できるようになりました。実際、アプリケーション内で同時に複数の言語設定を使用できます。

languageType には "4D Environment" テーマ内の以下の定数を渡します :

languageType (値)	説明
Default localization(0)	Resources フォルダとシステム環境に基づき、起動時に 4D が自動で設定する言語 (変更不可)。
Current localization(1)	アプリケーションの現在の言語 : デフォルト言語あるいは SET DATABASE LOCALIZATION コマンドで設定された言語。
User system localization(2)	システムのカレントユーザにより設定された言語。
Internal 4D localization(3)	ソートや検索のために 4D が使用する言語 (アプリケーション設定で設定可能)

参照 : SET DATABASE LOCALIZATION

Version type

Version type→ 倍長整数

引数	型		説明
戻り値	倍長整数	←	0 = 32-bit フルバージョン 1 = 32-bit デモバージョン 2 = 64-bit バージョン

Version type コマンドは、64-bit 4D Server と 32-bit バージョンを区別する新しい値を返すようになりました。

これらの値は "4D Environment" テーマの 2 つの新しい定数でアクセスできます :

定数	型	値
64 bit Version	倍長整数	2

SET DATABASE PARAMETER, Get database parameter

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

引数	型		説明
aTable	テーブル	→	パラメタを設定するテーブル
selector	倍長整数	→	更新するデータベースパラメタのコード
value	実数 | 文字	→	パラメタの値

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

引数	型		説明
aTable	テーブル	→	パラメタを取得するテーブル
selector	倍長整数	→	データベースパラメタのコード
stringValue	文字	←	パラメタの文字値
戻り値	実数	←	パラメタの値

- value 引数と Get database parameter コマンドの戻り値の型が変更され、実数型となりました。この変更は 64-bit 4D Server においてメモリをアドレスするために使用する値を新しくサポートするために必要でした (17 ページの "4D Server Windows 64-bit" 参照)。
- **セレクタ 34** (Debug Log Recording) は新しい値を受け入れます。このセレクタは値付きの詳細モードを有効にする値 3 を受け入れるようになりました。この詳細モードでは、コマンドやプロジェクトメソッド、プラグインコマンド等に渡された引数の値もデバッグファイルに記録されます。
- 以下の新しいセレクタが利用可能になりました :

  セレクタ = 55 (PHP Interpreter IP address)

  - 値 : "nnn.nnn.nnn.nnn" タイプのフォーマット文字列 (例えば "127.0.0.1")。
  - 説明 : FastCGI を経由して PHP インタプリタと通信を行うために、4D がローカルで使用する IP アドレス。デフォルトで値は "127.0.0.1" です。このアドレスは 4D が配置されているマシンに対応しなければなりません。このパラメタはデータベース設定を使用してすべてのマシン用にグローバルに設定できます (39 ページの "PHP" 参照)。
    PHP インタプリタに関する詳細は 62 ページの "4D で PHP スクリプトを実行する " を参照してください。

セレクタ = 56 (PHP Interpreter port)

- 値 : 正の倍長整数値。
- 説明: 4DのPHPインタプリタが使用するTCPポート番号。デフォルトで8002 です。

セレクタ = 57 (PHP Number of children)

- 値 : 正の倍長整数値。
- 説明 : 4D の PHP インタプリタがローカルで作成し、管理する子プロセスの数。デフォルトで値は 5 です。
  最適化の目的で、スクリプト実行リクエストを処理するために、PHP インタプリタは " 子プロセス " と呼ばれるシステムプロセスのセット (プール) を作成、使用します。アプリケーションのニーズに基づき、子プロセス数の数を変更できます。
  詳細は 62 ページの " アーキテクチャ " を参照してください。

Note: Mac OS では、すべての子プロセスは同じポートを共有します。
Windows では、それぞれの子プロセスが個別のポート番号を使用します。最初の番号は PHP インタプリタ用に設定された番号で、他の子プロセスは最初の番号をインクリメントします。例えばデフォルトポート番号が 8002 で、5 個の子プロセスを起動すると、ポート 8002 から 8006 が使用されます。

Note: データベース設定を使用してこの値をすべてのマシン用にグローバルに設定できます (39 ページの "PHP" 参照)。

セレクタ = 58 (PHP Max requests)

- 値 : 正の倍長整数値。
- 説明 : PHP インタプリタが受け入れるリクエストの最大数。デフォルト値は 500 です。この最大値に達すると、インタプリタは "server busy" タイプのエラーを返します。セキュリティおよびパフォーマンスのため、この値を変更できます。このパラメタに関する詳細は FastCGI-PHP のドキュメントを参照してください。

Note: 4D 側では、これらのパラメタは動的に適用されます ; 4D を終了しなくても設定は有効になります。他方 PHP インタプリタが既に起動されている場合、これらの設定を有効にするにはインタプリタを再起動する必要があります。

Note: データベース設定を使用してこの値をすべてのマシン用にグローバルに設定できます (39 ページの "PHP" 参照)。

セレクタ = 60 (PHP Use external interpreter)

- 値 : 0 = 内部インタプリタを使用、1 = 外部インタプリタを使用。
- 説明: 4D の PHP リクエストを 4D の内部インタプリタに送信するか、外部インタプリタに送信するかを指定する値。デフォルトで値は 0 (4D のインタプリタを使用) です。独自にインストールした PHP インタプリタを使用したい場合、例えば追加のモジュールを使用したり特定の設定を使用したい場合は、value に 1 を渡します。この場合、4D は PHP リクエストの場合でも内部インタプリタを起動しません。
  カスタム PHP インタプリタは FastCGI でコンパイルされ、4D エンジンと同じマシン上に配置されていなければなりません。この場合、開発者がインタプリタを完全に管理しなければならない点に留意してください。4D はインタプリタを開始したり停止したりしません。

Note: データベース設定を使用してこの値をすべてのマシン用にグローバルに設定できます (39 ページの "PHP" 参照)。

セレクタ = 66 (Cache unload minimum size)

- 値 : 1 より大きい正の倍長整数値。
- 説明: データベースキャッシュにオブジェクトを配置するためにエンジンがメモリの空きスペースを必要としたときに、キャッシュから解放を行うメモリの最小サイズ (バイト単位)。
  このセレクタの目的はキャッシュからデータが解放される回数を減らすことで、パフォーマンスを向上させることにあります。キャッシュのサイズや、データベース中で処理されるデータのブロックサイズに応じこの設定を変更できます。
  空きが必要になった時、デフォルトで 4D は最低キャッシュの 10% をアンロードします。

OPEN 4D PREFERENCES

OPEN 4D PREFERENCES (selector)

引数	型		説明
selector	文字列	→	環境設定や設定ダイアログボックス中のパラメタテーマやページを指定するキー
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をアクティブなままにする

4D v12 で環境設定の動作が完全に再構成されたため (29 ページの " 環境設定の再構成 " 参照)、selector 引数に渡すキーのリストが変更されまし

た。さらに OPEN 4D PREFERENCES コマンドは追加の access 引数を受け入れるようになりました。

２つのダイアログボックスを使用してそれぞれ、4D アプリケーションに関連するパラメタ (環境設定) と、開かれたアプリケーションに関するパラメタ (データベース設定) を設定できます。OPEN 4D PREFERENCES コマンドから両ダイアログボックスにアクセスできます。このコマンドからこれらのダイアログボックスが呼び出されると、モーダルウィンドウに表示されます。

selector 引数に使用できるキーのリストは以下の通りです。キーの最初の部分はダイアログボックスを示しています :

/4D
/4D/General
/4D/Structure
/4D/Form editor
/4D/Method editor
/4D/Client-Server
/4D/Shortcuts
/Database
/Database/General
/Database/Mover
/Database/Interface
/Database/Interface/Developper
/Database/Interface/User
/Database/Interface/Shortcuts
/Database/Compiler
/Database/Database
/Database/Database/Data storage
/Database/Database/Memory and cpu
/Database/Database/International
/Database/Backup
/Database/Backup/Scheduler
/Database/Backup/Configuration
/Database/Backup/Backup and restore
/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
/Database/Compatibility
/Database/Security

無効な値あるいはスラッシュ ("/") のみを渡した場合、コマンドはデータベース設定ダイアログの最後に閲覧されたページを開きます。

互換性に関する注意 : コマンドは以前のバージョンのキーを使用しても引き続き動作します。対応は 4D が管理します。しかしながら v12 のキーに置き換えることをお勧めします。

新しい access 引数を使用して、環境設定やデータベース設定ダイアログボックスの他のページをロックすることで、ユーザアクションを制御できます。ユーザに特定の設定に関する変更を許可し、他の部分については許可しないようにすることができます。この場合 access 引数に True を渡して、selector 引数で指定したページだけをアクティブにし、変更可能にします。他のページへのアクセスはロックされます (ナビゲーションバーのボタンをクリックしても何も起きません)。access 引数に False を渡すか省略すると、ダイアログボックスのすべてのページにアクセスできます。

フォームイベント

CALL SUBFORM CONTAINER

CALL SUBFORM CONTAINER(event)

引数	型		説明
event	倍長整数	→	送信するイベント

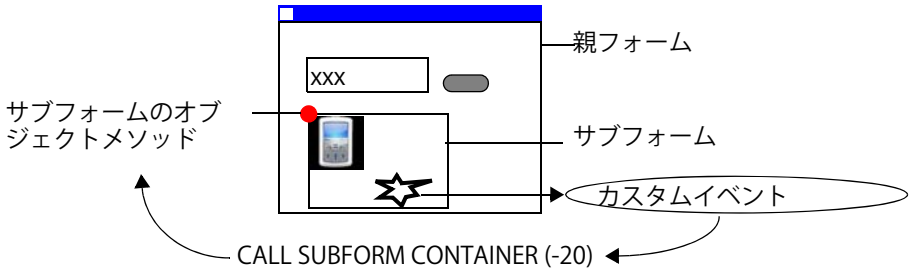
新しい CALL SUBFORM CONTAINER を使用して、サブフォームインスタンスからそれを含むサブフォームコンテナオブジェクトにイベントを送信できます。そしてサブフォームコンテナは親フォームのコンテキストでイベントを処理できます。

このコマンドはサブフォームのフォームメソッドまたはサブフォームオブジェクトのオブジェクトメソッドで使用しなければなりません。サブフォームコンテナのオブジェクトメソッドだけがイベントを受信します。

event には、サブフォームコンテナオブジェクトに利用できる 4D の定義済みフォームイベント ("Form Events" テーマの定数を使用できます) 、あるいはカスタムイベントに対応する値をを渡すことができます。前者の場合、イベントはサブフォームコンテナオブジェクトに対して有効にさ

れていなければなりません。カスタムイベントの場合、既存あるいは将来の 4D イベント番号との重複を避けるため、event に負数を渡すことを推奨します。

CALL SUBFORM CONTAINER コマンドの実行例



4D v12 におけるサブフォームのメカニズムに関する詳細は、71 ページの " サブフォームの能力の拡張 " を参照してください。

フォーム

このテーマのコマンドは名称が変更され、また 2 つのコマンドが追加されました。

名称が変更されたコマンド

より標準化されたものにするため、" フォーム " テーマのいくつかのコマンドの名称が 4D v12 で変更されました。"FORM" 接頭辞がシステマチックに追加されました。これらのコマンドの動作は変更されていません。

4D v12 での新しい名称	以前の名称
FORM FIRST PAGE	FIRST PAGE
FORM Get current page	Current form page
FORM GET OBJECTS	GET FORM OBJECTS
FORM GET PARAMETER	GET FORM PARAMETER
FORM GET PROPERTIES	GET FORM PROPERTIES
FORM GOTO PAGE	GOTO PAGE
FORM LAST PAGE	LAST PAGE
FORM NEXT PAGE	NEXT PAGE
FORM PREVIOUS PAGE	PREVIOUS PAGE
FORM SET HORIZONTAL RESIZING	SET FORM HORIZONTAL RESIZING
FORM SET VERTICAL RESIZING	SET FORM VERTICAL RESIZING
FORM SET SIZE	SET FORM SIZE
FORM SET INPUT	INPUT FORM
FORM SET OUTPUT	OUTPUT FORM
FORM GET PARAMETER	GET FORM PARAMETER

FORM GET HORIZONTAL RESIZING

FORM GET HORIZONTAL RESIZING (resize{; minWidth{; maxWidth}})

引数	型		説明
resize	ブール	←	True: フォームを水平方向にリサイズ可 False: フォームを水平方向にリサイズ不可
minWidth	倍長整数	←	最小フォーム幅 (ピクセル)
maxWidth	倍長整数	←	最大フォーム幅 (ピクセル)

新しい FORM GET HORIZONTAL RESIZING コマンドはカレントフォームの水平サイズ変更プロパティを resize、minWidth、そして maxWidth 変数に返します。これらのプロパティはデザインモードのフォームエディタ、またはカレントプロセス用に FORM SET HORIZONTAL RESIZING コマンドで設定されます。

FORM GET VERTICAL RESIZING

FORM GET VERTICAL RESIZING (resize{; minHeight{; maxHeight}})

引数	型		説明
resize	ブール	←	True: フォームを縦方向にリサイズ可 False: フォームを縦方向にリサイズ不可
minHeight	倍長整数	←	最小フォーム高さ (ピクセル)
maxHeight	倍長整数	←	最大フォーム高さ (ピクセル)

新しい FORM GET VERTICAL RESIZING コマンドはカレントフォームの垂直サイズ変更プロパティを resize、minHeight、そして maxHeight 変数に返します。これらのプロパティはデザインモードのフォームエディタ、またはカレントプロセス用に FORM SET VERTICAL RESIZING コマンドで設定されます。

入力制御

GOTO OBJECT

GOTO OBJECT({*;}object)

互換性に関する注意 GOTO OBJECT コマンドは以前のバージョンの 4D で GOTO AREA という名称でした。

GOTO OBJECT コマンドをサブフォームのコンテキストで使用できるようになりました。このコマンドがサブフォームから呼び出されると、まずオブジェクトをサブフォーム内で探します。オブジェクトが見つからなければ、親フォーム内のオブジェクトを検索します。

4D v12 におけるサブフォームの新機能については、71 ページの " サブフォームの能力の拡張 " を参照してください。

ドラッグ & ドロップ

Drop position

Drop position {(columnNumber | pictPosX)}→ 数値

引数	型		説明
columnNumber | pictPosX	倍長整数	←	リストボックス列番号または -1 または ピクチャ中の X 座標位置
戻り値	数値	←	• 数値 (配列 / リストボックス) または • 位置 (階層リスト) または • ドロップ先項目の文字列中の位置 (テキスト / コンボボックス) または最後の配列要素やリスト項目より後ろにドロップされた場合 -1 • ピクチャ中の Y 座標

Drop position はピクチャフィールドやピクチャ変数で発生したドラッグ & ドロップのコンテキストで動作するようになりました。

この場合、コマンドはクリックの縦位置を返し、オプションの pictPosX にクリックの横位置を返します。返される値はピクセルで、ローカルの座標システム内で相対的に表現されます。

参照 : OBJECT GET SCROLL POSITION, OBJECT SET SCROLL POSITION

ピクチャ

このテーマにはいくつかの新しいコマンドが追加され、ピクチャファイルのテスト (Is picture file) やメタデータの操作 (SET PICTURE METADATA および GET PICTURE METADATA) ができるようになりました。CONVERT PICTURE と PICTURE CODEC LIST コマンドは追加の引数を受け入れるようになり、3 つの廃止予定コマンドの名称が変更されました。

ピクチャのエンコードやデコードのための新しい API

4D v12 は Windows および Mac OS 上で、ピクチャ (フィールドや変数) のエンコードやデコードに新しいネイティブ API を使用します。この実装により、現在デジタルカメラで使用されている RAW フォーマットを含む、追加のネイティブフォーマットにアクセスができるようになります。

- Windows で、4D は WIC (Windows Imaging Component) を使用するようになりました。WIC はネイティブに以下のフォーマットをサポートします : BMP, PNG, ICO (デコードのみ), JPEG, GIF, TIFF そして WDP (Microsoft Windows Digital Photo)。
サードパーティーの WIC コーデックをインストールすることで、JPEG-2000 などの追加のフォーマットも使用できます。
- Mac OS で、4D は ImageIO を使用するようになりました。そのためすべての ImageIO コーデックがデコード (読み込み) およびエンコード (書き込み) でネイティブにサポートされます :

デコード	エンコード
public.jpeg	public.jpeg
com.compuserve.gif	com.compuserve.gif
public.png	public.png
public.jpeg-2000	public.jpeg-2000
com.nikon.raw-image	public.tiff
com.pentax.raw-image	com.adobe.photoshop.image
com.sony.arw-raw-image	com.adobe.pdf
com.adobe.raw-image	com.microsoft.bmp
public.tiff com.canon.crw-raw-image	com.truevision.tga-image
com.canon.cr2-raw-image	com.sgi.sgi-image
com.canon.tif-raw-image	com.apple.pict
com.sony.raw.image	com.ilm.openexr-image
com.olympus.raw-image	
com.konicaminolta.raw-image	
com.panasonic.raw-image	
com.fuji.raw-image	
com.adobe.photoshop-image	
com.adobe.illustrator.ai-image	
com.adobe.pdf	
com.microsoft.ico	
com.microsoft.bmp	
com.truevision.tga-image	
com.sgi.sgi-image	
com.apple.quicktime-image	
com.apple.icns	
com.apple.pict	
com.apple.macpaint-image	
com.kodak.flashpix-image	
public.xbitmap-image	
com.ilm.openexr-image	

public.radiance	

Windows で Mac OS と同様、サポートされるフォーマットは OS やマシンにインストールされているカスタム Codec により異なります。どの Codec が利用可能かを知るには PICTURE CODEC LIST コマンドを使用します。

WIC および ImageIO ではピクチャにメタデータが使用できることに留意してください。2 つの新しいコマンド SET PICTURE　METADATA と GET PICTURE METADATA が追加され、開発プロジェクトにおいてメタデータを利用できるようになりました。

Is picture file

Is picture file (filePath{*; })→ ブール

引数	型		説明
filePath	テキスト	→	ファイルパス名
*	*	→	データの検証
戻り値	ブール	←	True = filePath はピクチャファイルである、そうでなければ False

新しい Is picture file コマンドは filePath 引数で指定されたファイルをテストし、有効なピクチャファイルであれば True を返します。ファイルがピクチャファイルでないか、ファイルが見つからない場合は False を返します。

filePath 引数にはテストするピクチャファイルのパス名を渡します。パスはシステムシンタックスで指定しなければなりません。絶対パスまたははデータベースストラクチャファイルに対する相対パスを渡すことができます。空の文字列 ("") を渡すと、コマンドは False を返します。

* 引数を渡さない場合、コマンドはファイルの拡張子と利用可能なコーデックのリストを照合することでテストを行います。拡張子なしやより詳細なテストを行いたい場合は、* 引数を渡します。この場合、コマンドは追加のテストを行います。つまりファイルヘッダをロード・検査し、コーデックを照合してピクチャの妥当性を検査します。このシンタックスはコマンド実行を遅くします。

Note: コマンドは Windows で PDF ファイルに対し、Mac OS で EMF ファイルに対し、True を返します。

SET PICTURE METADATA

SET PICTURE METADATA (picture; metaName1; contents1{ ;...; metaNameN; contentsN})

引数	型		説明
picture	ピクチャ	→	メタデータを設定するピクチャ
metaName1...N	テキスト	→	設定するブロックの名前またはパス
contents1...N	変数	→	メタデータの内容

新しい SET PICTURE　METADATA コマンドを使用して、picture (4D ピクチャフィールドまたは変数) 内のメタデータ (またはメタタグ) の内容を書き込んだり更新したりできます。

メタデータはピクチャに挿入された追加の情報です。4D では 4 タイプの標準メタデータ EXIF, GPS, IPTC そして TIFF を処理できます。

Note: これらのメタデータタイプについては以下のドキュメントを参照できます :
http://www.iptc.org/std/IIM/4.1/specification/IIMV4.1.pdf (IPTC)
http://exif.org/Exif2-2.PDF (TIFF, EXIF, GPS)

metaName 引数には設定や更新を行うメタデータタイプを指定する文字列を渡します。以下を渡すことができます :

- 新しい "Picture metadata names" テーマの定数。このテーマには 4D がサポートするすべてのタグがグループ化されています。それぞれの定数はタグパスを含んでいます (例 "TIFF/DateTime")。
- メタデータの完全なブロック名 ("TIFF", "EXIF", "GPS" あるいは "IPTC")。
- 空の文字列 ("")。

contents 引数にはメタデータの新しい値を渡します :

- metaName にタグパス定数を渡した場合、設定する値を直接、または新しい "Picture metadata values" テーマの適切な定数を、contents 引数に渡せます。値は指定されたメタデータに応じて、テキスト、倍長整数、実数、日付、時間タイプを使用できます。メタデータが一つ以上の値を含む場合、配列を使用できます。文字列を渡すときは XML (XMP 標準) でフォーマットされていなければなりません。空の文字列 ("") を渡すと既存のメタデータが消去されます。
- metaName にブロック名か空の文字列を渡すときは、contents 引数には設定するメタデータを含む XML DOM 参照を渡します。空の文字列の場合、すべてのメタデータが更新されます。

Note: 多くの定数が追加されているため、新しい "Picture metadata names" と "Picture metadata values" テーマの定数は 319 ページの appendix C、" メタデータ定数 " で説明しています。

Windows では、コマンド実行中にエラーが発生すると、OK 変数が 0 に設定されます。
Mac OS 上では技術的な理由からメタデータ書き込みエラーを検知することはできません。そのためこのコマンドは Mac OS 上では OK システム変数を書き換えません。

- 配列を使用して "Keywords" メタデータの値を複数設定します :

```
ARRAY TEXT($arrTkeywords;2)
$arrTkeywords{1}:="France"
$arrTkeywords{2}:="Europe"
SET PICTURE METADATA(vPicture;"IPTC Keywords";$arrTkeywords)
```

- DOM 参照を使用して GPS ブロックを設定します :

```
C_TEXT($domMetas)
$domMetas:=DOM Parse XML source("metas.xml")
C_TEXT($gpsRef)
$gpsRef:=DOM Find XML element ($domMetas;"Metadatas/GPS")
If (OK=1)
   SET PICTURE METADATA(vImage;"GPS";$gpsRef)
      // ここで $gpsRef は GPS 要素を指しています
   ...
End if
DOM CLOSE XML($domMetas)
```

Note: すべてのメタデータが DOM 要素参照で処理されるとき、タグは名前がブロック名 (TIFF, IPTC 等) である要素に付加される属性として格納されます (参照された要素の子)。特定のメタデータブロックが操作されると、ブロックタグは、コマンドにより参照される要素に直接、属性として格納されます。

参照 : GET PICTURE METADATA

GET PICTURE METADATA

GET PICTURE METADATA (picture; metaName1; contents1{ ;...; metaNameN; contentsN})

引数	型		説明
picture	ピクチャ	→	メタデータを読むピクチャ
metaName1...N	テキスト	→	取得するブロックの名前またはパス
contents1...N	変数	←	メタデータの内容

新しい GET PICTURE METADATA コマンドを使用して、ピクチャ (4D のピクチャフィールドや変数) 内のメタデータ (またはメタタグ) の内容を読みだすことができます。メタデータに関する詳細は SET PICTURE METADATA コマンドの説明を参照してください。

metaName 引数には取り出すメタデータのタイプを指定する文字列を渡します。以下を渡すことができます :

- タグパスが含まれる新しい "Picture metadata names" テーマの定数
- メタデータの完全ブロック名 ("TIFF", "EXIF", "GPS" あるいは "IPTC")。
- 空の文字列 ("")。

contents 引数にはメタデータを受け取る変数を渡します :

- metaName にタグパスを渡した場合、contents 引数は直接取得した値を含みます。値は変数の型に合わせて変換されます。テキスト型の変数では XML (XMP 標準) でフォーマットされます。空の文字列 ("") を渡すと既存のメタデータが消去されます。(特に IPTC Keywords タグのように) メタデータに一つ以上の値が含まれる場合、配列を渡すことができます。
- metaName にブロック名か空の文字列を渡すとき、contents 引数は有効な XML DOM 要素参照でなければなりません。この場合、指定されたブロック (あるいは metaName に空の文字列を渡した場合はすべてのブロック) の内容は参照された要素に再コピーされます。

Note: 多くの定数が追加されているため、新しい "Picture metadata names" と "Picture metadata values" テーマの定数は 319 ページの appendix C、" メタデータ定数 " で説明しています。

メタデータの取得が正しく行われると、OK システム変数に 1 が設定され、エラーが発生したり 1 つ以上のタグが見つからない場合は 0 が設定されます。どのような場合でも、読みだされた値は返されます。

▶ DOM ツリーストラクチャを使用する :

```
$xml:=DOM Create XML Ref ("Root") \\XML DOM ツリーの作成

  \\TIFF メタデータの読み出し
$_Xml_TIFF:=DOM Create XML element ($xml;"/Root/TIFF")
GET PICTURE METADATA(vPicture;"TIFF";$_Xml_TIFF)

  \\GPS メタデータの読み出し
$_Xml_GPS:=DOM Create XML element($xml;"/Root/GPS")
GET PICTURE METADATA(vPicture;"GPS";$_Xml_GPS)
```

- 変数を使用する :

```
C_DATE($dateAsDate)
GET PICTURE METADATA("TIFF/DateTime";$dateAsDate)
  //$dateAsDate が日付型のため、日付のみが返される

C_TEXT($dateAsText)
GET PICTURE METADATA("TIFF/DateTime";$dateAsText)
  // 日付のみが XML フォーマットで返される

C_INTEGER($urgency)
GET PICTURE META(vPicture;"IPTC/Urgency";$urgency)
```

- タグの複数の値が配列に返される :

```
ARRAY TEXT($arrTkeywords;0)
GET PICTURE METADATA(vPicture;"IPTC/Keywords";$tTkeywords)
```

コマンドの実行後、$arrTkeywords は例えば以下のようになります :

```
  $arrTkeywords{1} = "France"
  $arrTkeywords{2} = "Europe"
```

- テキスト変数に複数の値を持つタグを受信する :

```
C_TEXT($vTwords;0)
GET PICTURE METADATA(vPicture;"IPTC/Keywords";$vTwords)
```

コマンド実行後、$vTwords は例えば "France;Europe" のようになります。

参照 : SET PICTURE　METADATA

CONVERT PICTURE

CONVERT PICTURE (picture; codec{; compression})

引数	型		説明
picture	ピクチャ	→	変換元ピクチャ
		←	変換後のピクチャ
codec	文字列	→	ピクチャ Codec ID
compression	実数	→	圧縮品質

CONVERT PICTURE コマンドは新しくオプションの compression 引数を受け入れ、互換のある Codec が使用されたときの結果のピクチャに適用する圧縮品質を指定できるようになりました。

compression には品質を指定する 0 から 1 の間の値を渡します。0 は低品質 (高圧縮) で 1 は最高品質 (低圧縮) です。この引数は Codec が圧縮をサポートする場合 (例えば JPEG や HDPhoto) のみ有効で、また WIC や

ImageIO API でサポートされています。したがって QuickTime のみで管理される Codec では利用できません。

Note: 4D v12 のピクチャ管理 API については、139 ページの " ピクチャのエンコードやデコードのための新しい API" を参照してください。

この引数を省略すると、デフォルトで最高品質が適用されます (compression =1)。

- 60% の品質でピクチャを変換する :

CONVERT PICTURE (vPicture;".JPG";0.6)

PICTURE CODEC LIST

PICTURE CODEC LIST (codecArray{; namesArray}{; *})

引数	型		説明
codecArray	文字配列	←	利用可能なピクチャ Codec の ID
namesArray	文字配列	←	ピクチャ Codec 名
*	*	→	読み込み (デコード) 用の Codec リストを返す

PICTURE CODEC LIST コマンドはピクチャのデコード (読み込み) に使用する Codec のリストを取得できるようになりました。これを行うにはオプションの * 引数を渡します。この引数が省略されるとデフォルトでコマンドはエンコード (書き込み) に使用できる Codec を返します。

2 つのリストは排他的ではありません。いくつかの Codec は読み込みおよび書き込みで使用できます。ピクチャエンコード用の Codec は通常デコードでも使用されます。他方、デコードに使用される Codec は、必ずしもエンコードに必要ではありません。例えば ".jpg" Codec は両方のリストにありますが、".xbmp" Codec は読み込み (デコード) Codec のリストにのみ存在します。

WRITE PICTURE FILE

WRITE PICTURE FILE (fileName ; picture {; codec})

WRITE PICTURE FILE コマンドの動作が変更され、ネイティブフォーマットでのピクチャの処理や格納が容易になりました。

今後 codec 引数を省略すると、コマンドは fileName 引数に渡されたファイル名の拡張子に基づき codec を判断しようとします。例えば以下のコードの場合 :
WRITE PICTURE FILE ("c:\folder\photo.jpg";myphoto)

コマンドは JPEG codec を使用してピクチャを格納します。

拡張子が利用可能な codec に対応しない場合、ファイルは保存されず OK システム変数に 0 が設定されます。codec もファイル拡張子も渡さない場合、PICT フォーマットで保存されます。

名称が変更されたコマンド

4D v12 で " ピクチャ " テーマの 3 つのコマンドの名称が変更されました。これらのコマンドは Apple の QuickTime 拡張に基づくものであり、廃止予定であることを示すため、"QT" が名称に付加されました :

4D v12 での新しい名称	以前の名称
QT COMPRESS PICTURE	COMPRESS PICTURE
QT LOAD COMPRESS PICTURE FROM FILE	LOAD COMPRESS PICTURE FROM FILE
QT COMPRESS PICTURE FILE	COMPRESS PICTURE FILE

これらのコマンドは 4D v12 では使用してはいけません。これらは互換性の目的でのみ保持されています。

読み込みと書き出し

このテーマに 2 つのコマンド、**IMPORT ODBC** と **EXPORT ODBC** が追加されました。これらのコマンドは以前 SQL IMPORT と SQL EXPORT という名称で、SQL テーマにありました。動作に変更はありません。2 つのテーマ間でコマンドの利用目的を明確にするためにこの変更が行われました。

印刷

4D v12 でおもに 2 つの点で印刷機能が強化されました :

- カスタマイズのため、新しい OPEN PRINTING FORM と Print object コマンドを使用して、フォームオブジェクトの印刷管理を完全に行うことが可能になります。
- オープンソースの PDFCreator ドライバの統合と、SET PRINT OPTION と GET PRINT OPTION コマンドによるドライバのサポートにより、Windows でも PDF を印刷できるようになりました。

OPEN PRINTING FORM

OPEN PRINTING FORM(form)

引数	型		説明
form	文字列	→	印刷のために開くプロジェクトフォームの名前、または 空の文字列を渡すと、カレントのプロジェクトフォームを閉じる

新しい OPEN PRINTING FORM コマンドを使用して、印刷用にプロジェクトフォームをロードできます。フォームをロードすると、それはカレントの印刷フォームとなります。すべてのオブジェクト管理コマンドや新しい Print object コマンドはこのフォームに対して動作します。

(OPEN PRINTING FORM を使用して) すでに印刷フォームがロードされている場合、このコマンドを呼び出すとそのフォームは閉じられ、form により置き換えられます。ひとつの印刷セッションで複数のプロジェクトフォームを開いたり閉じたりできます。OPEN PRINTING FORM で印刷フォームを変更してもページブレークは生成されません。ページブレークを管理するのは開発者の仕事です。form に空の文字列を渡すと、カレントの印刷プロジェクトフォームが閉じられます。

プロジェクトフォームを開く際は、"On Load" フォームイベントのみが実行されます。他のフォームイベントは無視されます。印刷の終わりには "On Unload" フォームイベントが実行されます。

フォームのグラフィックな一貫性を保持するために、プラットフォームにかかわらず " 印刷 " アピアランスプロパティを適用することをお勧めします。

CLOSE PRINTING JOB コマンドは呼び出されると、カレント印刷フォームは自動で閉じられます。

参照 : Print object

Print object

Print object({*;}object{;posX{;posY{;width{;height}}}}) → 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
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)
posX	倍長整数	→	オブジェクトの横位置
posY	倍長整数	→	オブジェクトの縦位置
width	倍長整数	→	オブジェクトの幅 (ピクセル)
height	倍長整数	→	オブジェクトの高さ (ピクセル)
戻り値	ブール	←	True = オブジェクトが完全に印刷された ; そうでなければ False

新しい Print object コマンドを使用して、object と * 引数で指定したフォームオブジェクトを、posX と posY の位置に、任意のサイズで印刷できます。

Print object コマンドはプロジェクトフォーム上のオブジェクトの印刷のみに使用できます。このコマンド呼び出し前に、印刷するオブジェクトを含むプロジェクトフォームを新しい OPEN PRINTING FORM で指定しなければなりません。

オプションの * 引数を渡すと、object 引数にはオブジェクト名 (文字列) を渡します。* 引数を渡さない場合、object には変数を指定します。この場合、文字列ではなく変数参照 (オブジェクトタイプのみ) を渡します。

posX と posY 引数はオブジェクトを印刷する開始位置を指定します。これらの値はピクセル単位で指定ます。これらの引数を省略すると、オブジェクトはフォーム上の位置に基づいて印刷されます。

width と height 引数はフォームオブジェクトの幅と高さを指定します。Print object コマンドは可変長のオブジェクトを管理しません。OBJECT GET BEST SIZE コマンドでオブジェクトのサイズを管理しなければなりません。また OBJECT GET BEST SIZE コマンドでテキストを含むオブジェクトのもっとも適切なサイズを知ることができます。同様に、Print object はページブレークを自動では生成しません。必要に応じて開発者が管理しなければなりません。

4D コマンドを使用してオブジェクトプロパティ (色やサイズなど) をオンザフライで変更できます。

オブジェクトが完全に印刷されるとコマンドは True を返します。そうでないばあい、言い換えればフレームワーク内のオブジェクトに割り当てられたデータをすべて印刷できな買った場合、コマンドは False を返します。特にリストボックスのすべての行を印刷できなかった場合、コマンドは False を返します。この場合 Print object コマンドを、それが True を返すまで繰り返し呼び出します。特別なメカニズムが自動で使用され、オブジェクトの内容が呼び出しごとに自動でスクロールされます。

Note: - 4D v12 の現在のバージョンでは、リストボックスタイプのオブジェクトのみがこのメカニズムを持っています (他のオブジェクトではコマンドは常に True を返します)。4D の将来のバージョンでこの機能は他の可変長オブジェクトに拡張されます。
- LISTBOX GET PRINT INFORMATION コマンドを使用して、処理中に印刷状況を知ることができます。詳細は 114 ページの " リストボックスの印刷 " を参照してください。

Print object コマンドは事前に OPEN PRINTING JOB で開かれた印刷ジョブのコンテキストでのみ使用できます。このコンテキストの外で呼び出された場合、コマンドはなにも行いません。同じ印刷ジョブ内で Print object コマンドを複数回呼び出すことができます。

Note: 階層リストおよび Web エリアを印刷することはできません。

▶ フォーム上の 10 個のオブジェクトを印刷する例 :

```
PRINT SETTINGS
If (OK=1)
  OPEN PRINTING JOB
  If (OK=1)
    OPEN PRINTING FORM("PrintForm")
    x:=100
    y:=50
    GET PRINTABLE AREA(hpaper;wpaper)
    For ($i;1;10)
      OBJECT GET BEST SIZE(*;"Obj"+String($i);bestwidth;bestheight)
      $end:=Print object(*;"Obj"+String($i))
      y:=y+bestheight+15
      If (y>hpaper)
        PAGE BREAK(>)
        y:=50
      End if
    End for
  End if
  CLOSE PRINTING JOB
```

```
End if
```

▶ リストボックスを完全に印刷する例 :

```
Repeat
   $end:=Print object(*;"mylistbox")
Until ($end)
```

参照 : OPEN PRINTING FORM

Windows での PDFCreator ドライバの統合

4D v12 で Windows における PDF 印刷のサポートが拡張されました。PDF 印刷を行うために、4D は PDFCreator ドライバに依存します。SET PRINT OPTION と GET PRINT OPTION はこのドライバを使用できるようにするために更新されました。

PDFCreator はオープンソースの無料のドライバで、AFPL (Aladdin Free Public License) により管理されています。このライセンスについての詳細は以下の Web ページを参照してください :
http://en.pdfforge.org/content/license

Note: Mac OS ではシステムがネイティブに PDF 印刷をサポートしています。

PDFCreator ドライバを使用できるようにするためには、適切なバージョンのドライバをダウンロードしてご利用の環境にインストールする必要があります。4D はデフォルトでインストールしません。4D v12 が対応している PDFCreator ドライバのバージョンは 0.9.9 です。このバージョンは以下のアドレスで入手できます :
http://sourceforge.net/projects/pdfcreator/files/PDFCreator/PDFCreator%200.9.9
ドライバをインストールするには管理者権限が必要です。

インストール時、デフォルトで "PDFCreator" という名前の新しい仮想プリンタがシステムにインストールされます。この名前は変更できます。

SET PRINT OPTION

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

引数	型		説明
option	倍長整数 | 文字	→	オプション番号または PDF オプションコード
value1	倍長整数 | 文字	→	オプションの値 1
value2	倍長整数 | 文字	→	オプションの値 2

SET PRINT OPTION コマンドが変更され、Windows で **PDFCreator** ドライバが利用できるようになりました。この新しい機能を使用するためには

お使いの 4D 環境にこのドライバをインストールしなければなりません (詳細は 151 ページの "Windows での PDFCreator ドライバの統合 " を参照してください)。

option 引数はテキスト型の値を受け入れるようになり、PDF オプションコードを渡せるようになりました。オプションコードは 2 つのパート OptionType と OptionName からなり、"OptionType:OptionName" のように合成されます。このコードについての説明は以下の通りです :

- OptionType: ネイティブの PDFCreator オプションまたは 4D PDF 管理オプションを指定します。2 つの値を渡せます :
  - **PDFOptions** = ネイティブオプション
  - **PDFInfo** = 内部オプション
- OptionName: 設定するオプションを指定します (OptionType の値に依存します)。
  - OptionType = **PDFOptions** のとき、OptionName には PDFCreator オプションのいずれかを渡せます。例えば **UseAutosave** オプションは自動バックアップに影響します。このオプションを変更するには "PDFOptions:UseAutosave" を option 引数に渡し、使用する値を value1 引数に渡します。完全な PDFCreator ネイティブオプションの説明については、PDFCreator ドライバのドキュメントを参照してください。
  - OptionType = **PDFInfo** のとき、OptionName には以下のいずれかのセレクタを渡せます :
    - **Reset print**: 特に無限ループから抜け出すための、内部的な街プロセスをリセットするために使用します。
    - **Reset standard options**: すべての PDFCreator オプションをデフォルト値に再設定するために使用します。印刷中の場合は、デフォルト設定はその印刷終了後に適用されます。この場合 value1 は使用されません。
    - **Start**: PDFCreator スプーラーを 開始または停止するために使用します。value1 に 0 を渡すと停止し、1 を渡すと開始します。
    - **Reset options**: セッションの開始から SET PRINT OPTION コマンドや **PDFOptions** を使用して変更されたすべてのオプションをリセットするために使用します。
    - **Version**: PDFCreator ドライバの現在のバージョンを読み取るために使用します。このオプションは GET PRINT OPTION コマンドでのみ使用できます。値は value1 引数に返されます。
    - **Last error**: PDFCreator ドライバから最後に返されたエラーを読み取るためん氏使用します。このセレクタは GET PRINT OPTION コマンドでのみ使用できます。エラー番号は value1 引数に返されます。

- **Print in progress**: 4D が PDFCreator による印刷を待っているか知るために使用します。このセレクタは GET PRINT OPTION コマンドでのみ使用できます。4D が PDFCreator を待っている間 value1 には 1 が、そうでなければ 0 が返されます。
- **Job count**: 印刷キュー中の待ちジョブ数を知るために使用します。ジョブ数は value1 に返されます。
- **Synchronous Mode**: 4D が送信した印刷リクエストと、PDFCreator ドライバとの間の同期モードを設定するために使用します。4D は印刷キュー内の印刷ジョブに関する現在の状況に関する情報を取得できないので、このオプションを使用して、PDFCreator ドライバの状態が空きのときにのみジョブを送信することで、ジョブの実行をよりよく制御できます。この場合、4D はドライバと同期されます。value1 に 0 を渡すと、4D は即座に印刷リクエストを送信し (デフォルト値) 、1 を渡すと 4D は実行中のジョブが終了するまでドライバを待ってから、他のジョブを送信します。

Note: それぞれの印刷の後、4D は自動で PDFCreator の前の設定を再ロードします。in order to avoid any interference with other programs using PDFCreator.

▶ 以下のメソッドは PDF ドライバを設定して、テーブル中のすべてのレコードを C:\test\Test_PDF_X (X は一連番号) ファイルに印刷します :

```
SET CURRENT PRINTER(PDFCreator Printer name)
    // PDFCreator がインストールした仮想プリンタを使用
SET PRINT OPTION ("PDFOptions:UseAutosave";1)
SET PRINT OPTION("PDFOptions:UseAutosaveDirectory";1)
SET PRINT OPTION("PDFOptions:AutosaveDirectory";"C:\\test\\")

ALL RECORDS([Table_1])
For ($i;1;Records in selection([Table_1]))
  SET PRINT OPTION("PDFOptions:AutosaveFilename";
                                        "Test_PDF_"+String($i))
  PRINT RECORD([Table_1];*)
  NEXT RECORD([Table_1])
End for
SET PRINT OPTION("PDFInfo:Reset standard options";0)
    // PDFCreator ドライバーオプションをリセット
```

GET PRINT OPTION

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

引数	型		説明
option	倍長整数 | 文字列	→	オプション番号、または PDF オプションコード

value1	倍長整数	←	オプションの値 1
value2	倍長整数	←	オプションの値 2

GET PRINT OPTION コマンドは option 引数に文字列を受け入れるようになりました。この変更によりコマンドを使用して PDFCreator ドライバオプションを管理できます。この新しい機能に関する詳細は SET PRINT OPTION コマンドの説明を参照してください。

SET CURRENT PRINTER

SET CURRENT PRINTER (printerName)

SET CURRENT PRINTER コマンドを使用して、PDFCreator ドライバがインストールした仮想プリンタを印刷先として指定できます。4D v12 は Windows での PDF ドキュメントの印刷を PDFCreator ドライバに依存します (151 ページの "Windows での PDFCreator ドライバの統合 " 参照)。

PDF ドキュメントを印刷するには、printerName 引数に PDFCreator ドライバがインストールした仮想プリンタの名前を渡します。

デフォルトで仮想プリンタの名前は "PDFCreator" です。しかしこの名称はドライバインストールの際に変更されているかもしれません。4D が自動で仮想プリンタ名を自動で探して使用できるようにするためには、カスタマイズされているかどうかにかかわらず、PDFCreator Printer name 定数を printerName に渡します。この定数は "Print options" テーマにあります。

ユーザインタフェース

このテーマに新しい OBJECT Get name と OBJECT Get pointer コマンドが追加されました。HIGHLIGHT TEXT と GET HIGHLIGHT コマンドがオブジェクトシンタックスを受け入れるようになりました。SCROLL LINES コマンドが OBJECT SET SCROLL POSITION に名称変更され、" オブジェクトプロパティ " テーマに移されました。

OBJECT Get name

OBJECT Get name{(selector)} → テキスト

引数	型		説明
selector	倍長整数	→	オブジェクトカテゴリ
戻り値	テキスト	←	オブジェクトの名前

新しい OBJECT Get name コマンドはフォームオブジェクトの名前を返します。

コマンドを使用して、selector 引数の値に基づき、2 タイプのオブジェクトを指定できます。この引数には以下の定数を渡せます ("Form objects" テーマ):

- Object current (省略時のデフォルト): このセレクタを渡すかselector引数を省略した場合、コマンドはコマンドを呼び出した (オブジェクトメソッドあるいはオブジェクトメソッドから呼ばれたサブメソッド) オブジェクトの名前を返します。この場合、コマンドはフォームオブジェクトのコンテキストから呼ばれなければなりません。そうでなければ空の文字列を返します。
- Object with focus: このセレクタを渡した場合、コマンドはフォーム上でフォーカスを持つオブジェクトの名前を返します。

▶ "bValidateForm" ボタンのオブジェクトメソッド :

```
$btnName:=OBJECT Get name(Object current)
```

このオブジェクトメソッド実行後、$btnName 変数には "bValidateForm" が格納されています。

参照 : OBJECT Get pointer

OBJECT Get pointer

OBJECT Get pointer {(selector{; objectName{; subformName}})} → ポインタ

引数	型		説明
selector	倍長整数	→	オブジェクトカテゴリ
objectName	テキスト	→	オブジェクト名
subformName	テキスト	→	サブフォームオブジェクト名
戻り値	ポインタ	←	オブジェクト変数へのポインタ

OBJECT Get pointer コマンドはフォームオブジェクトの変数へのポインタを返します。

このコマンドを使用して、selector 引数の値に基づき、異なるオブジェクトを指定できます。この引数には新しい "Form objects" テーマの定数を渡します :

- Object current (省略時のデフォルト): このセレクタを渡すかselector引数を省略した場合、コマンドはカレントオブジェクト (フォームメソッドを実行しているオブジェクト) に割り当てられた変数へのポインタを返します。

Note: これは、以前のバージョンの " ランゲージ " テーマにあった Self コマンドと全く同じ動作です。ただし Self コマンドは互換性のために保持されています。

- Object with focus: このセレクタを渡すと、コマンドはフォーム内でフォーカスを持つオブジェクトに割り当てられた変数へのポインタを返します。残り 2 つのオプション引数は渡されても無視されます。

Note: これは完全に Focus object コマンドと同じ動作です。Focus object コマンドは 4D v12 で廃止予定となりました。

- Object subform container: このセレクタを渡すと、コマンドはサブフォームコンテナにバインドされた変数へのポインタを返します。残り 2 つのオプション引数は渡されても無視されます。つまりこのセレクタはサブフォームとして使用されるフォームのコンテキストでのみ利用できます。詳細は 74 ページの " サブフォームの内容の更新 " を参照してください。
- Object named: このセレクタを渡す場合、2 番目の objectName 引数も渡さなければなりません。この場合、コマンドはこの引数に渡された名前を持つオブジェクトに割り当てられた変数へのポインタを返します。
  objectName がサブフォームに対応し、" 一覧サブフォーム " オプションがチェックされていると、コマンドはソーステーブルが指定されていればサブフォームのテーブルへのポインタを返します。そうでなければ Nil を返します。

オプションの subformName 引数を使用してカレントのコンテキスト (すなわち親フォーム) に属さない objectName オブジェクトへのポインタを取得できます。この引数を使用可能にするには、Object named セレクタを使用しなければなりません。
subformName 引数が渡されると、OBJECT Get pointer コマンドはまずカレントフォーム内で subformName という名称のサブフォームを探し、その中で objectName という名称のオブジェクトを探します。このオブジェクトが見つかると、このオブジェクトの変数へのポインタを返します。

▶ 同じ親フォーム上でサブフォームとして 2 回使用される "SF" フォームがあります。サブフォームオブジェクトにはそれぞれ "SF1" と "SF2" という名前が与えられます。"SF" フォームには CurrentValue という名称のオブジェクトがあります。親フォームのフォームメソッドの "On Load" フォームイベントで、SF1 の CurrentValue オブジェクトを "January" に、SF2 のそれを "February" に初期化します :

```
C_POINTER($Ptr)
$Ptr:=OBJECT Get pointer(Object named;"CurrentValue";"SF1")
```

```
$Ptr->:="January"
$Ptr:=OBJECT Get pointer(Object named;"CurrentValue";"SF2")
$Ptr->:="February"
```

参照 : OBJECT Get name

HIGHLIGHT TEXT

HIGHLIGHT TEXT({*; }object; startSel; endSel)

引数	型		説明
*	*	→	指定時 object はオブジェクト名 (文字列) 省略時 object はフィールドまたは変数
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) または フィールドまたは変数 (* 省略時)
startSel	数値	→	新しいテキスト選択開始位置
endSel	数値	→	新しいテキスト選択終了位置

HIGHLIGHT TEXT コマンドは " オブジェクト名 " タイプのシンタックスを受け付けるようになり、変数参照および名前でオブジェクトを指定できるようになりました。

オプションの * 引数を渡した場合 object 引数にはオブジェクト名 (文字列) を渡します。* 引数を渡さない場合は object にフィールドや変数を渡します。この場合文字列でなく、フォーム上に存在するフィールド / 変数参照を渡します。

GET HIGHLIGHT

GET HIGHLIGHT({*; }object; startSel; endSel)

引数	型		説明
*	*	→	指定時 object はオブジェクト名 (文字列) 省略時 object はフィールドまたは変数
object	フォームオブジェクト	→	オブジェクト名 (* 指定時) または フィールドまたは変数 (* 省略時)
startSel	数値	←	現在のテキスト選択開始位置
endSel	数値	←	現在のテキスト選択終了位置

GET HIGHLIGHT コマンドは " オブジェクト名 " タイプのシンタックスを受け付けるようになり、変数参照および名前でオブジェクトを指定できるようになりました。

オプションの * 引数を渡した場合 object 引数にはオブジェクト名 (文字列) を渡します。* 引数を渡さない場合は object にフィールドや変数を渡します。この場合文字列でなく、フォーム上に存在するフィールド / 変数参照を渡します。

- この新しいシンタックスにより、新しい OBJECT SET STYLED TEXT ATTRIBUTES コマンドを容易に使用できます :

```
GET HIGHLIGHT(*;"myText";$startsel,$endsel)
OBJECT SET STYLED TEXT ATTRIBUTE(*;"myText";$startsel,$endsel;
                    Attribute underline style; 1; Attribute bold style;1)
```

割込

4 つの新しいコマンドにより、4D コード中でアサーション (表明) をサポートします。これらの新しいツールは開発途中のアプリケーションのデバッグを容易にします。

Note: プログラミングエラーの正確な場所を知るために使用できる 2 つのシステム変数が追加されています。詳細は 239 ページの " 新しいシステム変数 " を参照してください。

ASSERT

ASSERT(boolExpression {; messageText})

引数	型		説明
boolExpression	ブール	→	ブール式
messageText	文字	→	エラーメッセージテキスト

新しい ASSERT コマンドを使用してメソッドコードにアサーションを置くことができます。

アサーションはコードに挿入された指示命令で、コード実行中の例外を検知するために使用します。ある時点において式の検証をしたときに true であれば正常であり、そうでなければ例外が発生したことになります。
アサーションはなにより、発生するはずのないケースを検知するために使用します。主にプログラミングバグを検知するために使用します。新しい SET ASSERT ENABLED コマンドを使用して、(例えばバージョンにより) アプリケーションのすべてのアサーションを全体として有効にしたり無効にしたりできます。
プログラミングにおけるアサーションについての詳細は、Wikipedia の関連情報をご覧ください : http://ja.wikipedia.org/wiki/ 表明

ASSERT コマンドは引数として渡されたブール式を評価します。式が true であればなにも起こりません。false の場合、コマンドはエラー -10518 を生成し、" アサーション違反 :" のあとにアサーションのテキストを表示します。ON ERR CALL コマンドを使用してインストールしたメソッドを使用して、例えばログファイルなどに記録を行うなどのために、エラーをとらえることができます。

コマンドはオプションで 2 番目の引数を受け入れます。これはブール式が False のときに表示されるテキストメッセージを変更します。

- レコードに対する処理を実行する前に、開発者はレコードが正しく読み / 書きモードでロードされたかを確認する必要があります :

```
READ WRITE([Table 1])
LOAD RECORD([Table 1])
ASSERT(Not(Locked([Table 1])))
  // レコードがロックされていると -10518 エラーが生成される
```

- アサーションはプロジェクトメソッドに渡された引数をテストして、異常な値を検知するために使用できます。以下の例題では、カスタム警告メッセージが表示されます :

```
  // $1 に渡された名前に基づき、クライアントの番号を返す
C_TEXT($1) // クライアントの名前
ASSERT($1 # "";" クライアント名が空です ")
  // このケースでは空の名前は異常な値です
  // アサーションが false の場合、以下がエラーダイアログに表示されます :
  // " アサーション違反 : クライアント名が空です "
```

参照 : Asserted, SET ASSERT ENABLED

Asserted

Asserted (boolExpression {; messageText})→ ブール

引数	型		説明
boolExpression	ブール	→	ブール式
messageText	文字	→	エラーメッセージテキスト
戻り値	ブール	←	boolExpression の評価結果

新しい Asserted コマンドは ASSERT コマンドと同様の処理を行います。1 つの違いは、このコマンドは boolExpression 引数の評価結果を戻り値として返すことです。このため、条件の評価としてアサーションを使用できます (例題参照)。アサーションの処理とこのコマンドの引数に関する詳細情報は ASSERT コマンドの説明を参照してください。

Asserted はブール式を引数として受け入れ、この式の評価結果を返します。アサーションが有効で式が false の場合 (SET ASSERT ENABLED コマンド参照)、ASSERT と同様エラー -10518 が生成されます。アサーションが無効にされていると、Asserted はエラー生成することなしに、渡された式の結果を返します。

▶ 式の評価部にアサーションを挿入する :

```
READ WRITE([Table 1])
LOAD RECORD([Table 1])
If (Asserted(Not(Locked( [Table 1]))))
   // このコードはレコードロック時にエラー -10518 を生成する
   ...
End if
```

参照 : ASSERT, SET ASSERT ENABLED

SET ASSERT ENABLED

SET ASSERT ENABLED(asserts{; *})

引数	型		説明
assertions	ブール	→	True = アサーションを有効にする False = アサーションを無効にする
*	*	→	省略時 = すべてのプロセスにコマンドを適用 (既存および後で作成されるものを含む) 指定時 = カレントプロセスのみにコマンドを適用

新しい SET ASSERT ENABLED コマンドを使用して、アプリケーションの 4D コードに挿入されたアサーションを無効にしたり、再度有効にしたりできます。アサーションに関する詳細は ASSERT コマンドの説明を参照してください。

デフォルトで、プログラムに追加されたアサーションは有効です。このコマンドは、ときにアサーションの評価に実行時間のコストが必要であったり、アプリケーションのエンドユーザからアサーションを隠ぺいしたいなどの理由で、アサーションを無効にしたいときに使用できます。典型的には、SET ASSERT ENABLED コマンドを On Startup データベースメソッドで使用して、アプリケーションがテストモードや運用モードかによって、アサーションを有効にしたり無効にしたりします。

デフォルトで SET ASSERT ENABLED コマンドはアプリケーションのすべてのプロセスに効果を及ぼします。コマンドの効果をカレントプロセスに限定するには、* 引数を渡します。

アサーションが無効にされていると、ASSERT コマンドに渡された式は評価されないことに留意してください。ASSERT を呼び出すコード行は動作においてもパフォーマンスにおいても、アプリケーションの処理に一切影響を及ぼしません。

▶ アサーションを無効にする :

```
SET ASSERT ENABLED(False)
ASSERT(TestMethod) // アサーションが無効なので TestMethod は呼び出されない
```

参照 : Get assert enabled

Get assert enabled

Get assert enabled → ブール

引数	型		説明
戻り値	ブール	←	True = アサーションは有効 False = アサーションは無効

新しい Get assert enabled コマンドは、データベースでアサーションが有効か無効かによって True または False を返します。アサーションについての詳細は ASSERT コマンドの説明を参照してください。

デフォルトでアサーションは有効ですが、SET ASSERT ENABLED コマンドを使用して無効にできます。

参照 : SET ASSERT ENABLED

エラーウィンドウ中で繰り返されるエラーを無視する

4D のエラーウィンドウに新しいショートカットが提供され、例えばループ中などで繰り返し発生するエラーを無視することができるようになりました。

これを行うには 4D エラーウィンドウ中で**続行**ボタンをクリックする際に、**Alt** (Windows) または **Option** (Mac OS) キーを押します。このアクションを行うと、同じメソッドの同じ行で同じエラーが再び発生しても、ウィンドウが表示されなくなります。この場合、ユーザが毎回**続行**ボタンを押したのと同様にコードの実行が続行されます。

ランゲージ

EXECUTE METHOD IN SUBFORM

EXECUTE METHOD IN SUBFORM(subformObject; methodName{; return{; param;...;paramN}})

引数	型		説明
subformObject	文字列	→	サブフォームオブジェクトの名称
methodName	文字列	→	実行するプロジェクトメソッドの名前
return	変数 | *	←	メソッドから返される値、または メソッドが値を戻さない場合 *
param	変数	→	メソッドに渡す引数

新しい EXECUTE METHOD IN SUBFORM コマンドを使用して、サブフォームオブジェクト subformObject のコンテキストでプロジェクトメソッド methodName を実行できます。

呼ばれるプロジェクトメソッドは任意の数の引数を param に受け取ることができ、また戻り値は return に返されます。メソッドが値を戻さない場合は、return に * を渡します。

データベースあるいはコマンドを実行するコンポーネントからアクセスが可能なプロジェクトメソッドを指定できます。実行コンテキストは呼び出されたメソッド内でも保持されます。つまりカレントフォームおよびカレントフォームイベントは指定されたまま引き継がれます。サブフォームがコンポーネント由来の場合、メソッドはコンポーネントに属していなければならず、また " コンポーネントとホストデータベースで共有する " プロパティがチェックされていなければなりません。

このコマンドは (subformObject オブジェクトを含む) 親フォームのフォームメソッドから実行しなくてはなりません。

4D v12 のサブフォームメカニズムについては 76 ページの " 上級フォーム間通信プログラミング " を参照してください。

- 親フォーム "Company" 中にサブフォーム "ContactDetail" が置かれています。ContactDetail フォームが設定されたサブフォームオブジェクトの名前は "ContactSubform" です。ここで company のフィールド値に基づき、サブフォーム内の特定の要素のアピアランスを変更したいとします (例えば [Company]City="New York" のときは "contactname" を赤に、[Company]City="San Diego" のときは青にするなど)。このメカニズムは **SetToColor** メソッドに実装されています。この結果を得るために、**SetToColor** メソッドを Company 親フォームの "On Load" フォームイベントのプロセスから直接呼び出すことはできません。なぜなら

"contactname" オブジェクトはカレントフォームではなく、"ContactSubform" サブフォームオブジェクト中に表示されているフォームに属しているからです。そのため正しく動作させるために、メソッドは EXECUTE METHOD IN SUBFORM コマンドを使用して実行されなければなりません。

```
Case of
  : (Form event= On Load)
     Case of
       : ([Company]City = "New York")
            $Color:=$Red
       : ([Company]City = "San Diego")
          $Color:=$Blue
       Else
          $Color:=$Black
     End case
     EXECUTE METHOD IN
SUBFORM("ContactSubform";"SetToColor";*;$Color)
End case
```

- コンポーネントとして使用される予定のデータベースを開発しています。このデータベースには共有プロジェクトフォーム (例として "Calender" と名付けます) が含まれています。またこのフォームにはダイナミック変数やカレンダを調整するための公開プロジェクトメソッド (**SetCalendarDate(varDate)**) が含まれています。
このメソッドが Calender フォームメソッドで直接使用される場合、開発者は直接このメソッドをフォームの "On Load" イベントで呼び出すことができます : **SetCalendarDate(Current date)**。しかしホストデータベースのコンテキストで、プロジェクトフォームに 2 つの "Calendar" サブフォーム "Cal1" および "Cal2" が配置されている状況を想像してください。Cal1 の日付を 2010/01/01、Cal2 の日付を 2010/05/05 に設定する必要があるとします。このケースでは、コンポーネントメソッド **SetCalendarDate** を直接呼び出すことはできません。なぜならコンポーネントメソッドはどちらのフォームに処理を適用したらよいかわからないからです。そこで、このメソッドを以下の方法で呼び出す必要があります :

```
EXECUTE METHOD IN SUBFORM("Cal1";
                                    "SetCalendarDate";*;!10/01/01!)
EXECUTE METHOD IN SUBFORM("Cal2";
                                    "SetCalendarDate";*;!10/05/05!)
```

▶ 上級例題 : 先と同じ状況で、ここでは汎用メソッドを作成してみます :

```
  // SetCalendarDate メソッドの内容
C_DATE($1)
C_TEXT($2)
Case of
  : (Count parameters=1)
      // 標準のメソッド実行 ( フォーム内で呼び出された場合 )
      // またはケース 2 からの再帰呼び出し

  : (Count parameters=2)
      // 外部呼出し、1 つの引数で再帰呼び出しを行う
    EXECUTE METHOD IN SUBFORM("$2";"SetCalendarDate";*;$1)
End case
```

リストボックス

4D v12 のリストボックステーマのコマンドにいくつかの更新が行われました :

- 新しい階層リストボックスを管理するための 4 つのコマンド (階層リストボックスに関する詳細は 103 ページの " リストボックス " を参照してください)。
- リストボックスの印刷をコントロールするための LISTBOX GET PRINT INFORMATION コマンド。
- 列のリサイズを制御するために使用する LISTBOX SET COLUMN WIDTH と LISTBOX Get column width コマンドの新しい引数 。
- タイプアヘッド入力などを最適化するため、コマンドに LISTBOX 接頭辞を追加。

LISTBOX SET HIERARCHY

LISTBOX SET HIERARCHY({*; }object; hierarchical{; hierarchy})

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)
hierarchical	ブール	→	True = 階層リストボックス False = 非階層リストボックス
hierarchy	配列	→	ポインタの配列

新しい LISTBOX SET HIERARCHY コマンドを使用して object と * で指定されたリストボックスを階層モードにするか非階層モードにするか設定できます。

Note: このコマンドは配列を表示するリストボックスに対してのみ機能します。セレクション表示モードのリストボックスに対してこのコマンドが適用された場合、なにも行いません。

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。

ブール型の hierarchical 引数を使用してリストボックスのモードを指定します :

- True を渡すと、リストボックスは階層モードで表示されます。
- False を渡すと、リストボックスは (階層モードではない) 標準配列モードで表示されます。

階層モードのリストボックスを渡すと、特定のプロパティは自動で制限されます。詳しくは 105 ページの " 変更されたプロパティ " を参照してください。

hierarchy 引数には、階層を構成するために使用されるリストボックスの配列を指定します。
この引数を省略すると :

- すでにリストボックスが階層モードの場合、コマンドはなにも行いません。
- リストボックスが現在非階層モードで、過去にも階層であると設定されたことがなければ、デフォルトで最初の配列が階層として使用されます。
- リストボックスが現在非階層モードで、以前に階層モードとして設定されたことがあれば、最後の時点の階層が再構築されます。

▶ aCountry、aRegion、そして aCity 配列をリストボックスの階層として定義する :

```
ARRAY POINTER($ArrHierarch;3)
$ArrHierarch{1}:=->aCountry ` 一番目のブレークレベル
$ArrHierarch{2}:=->aRegion ` 二番目のブレークレベル
$ArrHierarch{3}:=->aCity ` 三番目のブレークレベル
LISTBOX SET HIERARCHY(*;"mylistbox";True;$ArrHierarch)
```

参照 : LISTBOX GET HIERARCHY

LISTBOX GET HIERARCHY

LISTBOX GET HIERARCHY ({*; }object; hierarchical{; hierarchy})

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)
hierarchical	ブール	←	True = 階層リストボックス False = 非階層リストボックス
hierarchy	配列	←	ポインタの配列

新しい LISTBOX GET HIERARCHY コマンドを使用して、object と * で指定したリストボックスのプロパティが階層であるかどうかを知ることができます。

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。

ブール型の hierarchical 引数はリストボックスのモードが階層モードであるかないかを示します :

- 引数が True を返すと、リストボックスは階層モードです。
- 引数が False を返すと、リストボックスは (非階層の) 標準配列モードで表示されます。

リストボックスが階層モードであるとき、hierarchy 引数には階層を設定するために使用されるリストボックスの配列へのポインタが返されます。

Note: リストボックスが非階層モードの場合、コマンドは hierarchical 配列の第一要素に、リストボックスの先頭列の配列へのポインタを返します。

参照 : LISTBOX SET HIERARCHY

LISTBOX EXPAND

LISTBOX EXPAND ({*; }object{;recursive{; selector{; row; column} | {level}}})

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)
recursive	ブール	→	True = サブレベルを展開 False = サブレベルを展開しない
selector	倍長整数	→	展開するリストボックスのパーツ
row	倍長整数	→	展開するブレーク行の番号
column	倍長整数	→	展開するブレーク列の番号
level	倍長整数	→	展開するリストボックスレベルの番号

LISTBOX EXPAND コマンドは object と * で指定したリストボックスオブジェクトのブレーク行を展開するために使用します。

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。

リストボックスが階層モードとして設定されていない場合、コマンドはなにも行いません。階層リストボックスに関する詳細は 103 ページの " リストボックス " を参照してください。

オプションの recursive 引数を使用してリストボックスの階層サブレベルの展開を指定できます。True を渡すか省略すると、すべてのレベルおよびすべてのサブレベルが展開されます。False を渡すと指定された一番目のレベルのみが展開されます。

オプションの selector 引数を使用して、コマンドのスコープを指定できます。この引数には以下の定数のいずれかを渡すことができます :

- Listbox all (デフォルト値): すべてのサブレベルを展開する。
- Listbox selection: 選択されたサブレベルを展開する。
- Listbox break row: row と column 引数で指定したセルが属するサブレベルを展開する。これらの引数は標準モードのリストボックスの行および列番号で表されることに留意してください。階層状態の番号ではありません。このとき row と column 引数が省略されると、コマンドはなにも行いません。

- Listbox level: level 列に対応するすべてのブレーク行を展開します。この引数は標準モードのリストボックスの列番号で表されます。階層状態の番号ではありません。このとき level 引数が省略されると、コマンドはなにも行いません。

このコマンドはブレーク行を選択しません。

選択あるいはリストボックスがブレーク行を含んでいないか、すべてのブレーク行がすでに展開されている場合、コマンドはなにも行いません。

▶ 以下の例題では、コマンドのさまざまな利用方法を示します。以下の配列がリストボックスに表示されているものとします:

France	Brittany	Brest	120000
France	Brittany	Quimper	80000
France	Brittany	Rennes	200000
France	Normandy	Caen	220000
France	Normandy	Deauville	4000
France	Normandy	Cherbourg	41000
Belgium	Wallonia	Namur	111000
Belgium	Wallonia	Liège	200000
Belgium	Flanders	Antwerp	472000
Belgium	Flanders	Louvain	95000

```
// リストボックスのすべてのブレーク行とサブ行を展開する
LISTBOX EXPAND(*;"MyListbox")
```

V France		
V Brittany		
	Brest	120000
	Quimper	80000
	Rennes	200000
V Normandy		
	Caen	220000
	Deauville	4000
	Cherbourg	41000
V Belgium		
V Wallonia		
	Namur	111000
	Liège	200000
V Flanders		
	Antwerp	472000
	Louvain	95000

```
  // 選択された第一レベルのブレーク行を展開する
LISTBOX EXPAND (*;"MyListbox";False;Listbox selection)
  // "Belgium" 行が選択されている場合
```

> France
V Belgium
> Wallonia
> Flanders

```
  // Brittany ブレーク行を展開し、サブレベルは展開しない
LISTBOX EXPAND (*;"MyListbox";False;Listbox break row;1;2)
```

V France		
V Brittany		
	Brest	120000
	Quimper	80000
	Rennes	200000
> Normandy		
> Belgium		

```
  // 一番目の列 ( 国 ) のみを展開
LISTBOX EXPAND (*;"MyListbox";False;Listbox level;1)
```

V France
> Brittany
> Normandy
V Belgium
> Wallonia
> Flanders

参照 : LISTBOX COLLAPSE

LISTBOX COLLAPSE

LISTBOX COLLAPSE ({*; }object{;recursive{; selector{; row; column} | {level}}}})

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)
recursive	ブール	→	True = サブレベルを折りたたむ False = サブレベルを折りたたまない
selector	倍長整数	→	折り畳むリストボックスのパーツ
row	倍長整数	→	折り畳むブレーク行の番号
column	倍長整数	→	折り畳むブレーク列の番号
level	倍長整数	→	折り畳むリストボックスレベルの番号

LISTBOX COLLAPSE コマンドを使用して、object と * で指定したリストボックスのブレーク行を折りたたみます。

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。

リストボックスが階層モードとして設定されていない場合、コマンドはなにも行いません。階層リストボックスに関する詳細は 103 ページの " リストボックス " を参照してください。

オプションの recursive 引数を使用してリストボックスの階層サブレベルの折りたたみを指定できます。True を渡すか省略すると、すべてのレベルおよびすべてのサブレベルが折りたたまれます。False を渡すと一番目のレベルのみが折りたたまれます。

オプションの selector 引数を使用して、コマンドのスコープを指定できます。この引数には以下の定数のいずれかを渡すことができます :

- Listbox all (デフォルト値): すべてのサブレベルを折りたたむ。
- Listbox selection: 選択されたサブレベルを折りたたむ。
- Listbox break row: row と column 引数で指定したセルが属するサブレベルを折りたたむ。これらの引数は標準モードのリストボックスの行および列番号で表されることに留意してください。階層状態の番号ではありません。
- Listbox level: level列に対応するすべてのブレーク行を折りたたみます。この引数は標準モードのリストボックスの列番号で表されます。階層状態の番号ではありません。

選択あるいはリストボックスがブレーク行を含んでいないか、すべてのブレーク行がすでに折りたたまれている場合、コマンドはなにも行いません。

- この例はリストボックス中、選択されたブレーク行の第一レベルを折りたたみます :

LISTBOX COLLAPSE (*;"MyListbox";**False**;Listbox selection)

参照 : LISTBOX EXPAND

LISTBOX SELECT BREAK

LISTBOX SELECT BREAK({*; }object; row; column{; action})

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)
row	倍長整数	→	ブレーク行の番号
column	倍長整数	→	ブレーク列の番号
action	倍長整数	→	選択アクション

新しい LISTBOX SELECT BREAK コマンドを使用して、object と * で指定したリストボックス中でブレーク行を選択できます。リストボックスは階層モードで表示されていなければなりません。

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。

ブレーク行は階層を表現するために追加されますが、それは配列の既存の行には対応しません。選択するためにブレーク行を指定するには、row および column 引数に、対応する配列中の最初のオカレンスに対応する行と列の番号を渡さなければなりません。これらの値はユーザがブレーク行を選択したとき、LISTBOX GET CELL POSITION コマンドから返されます。この原則は 108 ページの " 選択や位置の管理 " で説明されています。

action 引数が渡されると、ブレーク行が既にリストボックス中に存在するときの実行する選択アクションを設定できます。値または "List box" テーマの以下の定数を渡すことができます :

- Replace listbox selection (0): 選択されたブレーク行が新しいブレーク行の選択となり、既存の選択は置きかえられます。コマンドはユーザがブレーク行をクリックしたのと同じ動作を行います。action 引数が渡されなかった場合、この動作がデフォルトとなります。
- Add to listbox selection (1): 選択されたブレーク行が既存の選択に追加されます。指定されたブレーク行がすでに既存の選択に含まれていれば、コマンドはなにも行いません。
- Remove from listbox selection (2): 選択されたブレーク行は既存の選択から取り除かれます。指定されたブレーク行が既存のセレクションに含まれない場合、コマンドはなにも行いません。

▶ リストボックスに表示されている以下の配列があります :

(T1)	(T2)	(T3)	(T4)
France	Brittany	Brest	120000
France	Brittany	Quimper	80000
France	Brittany	Rennes	200000
France	Normandy	Caen	220000
France	Normandy	Deauville	4000
France	Normandy	Cherbourg	41000
Belgium	Wallonia	Namur	111000
Belgium	Wallonia	Liège	200000
Belgium	Flanders	Antwerp	472000
Belgium	Flanders	Louvain	95000

"Normandy" ブレーク行を選択します :

```
$row:=Find in array(T2;"Normandy")
$column:=2
LISTBOX COLLAPSE (*;"MyListbox") // すべてのレベルを折りたたむ
LISTBOX SELECT BREAK(*;"MyListbox";$row;$column)
```

以下のような結果になります :

V France
> Brittany
> Normandy
> Belgium

参照 : LISTBOX GET CELL POSITION, LISTBOX EXPAND

LISTBOX GET PRINT INFORMATION

LISTBOX GET PRINT INFORMATION ({*; }object; selector; info)

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)
selector	倍長整数	→	取得する情報
info	倍長整数 | ブール	←	現在の値

新しい LISTBOX GET PRINT INFORMATION コマンドは、object と * で指定したリストボックスオブジェクトの印刷に関連する現在の情報を返します。このコマンドを使用してリストボックスの内容の印刷を制御します。

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。

このコマンドは Print object コマンドによるリストボックスの印刷のコンテキストで呼ばれなければなりません。このコンテキスト外では、意味のある値を返しません。

selector に知りたい情報を示す値を渡し、info には数値またはブール変数を渡します。selector には "List box" テーマの以下の定数を渡します :

セレクタ	info に返される値
Listbox Last printed row number (0)	info には印刷された最後の行の番号が返されます。これにより次に印刷される行の番号が分かります。 リストボックスが非表示行を含む場合や、OBJECT SET SCROLL POSITION コマンドが呼び出されていた場合、返される値は実際に印刷される行数より大きくなることがあります。 例えば 1, 18, 20 行目が印刷されると、info には 20 が返されます。
Listbox Printed rows(1)	info には、最後の Print object コマンドの呼び出しの間、実際に印刷された行数が返されます。この数値には階層リストボックスの場合に追加されたブレーク行も含まれます。 例えばリストボックスに 20 行あり、奇数行が隠されているとき、info には 10 が返されます。

Listbox Printing is over (2)	info にはリストボックス中の最後の (表示されている) 行が印刷されたかどうかを示すブール地が返されます。True = 行は印刷された ; そうでなければ False が返されます。
Listbox Printed height (3)	info には実際に印刷されたオブジェクトの高さがピクセルで返されます (ヘッダ、罫線等を含む)。印刷する行数がリストボックスの最大容量より少ない場合、高さは自動で減少させられます。

リストボックスの印刷に関する原則については、114 ページの " リストボックスの印刷 " を参照してください。

▶ すべての行が印刷されるまで印刷を行う :

```
OPEN PRINTING JOB
OPEN PRINTING FORM("SalesForm")
…
$Over:=False
Repeat
   PRINT OBJECT (*;"mylistbox")
   LISTBOX GET PRINT INFORMATION(*;"mylistbox";
Listbox Printing is over;$Over)
Until ($Over)
…
CLOSE PRINTING JOB
```

▶ 特定の行が隠されていて、リストボックスを最低 500 行印刷する :

```
$GlobalPrinted:=0
Repeat
   PRINT OBJECT (*;"mylistbox")
   LISTBOX GET PRINT INFORMATION(*;"mylistbox";
Listbox Printed rows;$Printed)
   $GlobalPrinted:=$GlobalPrinted+$Printed
   PAGE BREAK
Until ($GlobalPrinted>=500)
```

LISTBOX SET COLUMN WIDTH

LISTBOX SET COLUMN WIDTH({*; }object; width{; minWidth{; maxWidth}})

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字) 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)
width	倍長整数	→	列幅 (ピクセル単位)
minWidth	倍長整数	→	最小列幅 (ピクセル単位)
maxWidth	倍長整数	→	最大列幅 (ピクセル単位)

LISTBOX SET COLUMN WIDTH コマンドは 2 つのオプション引数を受け入れ、列幅の手動による変更の制限値を設定できるようになりました。

minWidth と maxWidth にそれぞれ、ピクセル単位で最小および最大列幅を渡すことができます。

列幅の変更を禁止するには minWidth と maxWidth に同じ値を渡します。

参照 : LISTBOX Get column width

LISTBOX Get column width

LISTBOX Get column width({*; }object{; minWidth{; maxWidth}}) → 倍長整数

引数	型		説明
*		→	指定時 object はオブジェクト名 (文字) 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)
largeurMin	倍長整数	←	最小列幅 (ピクセル単位)
largeurMax	倍長整数	←	最大列幅 (ピクセル単位)
戻り値	倍長整数	←	列幅 (ピクセル単位)

LISTBOX Get column width コマンドは minWidth と maxWidth 引数にリサイズの制限値を返すようになりました。

これらの制限値は LISTBOX SET COLUMN WIDTH コマンドを使用して設定できます。

列幅の最小値や最大値が設定されていない場合、対応する引数には 0 が返されます。

参照 : LISTBOX SET COLUMN WIDTH

挿入および削除コマンド

既存の LISTBOX INSERT COLUMN、LISTBOX INSERT COLUMN FORMULA、そして LISTBOX DELETE COLUMN コマンドは階層モードで表示されているリストボックスの一番目の列に適用されても、効果はありません。

Note: これらのコマンドは 4D v12 で名称変更されました (以下の段落参照)。

名称変更されたコマンド

標準化と調和のため、" リストボックス " テーマの既存のコマンドの名称が v12 で変更されました。コマンドには "LISTBOX" 接頭辞が追加されました。これらのコマンドの動作に変更はありません。

4D v12 の新しい名称	以前の名称
LISTBOX SET GRID COLOR	SET LISTBOX GRID COLOR
LISTBOX SET ROWS HEIGHT	SET LISTBOX ROWS HEIGHT
LISTBOX SET COLUMN WIDTH	SET LISTBOX COLUMN WIDTH
LISTBOX SET TABLE SOURCE	SET LISTBOX TABLE SOURCE
LISTBOX INSERT COLUMN FORMULA	INSERT LISTBOX COLUMN FORMULA
LISTBOX INSERT COLUMN	INSERT LISTBOX COLUMN
LISTBOX INSERT ROW	INSERT LISTBOX ROW
LISTBOX Get rows height	Get listbox rows height
LISTBOX Get information	Get listbox information
LISTBOX Get column width	Get listbox column width
LISTBOX Get number of columns	Get number of listbox columns
LISTBOX Get number of rows	Get number of listbox rows
LISTBOX GET CELL POSITION	GET LISTBOX CELL POSITION
LISTBOX GET TABLE SOURCE	GET LISTBOX TABLE SOURCE
LISTBOX GET ARRAYS	GET LISTBOX ARRAYS
LISTBOX SHOW GRID	SHOW LISTBOX GRID
LISTBOX MOVED COLUMN NUMBER	MOVED LISTBOX COLUMN NUMBER
LISTBOX MOVED ROW NUMBER	MOVED LISTBOX ROW NUMBER
LISTBOX SELECT ROW	SELECT LISTBOX ROW
LISTBOX DELETE COLUMN	DELETE LISTBOX COLUMN
LISTBOX DELETE ROW	DELETE LISTBOX ROW
LISTBOX SORT COLUMNS	SORT LISTBOX COLUMNS

ツール

4D v12 の " ツール " テーマのコマンドにいくつかの変更が行われました :

- UUID 識別子を生成するために使用できる新しいコマンド。
- ENCODE と DECODE コマンドは名称が BASE64 ENCODE と BASE64 DECODE に変更されま、テキストを返すことができるようになりました。

Generate UUID

Generate UUID → 文字列

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	文字列	←	新しい UUID テキスト (非整形 32 文字)

Generate UUID は 32 文字の UUID 識別子を整形されていない形式で返します。

UUID 識別子に関する詳細は、19 ページの "UUID のサポート " を参照してください。

▶ UUID を生成して変数に代入する :

```
C_TEXT(MyUUID)
MyUUID:=Generate UUID
```

SET ENVIRONMENT VARIABLE

SET ENVIRONMENT VARIABLE コマンドは PHP Execute コマンドとともに機能するようになりました。

BASE64 ENCODE

BASE64 ENCODE (blob{; encodedText})

引数	型		説明
blob	BLOB	→	Base64 フォーマットにエンコードする BLOB
		←	Base64 フォーマットにエンコードされた BLOB (encodedText 省略時)
encodedText	テキスト	←	Base64 フォーマットにエンコードされた BLOB

互換性の注意 : BASE64 ENCODE は以前のバージョンの 4D で ENCODE という名称でした。

BASE64 ENCODE コマンドは新しいオプションの引数 encodedText を受け入れるようになりました。この引数を渡すと、コマンド実行後にエンコードされた BLOB の内容を受け取ります。この場合 blob 引数の内容は変更されません。

この変更により、XML 仕様に沿う形でコマンドを使用できます (222 ページの "4D v12 で XML BLOB 使用時の留意点 " 参照)。

encodedText 引数を省略すると、コマンドは以前のバージョンと同様に動作します。

BASE64 DECODE

BASE64 DECODE ({encodedText; } blob)

引数	型		説明
encodedText	テキスト	→	Base64 フォーマットにエンコードされた BLOB を含むテキスト
blob	BLOB	→	Base64 フォーマットにエンコードされた BLOB (encodedText 省略時)
		←	デコードされた BLOB

互換性の注意 : BASE64 DECODE は以前のバージョンの 4D で DECODE という名称でした。

BASE64 DECODE コマンドは新しいオプションの引数 encodedText を受け入れるようになりました。この引数を渡すと、コマンドはその内容をデコードし、blob 引数に返します。この引数には Base64 フォーマットにエンコードされた BLOB が格納されていなければなりません。この場合、blob 引数に最初に含まれている内容は無視されます。

この変更により、XML 仕様により沿う形でコマンドを使用できます (222 ページの "4D v12 で XML BLOB 使用時の留意点 " 参照)。

encodedText 引数を省略すると、コマンドは以前のバージョンと同様に動作します。

▶ 以下の例題では画像を BLOB 経由で転送します :

```
C_BLOB($blobSource)
C_PICTURE($mypicture)
$mypicture:=[people]photo
PICTURE TO BLOB($mypicture;$blobSource;".JPG")
C_TEXT($texteBASE64)
BASE64 ENCODE($blobSource;$base64Text) // テキストにエンコード
// バイナリは $base64Text に文字として格納されている
```

C_TEXT($base64Text)
C_BLOB($blobTarget)
BASE64 DECODE($base64Text;$blobTarget) // テキストのデコード
// Base64 にエンコードされていたバイナリは $blobTarget に BLOB として格納される

PHP

この新しいテーマグループには、4D で PHP スクリプトを実行するためのコマンドが集められています (62 ページの "4D で PHP スクリプトを実行する " 参照)。

PHP Execute

PHP Execute (scriptPath{; functionName{; * | result{; param1{;…{; paramN}}}}}) → ブール

引数	型		説明
scriptPath	文字列	→	PHP スクリプトへのパスまたは "" で PHP 関数を実行
functionName	文字列	→	実行する PHP 関数
* | result	* | 変数 | フィールド	←	PHP 関数実行結果または結果を受け取らない場合 *
param1...N		→	PHP 関数の引数
戻り値	ブール	←	True = 正しく実行された False = 実行時にエラーがあった

PHP Execute コマンドは PHP スクリプトや関数を実行するために使用します。

scriptPath 引数には、実行する PHP スクリプトのパス名を渡します。ファイルがデータベースストラクチャと同階層に存在する場合、相対パス名を指定できます。そうでなければ完全パスです。パス名はシステムシンタックスあるいは POSIX シンタックスで表現できます。
標準の PHP 関数を直接実行したい場合は、scriptPath に空の文字列 ("") を渡します。関数名は二番目の引数に渡さなければなりません。

scriptPath スクリプト内の特定の関数を実行したい場合、functionName 引数に PHP 関数名を渡します。
この引数に空の文字列を渡したり省略した場合、スクリプト全体が実行されます。

Note: PHP の関数名は大文字小文字を区別します。括弧は使用せず、関数名のみを入力してください。

result 引数は PHP 関数の実行結果を受け取ります。以下のいずれかを渡せます :

- 結果を受け取る変数、配列、またはフィールド
- 関数が結果を返さないか、結果を受け取る必要がない場合、* 文字。

テキスト、倍長整数、実数、ブール、または日付型の変数や配列、フィールドを渡すことができます。また配列を除き、BLOB や時間タイプのフィールドも渡せます。4D はデータの変換と必要な調整を実行します (例えば実数値の場合、区切り文字はシステムに基づいて選択されます)。結果が result 引数の型に対応していない場合、4D は可能であればそれを変換します。例えばブールタイプの変数を result に渡し、数値を受け取った場合、結果が 0 でなければ result は True となります。他方、数値型 (倍長整数や実数) を result に渡し、True や False を受け取ると、result は 1 または 0 になります。

結果がテキスト型の変数やフィールドに返される場合、デフォルトで結果には PHP インタープリターから返されるヘッダを含みません。テキストとしてヘッダを受け取りたい場合は、PHP SET OPTION コマンドで PHP Raw result 定数を使用します。返される結果がテキストとして利用できない場合、4D はそれを Base64 にエンコードする点に留意してください。

Note: - 結果が BLOB に置かれるとき、ヘッダは常に保持されます。
- 結果が日付に置かれるとき、PHP スクリプトは DATE_ATOM フォーマット ("YYYY-MM-DDTHH:MM:SS") の日付を返さなければなりません。カスタムフォーマットやローカライズされたフォーマットは使用できません。

呼ばれた PHP 関数が引数を受け入れる場合、param1...N 引数を使用して 1 つ以上の値を渡すことができます。値はセミコロンで分けられなければなりません。文字、テキスト、ブール、実数、整数、倍長整数、日付、時間タイプの値を渡すことができます。ピクチャと BLOB は渡せません。配列へのポインタを使用すれば、配列の値を渡すことができます (例題参照)。コマンドにはポインタ、ピクチャ、および 2D 配列を除き、すべての型の配列を渡せます。

Note: - 日付を渡すと、4D が自動で UTC テキストに変換します。例えば 2010/04/23 は "2010-04-23T00:00:00Z" として送信されます。同様に時間も自動で HH:MM:SS フォーマットに変換します。例えば 1:05 a.m. は "01:05:00" として送信されます。
フォーマットを変更したい場合、カスタマイズした文字列を使用できます。特に 4D ランゲージには "datetime" 型が無いため、"2008-12-10T19:12:28+02:00" のような ISO フォーマットの文字列を使用します。

- 技術的な理由で、FastCGI プロトコル経由で渡す引数のサイズは 64KB を超えてはなりません。テキスト型の引数を使用する際にはこの制限を考慮にいれる必要があります。

4D 側でコマンドが正しく実行できると、言い換えれば実行環境の起動、スクリプトのオープン、そして PHP インタープリターとの通信に成功すると、コマンドから True が返されます。そうでない場合、ON ERR CALL でとらえることのでき、GET LAST ERROR STACK で解析できるエラーが生成されます。
さらにスクリプト自身が PHP エラーを生成するかもしれません。この場合、PHP GET FULL RESPONSE コマンドを使用してエラーの発生元を解析しなければなりません (例題 4 参照)。

Note: PHP を使用してエラー管理を設定できます。詳細は例えば以下のページを参照してください :
http://www.php.net/manual/en/errorfunc.configuration.php#ini.error-reporting.

環境変数の利用

SET ENVIRONMENT VARIABLE コマンドを使用してスクリプトが使用する環境変数を指定できます。警告 : LAUNCH EXTERNAL PROCESS や PHP Execute 呼出し後、環境変数の設定は消去されます。

特別な関数

4D は以下の特別な関数を提供します :

- quit_4d_php: PHP インタープリターとそのすべての子プロセスを終了するために使用します。スクリプトを実行中の子プロセスが存在すると、インタープリターは終了せず、PHP Execute コマンドは False を返します。
- relaunch_4d_php: PHP インタープリターを再起動するために使用します。
PHP Execute から最初のリクエストが送信されると、インタープリターが自動で再起動されることに留意してください。

例題

▶ 例題 1: "myPhpFile.php" スクリプトを関数指定なしで呼び出します。スクリプトは以下の通りです :

```
<?php
echo 'Current PHP version: ' . php_version();
?>
```

以下の 4D コードを実行すると :

```
C_TEXT($result)
C_BOOLEAN($isOK)
$isOK:=PHP Execute("C:\\php\\myPhpFile.php"; "";$result)
ALERT ($Result)
```

"Current PHP version: 5.3" と表示されます。

- 例題 2: "myNewScript.php" 内の **myPhpFunction** 関数を引数付きで呼び出します。スクリプトは以下の通りです :

```
<?php
// . . . PHP code. . .
function myPhpFunction($p1, $p2) {
  return $p1 . ' '. $p2;
}
// . . . PHP code.. . .
?>
```

関数を呼び出します :

```
C_TEXT($result)
C_TEXT($param1)
C_TEXT($param2)
C_BOOLEAN($isOk)
$param1 := "Hello"
$param2 := "4D world!"
$isOk:=PHP Execute(
"C:\\MyFolder\\myNewScript.php";"myPhpFunction";$result;$param1;$param2)
ALERT($result) // "Hello 4D world!" と表示される
```

- 例題 3: PHP インタープリターを終了します。

```
$ifOk:=PHP Execute ("";"quit_4d_php")
```

- 例題 4: エラー管理

```
  // エラー管理メソッドをインストール
phpCommError :="" // PHPErrorHandler 内で更新される
$T_saveErrorHandler :=Method called on error
ON ERR CALL("PHPErrorHandler")

  // スクリプトを実行
C_TEXT($T_result)
If(PHP Execute("C:\\MyScripts\\MiscInfos.php";"";$T_result))
      // エラーなし , $T_Result には結果が返される
Else
```

```
      // エラーが検知された , PHPErrorHandler メソッドにより管理
    If(phpCommError="")
      ... // PHP エラー , PHP GET FULL RESPONSE を使用する
    Else
      ALERT(phpCommError)
    End if
End if

// エラー管理メソッドをアンインストール
ON ERR CALL ($T_saveErrorHandler)
```

PHP_errHandler メソッドは以下の通りです :

```
phpCommError:=""
GET LAST ERROR STACK(arrCodes; arrComps; arrLabels)
For ($i;1;Size of array(arrCodes))
  phpCommError:=phpCommError+arrCodes{$i}+" "+arrComps{$i}+" "+
        arrLabels{$i}+Char(Carriage return)
End for
```

- 例題 5: 実行前に 4D で動的にスクリプトを作成します。

```
DOCUMENT TO BLOB("C:\\Scripts\\MyScript.php";$blobDoc)
If (OK=1)
  $strDoc:=BLOB to text($blobDoc;UTF8 text without length)

  $strPosition:=Position("function2Rename";$strDoc)

  $strDoc:=Insert string($strDoc;"_v2";Length("function2Rename")
                                    +$strPosition)

  TEXT TO BLOB($strDoc;$blobDoc;UTF8 text without length)
  BLOB TO DOCUMENT("C:\\Scripts\\MyScript.php";$blobDoc)
  If (OK#1)
    ALERT(" スクリプトの作成中にエラーが発生しました。")
  End if
End if
```

その後スクリプトを実行します :

```
$err:=PHP Execute("C:\\Scripts\\MyScript.php";
                                    "function2Rename_v2";*)
```

- 例題 6: 日付と時間タイプの値を直接受け取ります。スクリプトは以下の通りです :

```
<?php
// . . . code php. . .
echo date(DATE_ATOM, mktime(1, 2, 3, 4, 5, 2009));
// . . . code php. . .
?>
```

4D 側で日付を受け取ります :

```
C_DATE($phpResult_date)
$result :=PHP Execute("C:\php_scripts\ReturnDate.php";"";
$phpResult_date)
  //$phpResult_date は !2009/04/05 !

C_TIME($phpResult_time)
$result :=PHP Execute("C:\php_scripts\ReturnDate.php";"";
$phpResult_time)

  //$phpResult_time は ?01 :02 :03 ?
```

- 例題 7: 配列にデータを配分します :

```
ARRAY TEXT($arText ;0)
ARRAY LONGINT($arLong ;0)
$p1 :="," 
$p2 := "11,22,33,44,55"
$phpok :=PHP Execute("" ;"explode";$arText;$p1;$p2)
$phpok :=PHP Execute("" ;"explode";$arLong;$p1;$p2)

    // $arText には文字値 "11", "22", "33", が格納されます。
    // $arLong には数値 11, 22, 33, が格納されます。
```

- 例題 8: 配列を初期化します :

```
ARRAY TEXT($arText ;0)
$phpok :=PHP Execute("";"array_pad";$arText;->$arText;50;"undefined")
    // PHP で以下を実行 : $arrTest = array_pad($arrTest, 50, 'undefined');
    // $arText は 50 要素の "undefined" で埋められます。
```

- 例題 9: 配列を使用して引数を渡します :

```
ARRAY INTEGER($arInt;0)
$phpok :=PHP Execute("";"json_decode";$arInt;"[13,51,69,42,7]")
    // Execute in PHP: $arInt = json_decode('[13,51,69,42,7]');
    // 配列に初期値が設定されます
```

PHP SET OPTION

PHP SET OPTION (option; value{; *})

引数	型		説明
option	倍長整数	→	設定するオプション
value	文字 | ブール	→	オプションの新しい値
*	*	→	指定時 : 変更は次の呼び出し時にのみ適用

PHP SET OPTION コマンドを使用して、PHP Execute コマンド呼び出し前に、特定のオプションを設定することができます。このコマンドのスコープはカレントプロセスです。

option 引数には、変更するオプションを指定する "PHP" テーマの定数を渡します。value 引数には option の新しい値を渡します。option の説明は以下の通りです :

Option	説明	値
PHP Privileges	スクリプト実行に関連して指定されるユーザ権限定義	"User:Password" 形式の文字列 例 : "root:jd51254d"
PHP Raw result	結果がテキスト型のときに実行結果中に PHP から返される HTTP ヘッダに関する処理モードの定義 (結果が BLOB 型のときヘッダは常に保持されます)。	ブール False (デフォルト): 結果から HTTP ヘッダを取り除く True: HTTP ヘッダを保持する

デフォルトで PHP SET OPTION はプロセス内で後に続くすべての PHP Execute のオプションを設定します。次の呼び出しにのみ有効なオプションを設定するためには、アスタリスク (*) 引数を渡します。

▶ Admin アクセス権で "myAdminScript.php" スクリプトを実行します :

```
PHP SET OPTION(PHP Privileges; "admin:mypwd"; *)
   `* を渡すので、admin 権限は一回のみ使用される
C_TEXT($result)
C_BOOLEAN($isOK)
$isOK:=PHP Execute("myAdminScript.php";$result)
If($isOK)
   ALERT($result)
End if
```

参照 : PHP GET OPTION

PHP GET OPTION

PHP GET OPTION (option; value)

引数	型		説明
option	倍長整数	→	取得するオプション
value	文字 | ブール	←	オプションの現在値

PHP GET OPTION コマンドを使用して PHP スクリプトの実行に関連するオプションの現在値を取得できます。

option 引数には取得するオプションを指定する "PHP" テーマの定数を渡します。コマンドは value 引数にオプションの現在の値を返します。以下のいずれかの定数を渡すことができます :

- PHP Privileges: 現在のユーザアカウントを返します (パスワードは返されません)。
- PHP Raw result: HTTP ヘッダーの処理モードを返します。

これらのオプションに関する詳細は PHP SET OPTION コマンドの説明を参照してください。

▶ 現在のユーザアカウントを取得します :

```
C_TEXT($userAccount)
C_BLOB($isOK)
$isOK:=PHP GET OPTION (PHP Privileges;$userAccount)
If($isOK)
   ALERT($userAccount)
End if
```

参照 : PHP SET OPTION

PHP GET FULL RESPONSE

PHP GET FULL RESPONSE (stdout{; errLabels; errValues}{; httpHeaderFields{; httpHeaderValues}})

引数	型		説明
stdout	テキスト /BLOB 変数	←	stdout バッファーの内容
errLabels	テキスト配列	←	エラーのラベル
errValues	テキスト配列	←	エラーの値
httpHeaderFields	テキスト配列	←	HTTP ヘッダーの名前
httpHeaderValues	テキスト配列	←	HTTP ヘッダーの値

PHP GET FULL RESPONSE コマンドを使用して、PHP インタープリターから返されるレスポンスに関する追加の情報を取得できます。このコマンドは特にスクリプトの実行中にエラーが発生したときに有効です。

PHP スクリプトは stdout バッファーにデータを書き込むことがあります (echo, print 等)。このデータは stdout 変数で受け取ることができます。

同期される errLabels と errValues テキスト配列は、PHP スクリプトの実行がエラーの原因であるときに値が返されます。これらの配列には特に、エラーのもと、スクリプト、そして行などの情報が提供されます。これら 2 つの配列はともに使用します。errLabels を渡すときは合わせて errValues も渡さなければなりません。

4D と PHP 間の交換は FastCGI 経由で実行されるため、PHP インタープリターは、それが HTTP サーバから呼び出されたかのように機能し、したがって HTTP ヘッダを送信します。httpHeaderFields と httpHeaderValues 配列を使用してこれらのヘッダを取得できます。

オブジェクトプロパティ

このテーマのコマンドは 4D v12 で大きく変更されます :

- 新しいコマンドを使用して新しいプロパティにアクセスできます。
- より簡単に使用できるようにするため、既存のコマンドに接頭辞が付与されたり、名称変更されたりします。そして必要に応じ、コマンドでプロパティの読み書きが行えるようにコマンドが追加されます。

OBJECT SET STYLED TEXT ATTRIBUTES

OBJECT SET STYLED TEXT ATTRIBUTES({*; }object; startSel; endSel; attName; attValue{; attName2; attValue2;...;attNameN; attValueN})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
startSel	数値	→	新しいテキスト選択の開始位置
endSel	数値	→	新しいテキスト選択の終了位置
attName	倍長整数	→	設定する属性
attValue	文字 | 数値	→	新しい属性値

新しい OBJECT SET STYLED TEXT ATTRIBUTES コマンドを使用して、object で指定したフォームオブジェクト中の 1 つ以上のスタイル属性を変更できます。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照を渡します。

属性の定義は HTML スタイルのタグをテキストに挿入したり変更したりすることにより行われます (95 ページの " リッチテキスト管理プロパティ " 参照)。OBJECT SET STYLED TEXT ATTRIBUTES は、object がマルチスタイルプロパティを設定されていないテキストのフォームオブジェクトを指定している場合でも、すべてのケースでスタイルタグを挿入することに留意してください。

Note: タグを追加すると文字列のサイズが増加します。文字フィールドのスタイル属性を変更する場合、結果の文字長が最大フィールドサイズを超えないように注意してください。そうでなければデータは切り取られます。

startSel と endSel 引数は object 内でスタイルの変更を適用するテキストを選択するために使用します。startSel にはスタイルの変更を行う最初の文字の位置を、endSel に変更を行う最後の文字の位置を渡します。
startSel と endSel の値が同じ場合、または startSel が endSel より大きい場合、エラーが返されます。endSel が object 中の文字数より大きい場合、startSel からテキストの最後までが変更されます。
startSel と endSel の値はエリアに既に存在するスタイルタグを考慮しません。これらの引数は生のテキスト (スタイルタグがフィルタされたテキスト) をもとに評価されます。

attName と attValue には変更する属性に対応する名前と値を渡します。必要なだけ属性 / 値の組を渡すことができます。

attName 引数を指定するために、"Multistyle text attributes" テーマの定義済み定数を使用します。attValue 引数に渡す値は、attName 引数に基づきます :

attName	attValue
Attribute bold style(1)	0 = 選択部からボールド属性を取り除きます 1= 選択部にボールド属性を適用します
Attribute italic style(2)	0 = 選択部からイタリック属性を取り除きます 1= 選択部にイタリック属性を適用します

Attribute strikethrough style(3)	0 = 選択部から取り消し線属性を取り除きます 1= 選択部に取り消し線属性を適用します
Attribute underline style (4)	0 = 選択部から下線属性を取り除きます 1= 選択部に下線属性を適用します
Attribute font name(5)	フォント名 (文字)
Attribute text size(6)	ポイント数 (数値)
Attribute text color(7)	16 進値または HTML カラー名 (" カラー " 節参照)
Attribute background color(8)	(Windows のみ) 16 進値または HTML カラー名 (" カラー " 節参照)

カラー

attName に Attribute text color または Attribute background color 定数を渡した場合、attValue には HTML カラー名か 16 進のカラー値を文字で渡さなければなりません :

HTML カラー名	**16 進値**
Aqua	#00FFFF
Black	#000000
Blue	#0000FF
Fuchsia	#FF00FF
Gray	#808080
Green	#008000
Lime	#00FF00
Maroon	#800000
Navy	#000080
Olive	#808000
Purple	#800080
Red	#FF0000
Silver	#C0C0C0
Teal	#008080
White	#FFFFFF
Yellow	#FFFF00

attValue 引数にはテキスト、ブール、整数、実数、日付、または時間タイプを渡せます。無効な属性を渡すと 4D はエラーを返します。

▶ この例題ではテキストのサイズやカラーのほか、ボールドおよび下線属性を 2 番目から 5 番目の文字に設定します :

```
OBJECT SET STYLED TEXT ATTRIBUTES([MyTable]MyField;2;5;
Attribute font name;"Arial";Attribute text size; 10;
Attribute underline style; 1; Attribute bold style;
1;Attribute text color;"Blue")
```

OBJECT GET STYLED TEXT ATTRIBUTES

OBJECT GET STYLED TEXT ATTRIBUTES({*; }object; startSel; endSel; attName; attValue{; attName2; attValue2;...;attNameN; attValueN})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または テキストフィールドまたは変数 (* 省略時)
startSel	数値	→	テキスト選択の開始位置
endSel	数値	→	テキスト選択の終了位置
attName	倍長整数	→	取得する属性
attValue	変数	←	属性の現在の値

新しい OBJECT GET STYLED TEXT ATTRIBUTES コマンドは、object で指定したフォームオブジェクト中で選択されたテキストのスタイル属性を取得するために使用します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照を渡します。

startSel と endSel 引数を使用して、オブジェクト中でスタイル属性を取得するテキストを選択します。startSel には選択する最初の文字位置を、endSel には選択する文字の最後の位置を渡します。
startSel と endSel の値が等しい場合や、startSel が endSel よりも大きい場合、エラーが返されます。
startSel と endSel、エリア中に既に存在するスタイルタグを考慮に入れません。文字数のカウントは (テキストからスタイルタグを取り除いた) 生テキストを基に行います。

attName 引数に取得する属性の名前を渡します。これを行うには、"Multistyle text attributes" テーマの定数の一つを使用しなければなりません。詳細はルーチン OBJECT SET STYLED TEXT ATTRIBUTES、ページ 187を参照してください。attValue 引数には属性値を受け取る変数を渡さなければなりません。
必要なだけ属性 / 値の組を渡すことができます。

attName 属性の値が選択された文字列中全体で同じ場合、attValue にそれが返されます。値が異なる場合やオブジェクトが SPAN タグを含まない場合、以下の値が返されます :

attName	属性値が一致しない場合や SPAN タグが含まれない場合の attValue
Attribute font name	"" (空の文字列)
Attribute text size	2
Attribute text color	FFFFFFFF
Attribute background color	FFFFFFFF
Attribute bold style	2
Attribute italic style	2
Attribute underline style	2
Attribute strikethrough style	2

OBJECT Get plain text

OBJECT Get plain text({*; }object) → テキスト

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) またはテキストフィールドまたは変数 (* 省略時)
戻り値	テキスト	←	タグなしのテキスト

新しい OBJECT Get plain text コマンドは object 引数で指定したテキスト変数やフィールドからスタイルタグを取り除き、プレーンテキストを返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照を渡します。

▶ マルチスタイル付きのテキストフィールドからテキスト "very nice" を探します。値は以下のような形式で保存されています : "The weather is very nice today".

```
QUERY BY FORMULA([Comments];
                 OBJECT Get plain text([Comments]Weather)="@very nice@")
```

Note: このコンテキストでは、スタイルタグがテキストに含まれるため、以下のコードでは期待通りの結果が得られません :
QUERY([Comments];[Comments]Weather="@very nice@")

OBJECT SET STYLED TEXT

OBJECT SET STYLED TEXT({*; }object; newText{; startSel{; endSel}})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
newText	文字	→	挿入するテキスト
startSel	倍長整数	→	テキスト選択開始位置
endSel	倍長整数	→	テキスト選択終了位置

新しい OBJECT SET STYLED TEXT コマンドは object 引数で指定されたスタイル付きのフィールドや変数に、newText 引数で渡されたテキストを挿入します。このコマンドは object 引数のプレーンテキストにのみ適用され、含まれるスタイルタグは更新しません。このコマンドはスクリーンに表示されているスタイル付きテキストをプログラムで変更するために使用できます。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照を渡します。

newText には挿入するテキストを渡します。

オプションの startSel と endSel 引数を使用して、object 中のテキストを選択できます。startSel と endSel の値はプレーンテキストの選択に使用され、テキスト中のスタイルタグは無視されます。このコマンドの動作はオプションの startSel と endSel 引数に基づき変わります :

- startSel と endSel を省略すると、OBJECT SET STYLED TEXT は object のすべてのテキストを newText で置き換えます。
- startSel のみを渡した場合、OBJECT SET STYLED TEXT は newText テキストを object の startSel の位置に挿入します。
- startSel と endSel 両方渡した場合、OBJECT SET STYLED TEXT はこれらの引数で指定されたプレンテキストを newText テキストで置き換えます。

startSel と endSel の値が等しかったり、startSel が endSel より大きい場合、エラーが返されます。

▶ ユーザが選択したテキストを vtemp 変数の内容で置き換えます :

GET HIGHLIGHT([Products]Notes;vStart;vEnd)
OBJECT SET STYLED TEXT([Products]Notes;vtemp;vStart;vEnd)

参照 : OBJECT Get styled text, OBJECT Get plain text

OBJECT Get styled text

OBJECT Get styled text({*; }object{; startSel{; endSel}}) → テキスト

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) またはフィールドまたは変数 (* 省略時)
startSel	倍長整数	→	文字選択の開始位置
endSel	倍長整数	→	文字選択の終了位置
戻り値	テキスト	←	スタイルタグを含むテキスト

新しい OBJECT Get styled text コマンドは、object 引数で指定されたフィールドや変数中のスタイル付きテキストを返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照を渡します。

コマンドはテキストに割り当てられたスタイルタグとともにテキストを返します。これは例えばスタイルを保持したままテキストのコピーとペーストを行う場合に使用します。

オプションの startSel と endSel 引数を使用して、object 中のテキストを選択できます。startSel と endSel の値はプレーンテキストの選択に使用され、スタイルタグは無視されます。

- startSel と endSel を省略すると、OBJECT Get styled text は object 中すべてのテキストを返します。
- startSel と endSel を渡すと、OBJECT Get styled text はこれらの引数で選択された範囲のテキストを返します。

  startSel と endSel の値が等しい場合や startSel が endSel よりも大きい場合、エラーが返されます。

参照 : OBJECT SET STYLED TEXT, OBJECT Get plain text

OBJECT SET SCROLL POSITION

OBJECT SET SCROLL POSITION({*; }object{; vPosition{; hPosition}}{; *})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 省略時 : object は変数、フィールドまたはテーブル
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、フィールドまたはテーブル (* 省略時)
vPosition	倍長整数	→	表示する最初の行数、または ピクセル単位の縦スクロール (ピクチャ)
hPosition	倍長整数	→	表示する最初の列数、または ピクセル単位の横スクロール (ピクチャ)
*	*	→	スクロール後に行を (および hPosition 引数が渡されていれば列も) 先頭位置に表示する

Note: OBJECT SET SCROLL POSITION コマンドは以前 " ユーザインタフェース " テーマで、SCROLL LINES という 名称でした。($$$ STILL in User Interface Theme$$)

OBJECT SET SCROLL POSITION コマンドのスコープは 4D v12 で拡張されました。サブフォーム、リストフォーム、そしてリストボックスの縦スクロールに加え、このコマンドはリストボックスの横スクロールや、ピクチャの縦横スクロールにも使用できるようになりました。

- リストボックスの横スクロール : object がリストボックスをさす場合、hPosition 引数に列番号を渡すことができます。この場合、コマンドを実行すると、リストボックスが横スクロールされ、指定された列が表示されるようになります。列がすでに表示されている場合、コマンドはなにも行いません。
縦スクロールと同様、2 番目のオプションの * 引数を渡すと、コマンドにより表示される列は、(リストボックスが実際にスクロールされると) 先頭位置に置かれます。

Note: このコマンドは、たとえリストボックスが階層モードで表示されていたとしても、階層でではない標準モードのリストボックスの表現に基づく点に留意してください。すなわち OBJECT SET SCROLL POSITION の実行結果はリストボックスが標準モードで表示されているか階層モードで表示されているかで、結果が異なることがあります。

- ピクチャのスクロール : object がフォーム中で表示されるピクチャをさす場合、OBJECT SET SCROLL POSITION コマンドを使用して内容をスクロールできます。ピクチャは " トランケート (中央合わせなし)" フォーマット

で表示されていなければなりません。vPosition と hPosition にはそれぞれ、ピクチャに適用する縦および横のスクロール量を渡します。縦スクロールをしたくない場合、vPosition に 0 を渡します。値はローカルコンテキストの基点に対して相対的にピクセル単位で表現されなければなりません。

Note: フォーム中でスクロールバーが非表示にされていても、プログラムによるスクロールは可能です。

▶ この例題は、リストボックスが標準モードで表示されているか、階層モードで表示されているかでコマンドの動作がどのように異なるかについて示します :

```
OBJECT SET SCROLL POSITION(*;"mylistbox";4;2;*)
  // リストボックスの 4 行 2 列目を先頭位置に表示する
```

標準モードのリストボックスの場合 :

France	Brittany	Brest	120000	...
France	Brittany	Quimper	80000	...
France	Brittany	Rennes	200000	...
France	**Normandy**	Caen	220000	...
France	Normandy	Deauville	4000	...
France	Normandy	Cherbourg	41000	...
...	...	...	...	...

行と列が実際にスクロールされます :

Normandy	Caen	220000	...	...
Normandy	Deauville	4000	...	...
Normandy	Cherbourg	41000	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

他方、同じステートメントを階層モードで表示中のリストボックスに適用すると、行はスクロールされますが、列はスクロールされません。これは 2 番目の列が階層の一部だからです :

∨ **Normandy**		
	Caen	220000
	Deauville	4000
	Cherbourg	41000
	...	

参照 : OBJECT GET SCROLL POSITION, OBJECT SET FORMAT

OBJECT GET SCROLL POSITION

OBJECT GET SCROLL POSITION({*; }object; vPosition{; hPosition})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 省略時 : object は変数、フィールドまたはテーブル
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、フィールドまたはテーブル (* 省略時)
vPosition	倍長整数	←	表示されている最初の行数、または ピクセル単位の縦スクロール (ピクチャ)
hPosition	倍長整数	←	表示されている最初の列数、または ピクセル単位の横スクロール (ピクチャ)

新しい OBJECT GET SCROLL POSITION は、object で指定されたフォームオブジェクトの、スクロールバーの位置に関連する情報を vPosition と hPosition 引数に返します。

オプションの * 引数を渡すと、object 引数はサブフォーム、階層リスト、スクロールエリア、リストボックス、またはピクチャタイプのオブジェクト名 (文字) です。この引数を渡さないと、object はテーブル (リストフォームまたはサブフォームテーブル) 、変数 (階層リストの ListRef、ピクチャまたはリストボックス変数) またはフィールドです。

- object がリストタイプのオブジェクト (サブフォーム、リストフォーム、階層リスト、スクロールエリア、またはリストボックス) を指定する場合、vPosition には object 中で表示されている最初の行の番号が返されます。リストボックスの場合のみ、hPosition にはリストボックス中で一番左に完全に表示されている列の番号が返されます。他のタイプのオブジェクトの場合、この引数には 0 が返されます。
- object がピクチャ (変数またはフィールド) をさす場合、vPosition には縦移動、hPosition には横移動が返されます。これらの値はピクセル単位で表現され、ローカル座標システムのピクチャの基点を 0 とします。

参照 : OBJECT SET SCROLL POSITION

OBJECT SET FORMAT

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

Note: OBJECT SET FORMAT コマンドは以前のバージョンで SET FORMAT という名称でした。

サーモメーターやルーラーの場合、displayFormat 引数に新しいサブ引数を追加することができるようになりました :

min;max;unit;step;flags{;format{;display}}

- ルーラーの場合、サブ引数には以下の値を指定できます :
  - display = 0 (または省略時): 標準のルーラーを表示
  - display = 1: " ステッパー" モードを有効にする。詳細は 88 ページの " 新しいステッパーオブジェクト " を参照。
- サーモメーター (進捗インジケーター) の場合、このサブ引数は flags サブ引数の値が 128 のときにのみ有効です。この場合 :
  - display = 0 (または省略時): サーモメーターを " バーバーショップ " タイプの連続したアニメーションで表示
  - display = 1: "非同期進捗 " モードを有効にする。詳細は 90 ページの "非同期進捗インジケータ " を参照。

テーマの再構成

標準化のため、" オブジェクトプロパティ " テーマ内の複数のコマンドが v12 で名称変更されました (とくに "OBJECT" 接頭辞が機械的に追加されました)。これらのコマンドの機能は変更されていません。
さらに、プロパティの読み書き機能をシンメトリックに提供するため、新しいコマンドが追加されました。

以下変更されたコマンドを一覧します。後半には追加されたコマンドを示します :

4D v12 での新しい名称	以前の名称
OBJECT SET FONT	FONT
OBJECT SET FONT STYLE	FONT STYLE
OBJECT SET FONT SIZE	FONT SIZE
OBJECT SET COLOR	SET COLOR
OBJECT SET CHOICE LIST NAME	SET CHOICE LIST
OBJECT SET FILTER	SET FILTER
OBJECT SET FORMAT	SET FORMAT
OBJECT SET ENTERABLE	SET ENTERABLE
OBJECT SET VISIBLE	SET VISIBLE
OBJECT SET SCROLLBAR	SET SCROLLBAR VISIBLE
OBJECT MOVE	MOVE OBJECT

OBJECT SET ALIGNMENT	SET ALIGNMENT
OBJECT SET RBG COLORS	SET RGB COLORS
OBJECT Get alignment	Get alignment
OBJECT Get format	Get format
OBJECT GET COORDINATES	GET OBJECT RECT
OBJECT GET BEST SIZE	BEST OBJECT SIZE
OBJECT SET TITLE	BUTTON TEXT
4D v12 の新しいコマンド	
OBJECT Get font	
OBJECT Get title	
OBJECT Get font size	
OBJECT Get font style	
OBJECT Get choice list name	
OBJECT Get enterable	
OBJECT Get enabled	
OBJECT Get filter	
OBJECT GET RGB COLORS	
OBJECT GET SCROLL POSITION	
OBJECT GET SCROLLBAR	
OBJECT Get visible	
OBJECT DUPLICATE	
OBJECT SET ENABLED	

OBJECT Get choice list name

OBJECT Get choice list name ({*; }object) → 文字

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) またはフィールドまたは変数 (* 省略時)
戻り値	文字	←	(デザインモードで指定された) 選択リストの名前

新しい OBJECT Get choice list name コマンドは object で指定されたオブジェクトまたはオブジェクトグループに割り当てられた選択リストの名前を返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場

合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET CHOICE LIST NAME

OBJECT Get enabled

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

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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	ブール	←	True = オブジェクトは有効 ; そうでなければ false

新しい OBJECT Get enabled コマンドは object で指定されたオブジェクトまたはオブジェクトグループがフォーム中で有効なら True を、無効なら False を返します。

有効なオブジェクトはマウスクリックやキーボードショートカットに反応します。

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) です。この引数を渡さないと、object は変数です。この場合、文字ではなく変数への参照 (変数オブジェクトのみ) を渡します。

このコマンドは以下のタイプのオブジェクトに適用できます :

- ボタン、デフォルトボタン、3D ボタン、非表示ボタン、ハイライトボタン
- ラジオボタン、3D ラジオボタン、ピクチャボタン
- チェックボックス、3D チェックボックス
- ポップアップメニュー、ドロップダウンリスト、コンボボックス、メニュー / ドロップダウンリスト
- サーモメータ、ルーラ

参照 : OBJECT SET ENABLED

OBJECT Get enterable

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

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	ブール	←	True = 入力可 ; そうでなければ false

新しい OBJECT Get enterable コマンドは object で指定されたオブジェクトまたはオブジェクトグループが入力可属性を持つ場合に True を、そうでなければ False 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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET ENTERABLE

OBJECT Get filter

OBJECT Get filter ({*; }object) → 文字

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	文字	←	フィルター名

新しい OBJECT Get filter コマンドは object で指定されたオブジェクトまたはオブジェクトグループに割り当てられたフィルターの名前を返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET FILTER

OBJECT Get font

OBJECT Get font ({*; }object) → 文字

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	文字	←	フォント名

新しい OBJECT Get font コマンドは object で指定されたフォームオブジェクトで使用されている文字フォントの名前を渡します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET FONT

OBJECT Get font size

OBJECT Get font size ({*; }object) → 倍長整数

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	倍長整数	←	ポイント単位のフォントサイズ

新しい OBJECT Get font size コマンドは、object で指定されたフォームオブジェクトで使用されている文字フォントのサイズをポイント単位で返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET FONT SIZE

OBJECT Get font style

OBJECT Get font style ({*; }object) → 倍長整数

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	倍長整数	←	フォントスタイル

新しい OBJECT Get font style コマンドは、object で指定されたフォームオブジェクトで使用されている文字フォントの現在のスタイルを返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

返される値を "Font Styles" テーマの定義済み定数と比較することができます :

定数	型	値
Plain	倍長整数	0
Bold	倍長整数	1
Italic	倍長整数	2
Underline	倍長整数	4

参照 : OBJECT SET FONT STYLE

OBJECT GET RGB COLORS

OBJECT GET RGB COLORS({*; }object; foregroundColor{; backgroundColor{; altBackgrndColor}})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
foregroundColor	倍長整数	←	描画色の RGB カラー値
backgroundColor	倍長整数	←	背景色の RGB カラー値
altBackgrndColor	倍長整数	←	奇数行の色の RGB カラー値

新しい OBJECT GET RGB COLORS コマンドは、object で指定されたオブジェクトまたはオブジェクトグループの描画色や背景色を返します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

このコマンドをリストボックスタイプのオブジェクトに適用すると、altBackgrndColor 引数に奇数行のカラー値が返されることがあります。この場合、backgroundColor の値は偶数行にのみ使用されます。

foregroundColor、backgroundColor、そして altBackgrndColor 引数に返される RGB カラー値は 4 バイトの倍長整数値で 0x00RRGGBB のフォーマットになっています。このフォーマットに関する詳細は OBJECT SET RGB COLORS コマンドの詳細を参照してください。

参照 : OBJECT SET RGB COLORS

OBJECT GET SCROLLBAR

OBJECT GET SCROLLBAR({*; }object; horizontal; vertical)

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
horizontal	ブール	←	True= 表示 , False= 非表示
vertical	ブール	←	True= 表示 , False= 非表示

新しい OBJECT GET SCROLLBAR コマンドは、object で指定されたオブジェクトまたはオブジェクトグループの縦横スクロールバーの表示 / 非表示状態を知るために使用します。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

このコマンドは以下のオブジェクトに対して使用できます :

- リストボックス
- スクロールエリア

- 階層リスト
- サブフォーム

OBJECT Get title

OBJECT Get title ({*; }object) → 文字

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 省略時 : object は変数またはフィールド
objet	フォームオブジェクト	→	オブジェクト名 (* 指定時) または フィールドまたは変数 (* 省略時)
戻り値	文字	←	ボタンのタイトル

新しい OBJECT Get title コマンドは、object で指定されたフォームオブジェクトのタイトル (ラベル) を返します。このコマンドはテキストを表示する " ボタン " タイプのオブジェクト (ボタン、チェックボックス、ラジオボタンなど) にのみ使用できます。

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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET TITLE

OBJECT Get visible

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

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) または フィールドまたは変数 (* 省略時)
戻り値	ブール	←	True = オブジェクトは表示 ; そうでなければ False

新しい OBJECT Get visible コマンドは、object で指定されたオブジェクトまたはオブジェクトグループが 表示属性を持っていれば True を、そうでなければ False 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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。

参照 : OBJECT SET VISIBLE

OBJECT DUPLICATE

OBJECT DUPLICATE ({*; }object{;newName{;newVar{; boundTo{; moveH{; moveV{; resizeH{; resizeV}}}}}}}{; *})

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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) またはフィールドまたは変数 (* 省略時)
newName	テキスト	→	新しいオブジェクトの名前
newVar	ポインタ	→	新しいオブジェクトの変数へのポインタ
boundTo	テキスト	→	直前の入力順の入力可能オブジェクトまたはラジオボタングループ
moveH	倍長整数	→	新しいオブジェクトの横シフト (>0 = 右方向 , <0 = 左方向)
moveV	倍長整数	→	新しいオブジェクトの縦シフト (>0 = 下方向 , <0 = 上方向)
resizeH	倍長整数	→	新しいオブジェクトの横リサイズ
resizeV	倍長整数	→	新しいオブジェクトの縦リサイズ
*	*	→	指定時 = 絶対座標 省略時 = 相対座標

新しい OBJECT DUPLICATE コマンドを使用して、object 引数で指定したオブジェクトのコピーを作成できます。コピーは (アプリケーションモードで) 実行されているフォームのコンテキストで生成されます。デザインモードのソースフォームは変更されません。

デフォルトで、割り当てられているオブジェクトメソッドを含む、ソースオブジェクトに対しプロパティリストで設定されているすべてのオプションがコピーに適用されます (サイズ、リサイズオプション、カラー等)。

しかし以下の例外について留意してください :

- デフォルトボタン : フォーム中にデフォルトボタンは 1 つだけ存在できます。" デフォルトボタン " プロパティを持つボタンを複製すると、このプロパティはコピーに割り当てられ、ソースオブジェクトからは取り除かれます。
- キーボードショートカット : ソースオブジェクトに割り当てられているキーボードショートカットは複製されません。このプロパティはコピー先では未設定となります。
- オブジェクト名 : フォーム中でオブジェクト名はユニークでなければなりません。newName 引数を渡さない場合、ソースオブジェクト名が自動でインクリメントされ新しいオブジェクトで使用されます (後述参照)。

  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) です。この引数を渡さないと、object はフィールドまたは変数です。この場合、文字ではなくフィールドまたは変数への参照 (フィールドまたは変数オブジェクトのみ) を渡します。
  フィールドや変数参照を渡した場合で、フォーム中に同じ参照を使用するオブジェクトが複数ある場合、最初に見つかったオカレンスが使用されます。このような曖昧さを避けるために、ユニークであるオブジェクト名の使用をお勧めします。

  newName 引数にはオブジェクトのコピーに割り当てる名前を渡します。この名前はオブジェクト名の命名規則に沿い、フォーム中でユニークでなければなりません。有効でない、あるいは既に使用されている名前を渡すと、コマンドはなにも行わず、OK 変数に 0 が設定されます。
  この引数を省略するか空の文字列を渡すと、ソースオブジェクト名をインクリメントすることで、新しい名前が自動生成されます。例えば :

ソース名	コピーの名前
Button	Button 1
Button20	Button21
Button21	Button23 (Button22 がすでに使用されている場合)

  newVar には新しいオブジェクトに割り当てられる変数へのポインタを渡します。ルールとして、ソースオブジェクトと同じ型の変数をポイントしなければなりません。しかし特定の種類の " 型変換 " が可能です。汎用的なコードを書けるようにするために、コマンドは自動処理を提供します :

- 通常すべての " 入力可 " 変数は型変換が可能です。例えば日付や倍長整数を表示するオブジェクトを複製し、テキスト型の変数を割り当てることができます。互換のあるプロパティは保持されます。またテキストオブジェクトとピクチャオブジェクト間の型の変更も許容します。テキストオブジェクトを複製してブール変数やフィールドを割り当てると、自動でチェックボックスとして表示されます。
- 通常動的に変数からフィールド、あるいはその逆に変換が可能です。

  他方、グラフィックオブジェクト (ボタン、チェックボックス等) を他のタイプのコントロールに変換することはできません。

  変数の型がオブジェクトと互換でない場合、コマンドはなにも行わず、OK 変数に 0 が設定されます。
  この引数を省略すると、4D が変数を動的に作成します (239 ページの " ダイナミック変数 " 参照)。
  スタティックオブジェクト (線、四角、スタティックピクチャ等) を複製するとき、この引数は無視されます。他の引数を使用できるようにするには Nil ポインタ (->[]) を渡します。

  boundTo 引数は 2 つのケースで使用します :

- 入力順の変更 : この場合 boundTo には複製したオブジェクトの直前の入力順の入力可オブジェクト名を渡します。新しいオブジェクトをページ中最初の入力順にしたい場合は、新しい Object First in entry order 定数を渡します (242 ページの ""Form Events" テーマに新しいフォームイベントが 2 つ追加されました :" 参照)。
- ラジオボタングループへの関連付け : ラジオボタンをグループ化するために使用します。複製したオブジェクトがラジオボタンのとき、boundTo に新しいオブジェクトを関連付けたいグループのラジオボタンの名前を渡します。

  この引数を省略するか空の文字列を渡すと、新しいオブジェクトはフォームページ中の最後の入力可能オブジェクトとなります。ラジオボタンの場合、オブジェクトはソースボタンのグループに含められます。

  新しいオブジェクトは moveH、moveV、resizeH そして resizeV 引数を使用して移動およびリサイズできます。OBJECT MOVE コマンドのように、移動やリサイズの方向は moveH や moveV 引数に渡された値の符号で指定されます。

- 値が正数の場合、移動やリサイズはそれぞれ右および下方向に行われます。
- 値が負数の場合、移動やリサイズはそれぞれ左および上方向に行われます。

デフォルトで moveH、moveV、resizeH そして resizeV の値は、以前の場所からの相対位置で、オブジェクトの座標を変更します。この引数で絶対座標を指定したい場合、最後のオプションの * 引数を渡します。

これらの引数を省略すると、新しいオブジェクトはソースオブジェクトの上に重ねて配置されます。

このコマンドはフォームを表示するコンテキストで使用されなければなりません。コマンドは通常 On Load フォームイベントやユーザアクション (On Clicked イベント) で実行されます。

Note: On Load フォームイベントがソースオブジェクトに割り当てられているとき、コマンド実行時に複製されたオブジェクトでも生成されます。これにより例えば値の初期化などが行えます。

技術的および論理的な理由により、OBJECT DUPLICATE は特定のイベント内では呼び出すことができません。特に :

- オブジェクトメソッド内の On Load イベント
- On Unload イベント
- 印刷コンテキストに関連するイベント (On Header, On Printing Detail 等) オブジェクトを複数回印刷するには Print object コマンドを使用します。

サポートされていないコンテキストでコマンドが呼び出されると、オブジェクトは複製されずに、OK 変数に 0 が設定されます。コマンドが印刷のコンテキストで呼び出されると、エラー -10601 が生成されます。

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

▶ 既存の "OKButton" オブジェクトの上に新しいボタン "CancelButton" を作成し、vCancel 変数を割り当てます :

OBJECT DUPLICATE(*;"OKButton";"CancelButton";vCancel)

▶ 既存のラジオボタン "bRadio5" を基に新しいラジオボタン "bRadio6" を作成します。このボタンには変数 <>r6 が割り当てられ、"bRadio5" ボタンのと同じグループに入ります。位置は 20 ピクセル下に作成されます。:

OBJECT DUPLICATE(*;"bRadio5";"bRadio6";<>r6;"bRadio5";0;20)

OBJECT SET ENABLED

OBJECT SET ENABLED({*; }object; active)

引数	型		説明
*	*	→	指定時 : object はオブジェクト名 (文字) 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)
active	ブール	→	True = オブジェクトは有効 ; そうでなければ False

新しい OBJECT SET ENABLED コマンドは、object で指定されたカレントフォーム中のオブジェクトあるいはオブジェクトグループを、有効または無効にするために使用します。

有効なオブジェクトはマウスクリックやキーボードショートカットに反応します。

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) です。この引数を渡さないと、object は変数です。この場合、文字ではなく変数への参照 (変数オブジェクトのみ) を渡します。

このコマンドは以下のタイプのオブジェクトに適用できます :

- ボタン、デフォルトボタン、3D ボタン、非表示ボタン、ハイライトボタン
- ラジオボタン、3D ラジオボタン、ピクチャボタン
- チェックボックス、3D チェックボックス
- ポップアップメニュー、ドロップダウンリスト、コンボボックス、メニュー / ドロップダウンリスト
- サーモメーター、ルーラー

Note: このコマンドは**入力**と**キャンセル**アクションを除き、標準アクションが割り当てられているボタンには効果がありません (4D が必要に応じて状態を変更します)。

参照 : OBJECT Get enabled, ENABLE BUTTON, DISABLE BUTTON

ENABLE BUTTON, DISABLE BUTTON

ENABLE BUTTON と DISABLE BUTTON コマンドは 4D v12 で廃止予定とされました。これらは互換性のためにのみ保持されます。これらのコマンドのスコープは、指定された変数のすべてのインスタンスを含み、またカレントフォーム以外にも影響を及ぼすという点で、" オブジェクトプロパティ " テーマのコマンドのスコープに対応していません。

4D v12 では、ENABLE BUTTON と DISABLE BUTTON は新しい OBJECT SET ENABLED と OBJECT Get enabled コマンドに置きかえられます。

バックアップ

RESTORE

RESTORE{(archivePath{; destFolderPath})}

引数	型		説明
archivePath	テキスト	→	復元するアーカイブのパス
destFolderPath	テキスト	→	保存先フォルダのパス名

RESTORE コマンドは 2 つのオプションの引数を受け入れるようになり、復元処理を自動化し、OK と document システム変数を更新するようになりました。

archivePath 引数を使用して、復元するアーカイブファイルを指定できます。このパス名はシステムシンタックスで表現されなければなりません。絶対パスまたはデータベーストラクチャファイルからの相対パスを渡すことができます。
この場合、(destFolderPath 引数が省略されると)、標準の復元ダイアログボックスが、アーカイブが選択された状態で表示され、ユーザは保存先フォルダを指定できます。この場合、コマンドの動作は以前のバージョンと同じです。処理が完了すると、警告ダイアログボックスが表示され、復元された要素が格納されたフォルダが表示されます。

復元された要素の保存先フォルダのパス名を destFolderPath 引数に渡すこともできます。このパス名はシステムシンタックスで表現されなければなりません。絶対パスまたはデータベーストラクチャファイルからの相対パスを渡すことができます。この引数を渡すと、設定済みの復元ダイアログボックスが表示され、ユーザは 復元を実行あるいはキャンセルできます。処理が完了すると、ウィンドウは追加の情報を表示せずに閉じられます。

今後、RESTORE コマンドは OK と Document システム変数の値を更新します。復元が正しく実行されると、OK は 1 に設定され、Document には復元フォルダのパスが格納されます。
ユーザが復元ダイアログボックスをキャンセルしたり、復元を中断したり、エラーが発生すると、OK は 0 に設定され、Document には空の文字列が設定されます。エラーは ON ERR CALL コマンドで インストールされるエラー処理メソッドでとらえることができます。

Web サーバ

PROCESS HTML TAGS

PROCESS HTML TAGS コマンドは 4D v12 で最適化されました。このコマンドにより実行される処理は完全に Web サーバ設定とは切り離されます。特に BLOB 型引数を使用する場合、コマンドは Web サーバで設定された文字セットを考慮に入れません。互換性のため、BLOB で使用される文字セットは MacRoman になります。
より効率的にするために、テキスト型の引数を使用することはお勧めできません。この場合、処理は自動で Unicode で行われます。

SQL

3 つの新しい SQL コマンドを使用して、4D データベースのデータを書き出したり、SQL スクリプトを実行したりできます。

4D の SQL エンジンに関連する新しい機能は 279 ページの Chapter 6、"SQL" で説明しています。

SQL EXPORT DATABASE

SQL EXPORT DATABASE(folderPath{; numFiles{; fileLimitSize}})

引数	型		説明
folderPath	文字	→	書き出しフォルダーのパス名、または "" でフォルダー選択ダイアログボックスを表示
numFiles	数値	→	フォルダーごとの最大ファイル数
fileLimitSize	数値	→	書き出しファイルのサイズ制限 (KB)

新しい SQL EXPORT DATABASE コマンドは開かれたデータベースのすべてのテーブルのすべてのレコードを SQL フォーマットで書き出します。SQL ではこのグローバルな書き出しを " ダンプ " と呼びます。

Note:s このコマンドは直接あるいは ODBC 経由で開かれた外部接続では使用できません。

コマンドは他のデータベースにデータを読み込む際に必要となる SQL 分を含むテキストファイルを生成します。このファイルは直接新しい SQL EXECUTE SCRIPT コマンドで使用して、他の 4D データベースにデータを読み込むことができます。

書き出しファイルは、folderPath 引数で指定される保存先フォルダに作成される "SQLExport" フォルダに配置されます。"SQLExport" フォルダが指定した場所に既に存在する場合、コマンドは警告メッセージを表示することなしにそれを置き換えます。
引数に空の文字列を渡すと、4D は標準のフォルダを選択ダイアログボックスを表示します。デフォルトでダイアログボックスはセッションを開いたユーザのカレントフォルダーを表示します (Windows では " マイ ドキュメント "、Mac OS では " 書類 ")。

書き出されるテーブルごとに、コマンドは以下のアクションを行います :

- 保存先フォルダにテーブル名によるサブフォルダを作成する
- サブフォルダ内にテキストファイル "Export.sql" を作成します。このファイルは BOM 付き UTF-8 でエンコードされます。
  ファイルには書き出されたデータに対応する SQL の INSERT 命令が含まれます。
  フィールドの値はコロンで区切られます。テーブル中のフィールド数よりデータが少ない場合がありますが、この場合残りのフィールドは NULL として扱われます。
- テーブルに BLOB やピクチャ、テキストフィールド (レコードの外に保存されたテキスト) が含まれる場合、追加のサブフォルダ "BLOBS" が "Export.sql" ファイルと同階層に作成されます。このサブフォルダには必要なだけ "BlobsX" サブフォルダが作成されます。これらのサブフォルダにはテーブルレコードのすべての BLOB、ピクチャおよび外部テキストフィールドの内容が分離されたファイルとして格納されます。
  BLOB ファイルは "BlobXXXXX.BLOB" (XXXXX はアプリケーションが生成するユニーク番号) という名称がつけられます。ピクチャファイルは PICTXXXXX.ZZZZ (XXXXX はアプリケーションが生成するユニーク番号で、ZZZZ は拡張子)。可能であれば、ピクチャはオリジナルのネイティブフォーマットで書き出され、対応する拡張子を付けられます (.jpg, .png 等)。ネイティブフォーマットでの書き出しができない場合、ピクチャは 4D の内部フォーマットで書き出され、.4PCT 拡張子が付けられます。

numFiles 引数を渡すと、サブフォルダが numFiles 数以上の BLOB やピクチャを含まないよう、必要な応じて "BlobsX" サブフォルダを作成します。numFiles 引数が省略されると、コマンドはデフォルトでファイル数を 200 に制限します。0 を渡すと、それぞれのサブフォルダには少なくとも 1 つのファイルが置かれます。

fileLimitSize 引数を渡すと、ディスク上に作成されるそれぞれの "Export.sql" のサイズを (バイト単位で) 制限できます。作成された書き出しファイルのサイズが fileLimitSize で設定した制限に達すると、4D はレコードの書き込みを停止し、ファイルを閉じ、"ExportX.sql" (X は一連番号を表す) という新しいファイルを同階層に作成します。このメカニズムにおいては、"ExportX.sql" ファイルの実際のサイズは fileLimitSize を超える点に留意してください。なぜならサイズの制限を超えるときに書き出されているレコードが完全に書き出された後にファイルが閉じられるからです。設定可能な最小サイズは 100KB で最大サイズは 10MB です。

書き出しファイル中では、テーブル中のフィールドより値の数が少ない場合があります。この場合、空のフィールドは NULL とみなされます。フィールドに NULL 値を渡すこともできます。

書き出しが正しく実行されると OK 変数に 1 が。そうでなければ 0 が設定されます。

参照 : SQL EXPORT SELECTION

SQL EXPORT SELECTION

SQL EXPORT SELECTION(aTable; folderPath{; numFiles{; fileLimitSize}})

引数	型		説明
aTable	テーブル	→	セレクションを書き出すテーブル
folderPath	文字	→	書き出しフォルダーのパス名、または "" でフォルダー選択ダイアログボックスを表示
numFiles	数値	→	フォルダーごとの最大ファイル数
fileLimitSize	数値	→	Export.sql ファイルの最大サイズ (KB)

新しい SQL EXPORT SELECTION コマンドは、aTable 引数で指定した 4D テーブルのカレントセレクションを SQL フォーマットで書き出します。

Note:s このコマンドは直接あるいは ODBC 経由で開かれた外部接続では使用できません。

このコマンドは SQL EXPORT DATABASE コマンドとほぼ同様のものです。これら 2 つのコマンドの主な違いは、SQL EXPORT SELECTION は aTable のカレントセレクションのみを書き出すのに対し、SQL EXPORT DATABASE はデータベース全体を書き出すことです。また SQL EXPORT DATABASE コマンドと異なり、SQL EXPORT SELECTION コマンドはエクスターナル SQL データベースでは動作しません。このコマンドはメインデータベースでのみ動作します。

これらのコマンドの動作と引数の説明については SQL EXPORT DATABASE コマンドを参照してください。

カレントセレクションが空の場合、コマンドはなにも行いません。この場合、保存先フォルダは空にされないことに留意してください。
書き出しが正しく実行されると OK 変数に 1 が、そうでなければ 0 が設定されます。

参照 : SQL EXPORT DATABASE

SQL EXECUTE SCRIPT

SQL EXECUTE SCRIPT(scriptPath; errorAction{; attribName1; attribValue1;...; attribNameN; attribValueN})

引数	型		説明
scriptPath	文字	→	実行する SQL スクリプトが書かれたファイルの完全パス名
errorAction	倍長整数	→	スクリプト実行中にエラーが発生した場合のアクション
attribName1...N	文字	→	使用する属性の名前
attribValue1...N	String	→	属性の値

新しい SQL EXECUTE SCRIPT コマンドを使用して、scriptPath で指定されたスクリプトファイルに書かれた一連の SQL ステートメントを実行できます。SQL EXECUTE SCRIPT はローカルマシンでのみ実行できます (ローカル 4D または 4D Server のストアドプロシージャ)。

Note: このコマンドは直接あるいは ODBC 経由で開かれた外部接続では利用できません。

scriptPath 引数には実行する SQL 文が書かれたテキストファイルの完全パス名を渡します。パス名は現在のシステムのシンタックスを使用して表現されなければなりません。scriptPath に空の文字列 ("") を渡すと、標準のファイルを開くダイアログボックスが表示され、実行するファイルをユーザが選択できます。

Note: 新しい SQL EXPORT DATABASE と SQL EXPORT SELECTION コマンドは自動でこのスクリプトファイルを生成します。

errorAction 引数を使用して、スクリプト実行中にエラーが発生した場合の動作を設定できます。"SQL" テーマ中の以下の 3 つの定数を使用できます :

定数	型	値
SQL On error abort	倍長整数	1
SQL On error confirm	倍長整数	2
SQL On error continue	倍長整数	3

- SQL On error abort: エラーが発生したら、4D はただちにスクリプト実行を停止します。
- SQL On error confirm: エラーが発生したら、4D はエラーを説明するダイアログを表示し、ユーザはスクリプトを実行するか中断するか決定できます。
- SQL On error continue: エラーが発生したら、4D はそれを無視し、スクリプトを実行し続けます。

attribName と attribValue 引数は組で渡さなければなりません。これらの引数はスクリプト実行のための特定の属性として使用されることを意図しています。現在のバージョンの 4D では、一つの属性を attribName に渡せます。"SQL" テーマの以下の定数を使用できます :

定数	型	値
SQL Use Access Rights	文字	SQL_Use_Access_Rights

- SQL Use Access Rights: スクリプトの SQL コマンド実行中に適用されるアクセス権を制限するために使用します。この属性を使用する場合、attribValue には 0 または 1 を渡さなければなりません :
  - attribValue = 1: 4D はカレントの 4D ユーザのアクセス権を使用する。
  - attribValue = 0 (または属性が指定されなかった場合): 4Dはアクセスを制限せず、Designer の権限が使用される。

(SET DATABASE PARAMETER のセレクター 28 や 45 で) 4D ログファイルが有効にされていると、SQL コマンドが実行されるごとに、以下の情報が書き込まれます :

- SQL コマンドのタイプ
- コマンドの影響を受けるレコード数
- コマンドの実行時間

- エラーを検知するごとに :
  - エラーコード
  - 可能であればエラーテキスト

スクリプトが (エラーなく) 正しく実行されると、OK 変数に 1 が設定されます。
エラーが発生した場合、OK システム変数に 0 が設定されるか、または errorAction 引数に設定に基づき :

- errorAction が SQL On error abort (値 1) の場合、OK は 0 に設定されます。
- errorAction が SQL On error confirm (値 2) の場合、ユーザが処理を中断すると OK に 0 が、続行を選択すると 1 が設定されます。
- errorAction が SQL On error continue (値 3) の場合、OK は常に 1 が設定されます。

Note: 大量のデータ読み込みなど、メモリを消費するアクションを実行するためにこのコマンドを使用する場合、一時的に SQL オプションを無効にするために新しい SQL の ALTER DATABASE コマンドの実行を検討できます。

SVG

新しい "SVG" テーマには、4D v12 のすべての SVG コマンドがまとめられています。以前のバージョンの 4D では、これらのコマンドは "XML ユーティリティ " テーマにありました。

4 つの新しい SVG コマンドが 4D v12 に加えられました : SVG SET ATTRIBUTE、SVG GET ATTRIBUTE、SVG Find element ID by rect、SVG SHOW ELEMENT。

SVG 描画エンジンについて

4D v12 で SVG 描画エンジンが更新されました。以下説明する新しいコマンドに加え、"pattern" や "CoreText" など多くの新しい要素や要素の属性がサポートされるようになりました。4D の **SVG component** ではこの新しい機能を利用しています。

SVG SET ATTRIBUTE

SVG SET ATTRIBUTE({*; }pictureObject; element_ID; attribName; attribValue{; attribName2; attribValue2; ...; attribNameN; attribValueN})

引数	型		説明
*	*	→	指定時 : pictureObject はオブジェクト名 (文字) 省略時 : pictureObject は変数
pictureObject	ピクチャ	→	オブジェクト名 (* 指定時) または変数 またはフィールド (* 省略時)
element_ID	文字	→	1 つ以上の属性を設定する要素の ID
attribName	文字	→	指定する属性
attribValue	文字 | 値	→	属性の新しい値

新しい SVG SET ATTRIBUTE コマンドは、表示されているイメージの SVG 描画ツリー中で、既存の属性の値を更新するために使用します。

このコマンドにより行われる更新はピクチャの表示に適用されます。更新は XML に格納されず、ピクチャがプログラムで消去されたり、フォームが閉じられるときに、失われます。

オプションの * 引数を渡すと、pictureObject 引数はオブジェクト名 (文字) です。この場合、コマンドはオブジェクトにアタッチされたイメージのパラメタに適用されます (パラメタおよびオブジェクトの表示されたイメージは、少なくとも一回 SVG SET ATTRIBUTE が呼び出されると作成されることに留意してください)。
* 引数を渡さないと、pictureObject 引数は変数またはフィールドです。従って文字ではなく変数参照 (変数オブジェクトのみ) またはフィールド参照を渡します。この場合コマンドは、その変数を使用するすべてのオブジェクトに表示されたイメージに適用されます (最初に表示されたイメージではなく)。

attribName と attribValue 引数にはそれぞれ、書き込む属性と値を (変数、フィールド、またはリテラル値で) 渡します。必要なだけ属性 / 値に組を渡せます。

SVG SET ATTRIBUTE コマンドは、'fill'、'opacity'、'font-family' などほとんどの SVG 属性を (追加や削除でなく) 変更するために使用します。SVG 属性の完全な説明は、インターネット上のドキュメント (例 : http://www.w3.org/TR/SVG11/attindex.html) などを参照してください。表示されるイメージは即座に更新されます。継承されるスタイルの場合、更新は子要素にも適用されます。

技術的な理由で、特定の要素や特定の属性は更新できません。以下の表は更新可能および不可能な要素、さらには変更不可能な属性のリストです :

属性を更新可能な要素	
svg	制限 : - "width" と "height" は変更できません。(1) - "viewBox" は、"width" と "height" がオリジナルのドキュメントで指定されているときのみ変更できます。
g	
defs	
use	
filter	制限 : 子要素 fe_xxx へ変更できません。
circle	
ellipse	
line	
polyline	
polygon	
path	
rect	
text	"4d-text" 属性を使用して "text"、"tspan"、および "textArea" 要素のテキストを更新します (例題参照)。
tspan	
textArea	
Image	
属性を変更できない要素	
linearGradient	このグループは参照可能または参照可能な要素に含有可能なすべての要素を示します。つまりこれは、例えばグラデーションの属性を再定義することはできないということを意味します (しかし使用するグラデーションを変更することはできます)。同様に黒のマーカーを赤のマーカーに変更するには、SVG ドキュメント内で両マーカー (1 つは黒でもう 1 つは赤) を定義し、どちらかを選択する必要があることを示します。また例えば親要素がシンボルまたはマーカー要素であるとき、四角の色を変えることもできません。
radialGradient	
Stop	
solidColor	
marker	
symbol	
clipPath	
fe で始まるフィルターと要素	
style	
pattern	
変更できない属性	
id や xml:id	
lang や xml:lang	
class や xml:class	

width	'svg' 要素のの属性のみ (1)
height	

(1) これらの属性は、それらが結果のイメージを定義および構築するため変更できません。svg 要素の width および height 属性は 4D 中で初期のサイズを決定し、ピクチャ作成後このサイズは一定でなければなりません (しかしながら 4D の TRANSFORM PICTURE コマンドを使用して結果のピクチャのサイズを変更できます。

サポートされていない要素の属性やその子要素を更新しようとすると、コマンドはなにも行わず、エラーが生成されます。

- テキスト型の要素の内容を更新する :

SVG SET ATTRIBUTE (*;picture_object_name;text_element_ID;"4d-text";"This is a text")

Note: 衝突の恐れなしに CSS スタイルシート内で属性を使用するため、名前空間がありません。

SVG GET ATTRIBUTE

SVG GET ATTRIBUTE({*; }pictureObject; element_ID; attribName; attribValue)

引数	型		説明
*	*	→	指定時 : pictureObject はオブジェクト名 (文字) 省略時 : pictureObject は変数
pictureObject	ピクチャ	→	オブジェクト名 (* 指定時) または変数 (* 省略時)
element_ID	文字	→	属性値を取得する要素の ID
attribName	文字	→	取得する属性
attribValue	文字 | 値	←	現在の属性値

新しい SVG GET ATTRIBUTE コマンドを使用してオブジェクトまたは SVG ピクチャの attribName 属性の現在値を取得できます。

オプションの * 引数を渡すと、pictureObject 引数はオブジェクト名 (文字) です。この場合、コマンドはオブジェクトにアタッチされた描画イメージの属性値を返します。この値は例えば SVG SET ATTRIBUTE で変更されているかもしれません。
* 引数を渡さないと、pictureObject 引数は変数です。従って文字ではなく変数参照 (変数オブジェクトのみ) を渡します。この場合コマンドは、最初に描画されたイメージの属性値を返します (変数のデーターソースに対応)。

Note: この原則は既存の SVG Find element ID by coordinates コマンドにも適用されます。

SVG 属性に関する詳細は、SVG SET ATTRIBUTE コマンドの説明を参照してください。以下は予約済みまたはアニメーションに関連する 4D の属性です :

属性	アクセス	コメント
4D-text	読み / 書き	テキストノードの内容を置き換え / 読み込みます。'text'、'tspan'、および 'textArea' 要素で利用できます。
4D-bringToFront	読み込み	'true' の場合、ノードを兄弟ノードの前面に移動します。SVG SET ATTRIBUTE コマンドとともにのみ使用できます。
4D-isOfClass-{IDENT [[S|COMMA] IDENT]*}	読み込み	ノードの継承クラス属性がすべてのクラス名を含む場合に 'true' を返します。そうでなければ 'false' を返します。 例えば "4D-isOfClass-land" に対してノードの継承されたクラスが "land department01" の場合、true を返します。

SVG Find element ID by rect

SVG Find element ID by rect ({*; }pictureObject; x; y; width; height; arrIDs) → ブール

引数	型		説明
*	*	→	指定時 : pictureObject はオブジェクト名 (文字) 省略時 : pictureObject は変数
pictureObject	ピクチャ	→	オブジェクト名 (* 指定時) またはフィールドや変数 (* 省略時)
x	倍長整数	→	選択領域の左上の横座標
y	倍長整数	→	選択領域の左上の縦座標
width	倍長整数	→	選択領域の幅
height	倍長整数	→	選択領域の高さ
arrIDs	文字配列	←	バインドされた四角が選択領域に交差する要素の ID
戻り値	ブール	←	True = 最低 1 つの要素が見つかった

新しい SVG Find element ID by rect コマンドは、バインドされた四角が選択領域に交差する XML 要素の ID ("id" または "xml:id" 属性) をテキストまたは文字配列の arrIDs 配列に返します。選択領域は x および y 引数で指定されます。

最低 1 つの要素が見つかると、言い換えれば arrIDs 配列が空でなければ、コマンドは True を返します。そうでなければ False を返します。

このコマンドは特にインタラクティブなグラフィックインタフェースで使用されます。

オプションの * 引数を渡すと、pictureObject 引数はオブジェクト名 (文字) です。この引数を渡さないと、pictureObject 引数はフィールドまたは変数です。この場合文字列ではなくフィールドや変数の参照を渡します。

x と y 引数に渡される座標はピクチャの左上座標 (0, 0) からピクセル単位で表現されます。MouseX と MouseY システム変数から返される値を使用できます。これらの変数は On Clicked、On Double Clicked や On Mouse Enter と On Mouse Move フォームイベントで更新されます。

Note: ピクチャの座標システム中 [x;y] は、" 繰り返し " フォーマットを除き、ピクチャ表示フォーマットにかかわらず常に同じ場所をポイントします。

バインドされた四角が選択領域中に含まれるすべての要素は、たとえ他の要素の下になっていても、対象となります。

SVG SHOW ELEMENT

SVG SHOW ELEMENT ({*; }pictureObject; id{;margin})

引数	型		説明
*	*	→	指定時 : pictureObject はオブジェクト名 (文字) 省略時 : pictureObject は変数
pictureObject	ピクチャ	→	オブジェクト名 (* 指定時) または変数またはフィールド (* 省略時)
id	文字	→	表示する要素の ID 属性
margin	倍長整数	→	表示のマージン (デフォルトでピクセル単位)

新しい SVG SHOW ELEMENT コマンドは、id 引数で指定した "id" 属性を持つ要素を表示するように、pictureObject SVG ドキュメントを移動します。

オプションの * 引数を渡すと、pictureObject 引数はオブジェクト名 (文字) です。この場合、コマンドはオブジェクトにアタッチされた描画ピクチャに適用されます。この引数を渡さないと、pictureObject 引数は変数やフィールドであり、変数参照 (変数オブジェクトのみ) またはフィールド参照を渡します。この場合この変数を使用するすべてのオブジェクトに描画されたピクチャに適用されます (最初に描画されたピクチャを除きます)。

コマンドは SVG ドキュメントを移動し、境界がバインドされた四角で定義されるすべてのオブジェクトが表示されるようにします。margin 引数を使用して、ドキュメントの縁からの、オブジェクトを表示するマージンを指定できます。言い換えればバインドした四角は margin ピクセルだけ高さと幅が大きくなります。デフォルトで移動値は 4 ピクセルです。

このコマンドはスクロールバー付きの " 左上 " 表示モードのみで効果があります。

XML

Note: 以前のバージョンの "XML ユーティリティ " テーマは "XML" テーマに置きかえられました。このテーマには DOM と SAX 両方のコマンドで使用される汎用 XML コマンドや、XSLT、SVG コマンドがグループ化されます。

統合された XML コマンドは 4D v12 で更新され、また新しいコマンドが追加されました。4D と XML 間のデータ変換メカニズムが強化されました。

4D v12 で XML BLOB 使用時の留意点

XML 構造はテキストタイプのデータに基づいています。しかし 4D XML コマンド (例えば DOM Parse XML variable) は、XML データの処理に、元の型にかかわらず、通常 BLOB 型の引数を受け入れます。これは以前のバージョンの 4D でテキスト型の変数に 32KB 制限があったために必要でした。

4D v11 よりテキスト型の変数とフィールドは最大までのデータを含むことができます。よってテキストを BLOB に入れることは全くお勧めできません。BLOB の利用はバイナリデータの処理にとどめるべきです。XML 仕様により準拠するため、4D v12 でバイナリデータは、たとえ BLOB にテキストが含まれていても、自動で Base64 にエンコードされます。

Note: ENCODE および DECODE コマンド (BASE64 ENCODE および BASE64 DECODE に名称変更) はテキスト型の引数を処理できるようになりました。これらのコマンドは XML データのエンコーディングのフレームワークで有用です。これらのコマンドは " ツール " テーマに含まれています (177 ページの " ツール " 参照)。

XML SET OPTIONS

XML SET OPTIONS(elementRef | document; selector; value{; selector2; value2;...;selectorN; valueN})

引数	型		説明
elementRef | document	文字 | DocRef	→	XML ルート要素参照、または開かれたドキュメント参照
selector1...N	整数	→	設定するオプション
value1...N	整数	→	オプションの値

新しい XML SET OPTIONS コマンドを使用して、カレントセッションおよびカレントユーザの XML パラメタを変更できます。

このコマンドは " ツリー " タイプ (DOM) や " ドキュメント " タイプ (SAX) の XML 構造に適用できます。第一引数にはルート要素参照 (elementRef)、あるいは開かれた SAX ドキュメント (document) を渡します。

このコマンドで設定されるオプションは、4D から XML の方向でのみ利用されます (4D への XML 値の読み込みには効果ありません)。以下のコマンドがこのオプションを使用します :

- DOM SET XML ELEMENT VALUE
- DOM SET XML ATTRIBUTE
- SAX ADD XML ELEMENT VALUE

selector に変更するオプションを渡し、value にオプションの新しい値を渡します。必要なだけ selector/value の組を渡すことができます。"XML" テーマの以下の定数を使用しなければなりません :

- selector = XML String encoding(1)
このセレクターは 4D の文字列を要素値に変換する方法を指定します。これはエスケープ文字を必要とする属性の変換とは関係ありません。
  - value = XML With escaping(1) (デフォルト値): 4D 文字列から XML 要素値への変換は文字の置き換えにより行われます。テキスト型のデータは自動で解析され、使用不可の文字 (<&>') は XML 実体参照 (&<> '") に置きかえられます。

- value = XML Raw data(2): 4D 文字列は生データとして送信されます。4D はエンコードや解析を行いません。4D の値は可能であれば XML フラグメントに変換され、ターゲット要素の子要素として挿入されます。値が XML フラグメントとして扱えない場合、新しい CDATA ノードに生データとして挿入されます。

- selector = XML Date encoding(2)
このセレクターは 4D の日付を変換する方法を指定します。例えば日本で !2010/01/14! は :

- - value = XML ISO (1) (デフォルト値): タイムゾーンを指定せずに xs:datetime フォーマットを使用します。結果 : "2010-01-14"。(SQL を使用して) 時間部が指定されていた場合、それは失われます。
  - value = XML Local (2): タイムゾーンを指定して xs:date フォーマットを使用します。結果 : "2010-01-14 +09:00"。(SQL を使用して) 時間部が指定されていた場合、それは失われます。
  - value = XML Datetime local (3): xs:dateTime (ISO 8601) フォーマットを使用します。タイムゾーンが指定されます。このフォーマットでは (SQL を使用して) 時間部が指定されていた場合、時間部が保持されます。結果 : "<Date>2010-01-14T00:00:00 +09:00</Date>".
  - value = XML UTC (4): xs:date フォーマットを使用します。結果 : "2010-01-14Z"。(SQL を使用して) 時間部が指定されていた場合、それは失われます。
  - value = XML Datetime UTC (5): xs:dateTime (ISO 8601) フォーマットを使用します。このフォーマットでは (SQL を使用して) 時間部が指定されていた場合、時間部が保持されます。結果 : "<Date>2010-01-14T00:00:00Z</Date>".

Note: XML Local および XML Datetime local 値は UTC (Universal Time Coordinated) で表現された日付を提供しません。日付は変更されず、時差が付加されます。このフォーマットは変換してその逆変換をおこなうような場合に便利です。
XML UTC および XML Datetime UTC 値はフォーマット上は先と同じですが、UTC で表現されます。相互の互換性のために、このフォーマットを優先的に使用すべきです。値は変更されません。

- selector = XML Time encoding(3)
このセレクターは 4D の時間を変換する方法を指定します。例えば日本で ?02:00:46? は以下のように変換されます。エンコーディングは時間もしくは時刻どちらを表現するかにより異なります。

時刻タイプの場合 :

- value = XML Datetime UTC (5): 時刻は UTC (Universal Time Coordinated) で表現されます。UTC への変換は自動です。結果 : "<Duration>0000-00-00T17:00:46Z</Duration>".
- value = XML Datetime local (3): 時刻は 4D エンジンのマシンの時差を使用して表現されます。結果 : "<Duration>0000-00-00T02:00:46+09:00</Duration>".
- value = XML Datetime local absolute (1) (デフォルト値): 時刻は時差なしで表現されます。値は変更されません。結果 : "<Duration>0000-00-00T02:00:46</Duration>".

時間タイプの場合 :

- value = XML Seconds (4): 0 時からの秒数。経過時間を表すため値は変更されません。結果 : "<Duration>7246</Duration>".
- value = XML Duration (2): XML Schema Part 2: Datatypes Second Edition に基づいた経過時間の表現。経過時間を表すため値は変更されません。結果 : "<Duration>PT02H00M46S</Duration>".

- selector = XML Binary encoding(5)
このセレクターでバイナリデータの変換方法を指定できます。
  - XML Base64 (1) (デフォルト値): バイナリデータはBase64に変換されます。
  - XML Data URI scheme (2): バイナリデータは Base64 に変換され、"data:;base64" ヘッダーが追加されます。このフォーマットは主にブラウザに自動でピクチャをデコードさせるために使用されます。また SVG ピクチャの挿入のためにも必要とされます。詳細は以下のサイトを参照してください : http://en.wikipedia.org/wiki/Data_URI_scheme.
- selector = XML Picture encoding(6)
このセレクターは (Base64 にエンコードされる前の) ピクチャの変換方法を指定します。
  - XML Convert to PNG (1) (デフォルト値): Base64 にエンコードされる前に、ピクチャは PNG に変換されます。
  - XML Native codec(2): Base64 にエンコードされる前に、ピクチャは自身の第一のネイティブストレージに変換されます。SVG ピクチャをエンコードするときにこれらのオプションを使用しなければなりません (例題参照)。
- selector = XML Indentation (4)
このセレクターは XML ドキュメントのインデントを指定します。
  - XML With indentation (1) (デフォルト値): ドキュメントはインデントされます。

- XML No indentation (2): ドキュメントはインデントされません。コンテンツは一行に収められます。

▶ 例題 1: SVG ピクチャの挿入

XML SET OPTIONS ($pictElemRef;XML Binary encoding;
XML Data URI scheme)
XML SET OPTIONS ($pictElemRef;XML Picture encoding;
XML Native codec)
DOM SET XML ATTRIBUTE ($pictElemRef;"xlink:href";PictVar)

参照 : XML GET OPTIONS

XML GET OPTIONS

XML GET OPTIONS(elementRef | document; selector; value{; selector2; value2;...;selectorN; valueN})

引数	型		説明
elementRef | document	文字 | DocRef	→	XML ルート要素参照 または 開かれたドキュメントの参照
selector1...N	整数	→	取得するオプション
value1...N	整数	←	オプションの現在値

新しい XML GET OPTIONS コマンドは、カレントセッションおよびカレントユーザで使用されている 1 つ以上の XML パラメタの現在値を取得するために使用します。

selector には取得するオプションを指定する、"XML" テーマの定義済み定数を渡します。

selector 引数に渡すオプションと返される値に関する詳細は、XML SET OPTIONS コマンドの説明を参照してください。

参照 : XML SET OPTIONS

XML DECODE

XML DECODE (xmlValue; object)

引数	型		説明
xmlValue	テキスト	→	XML 構造から取得したテキスト型の値
object	フィールド | 変数	←	変換した XML の値を受け取る 4D 変数またはフィールド

XML DECODE コマンドは XML 文字列に格納されている値を 4D 型の値に変換します。変換は以下のルールに基づいて自動で行われます :

値	例	日本語システム上での変換例
数値	<Price>8,5</Price> <Price>8.5</Price>	実数 : 8.5
ブール	<Double>1</Double> <Double>0</Double> または <Double>true</Double> <Double>false</Double>	ブール : True/False
BLOB		Base64 デコード
ピクチャ		Base64 デコード + BLOB to picture コマンド
日付	2009-10-25T01:03:20+09:00	時間部とタイムゾーンを取り除く : !2009/10/25!
時間	2009-10-25T01:03:20+09:00	日付部とタイムゾーンを取り除く : ?01:03:20?

- XML ドキュメントから、属性として格納されているデータを読み込む。

  XML ドキュメントの例 :

  <CD Date="2003-01-01T00:00:00Z" Description="This double CD reissued by EMI in 1995 combines 4 Stabat mater hymns. One by Rossini interpreted by the Berlin Symphony Orchestra, directed by Karl Forster. Followed by a work of Verdi, by the Philharmonic Orchestra, directed by Carlo Maria Giulini. On the second CD, you will find Francis Poulenc interpreted by Régine Crespin. This compilation ends with a little-known version, that of the Polish composer Karol Szymanowski. Polish National Radio Symphony Orchestra directed by Antoni Wit" Double="true" Duration="7246" Type="Sacred music" CD_ID="5" Performer="Various" Price="8.5" Title="4 Stabat mater"/>

```
Repeat
   MyEvent:=SAX Get XML node(DocRef)

   Case of
   : (MyEvent=XML Start Element)
      ARRAY TEXT(arrAttrNames;0)
      ARRAY TEXT(arrAttrValues;0)
      SAX GET XML ELEMENT
(DocRef;vName;vPrefix;arrAttrNames;arrAttrValues)
      If (vName="CD")
         CREATE RECORD([CD])
```

```
      For ($i;1;Size of array(arrAttrNames))
        $attrName:=arrAttrNames{$i}
        Case of
        : ($attrName="CD_ID")
          XML DECODE(arrAttrValues{$i};[CD]CD_ID)
        : ($attrName="Title")
          [CD]Work:=arrAttrValues{$i}
        : ($attrName="Price")
          XML DECODE(arrAttrValues{$i};[CD]Price)
        : ($attrName="Date")
          XML DECODE(arrAttrValues{$i};[CD]Date entered)
        : ($attrName="Duration")
          XML DECODE(arrAttrValues{$i};[CD]Total_duration)
        : ($attrName="Double")
          XML DECODE(arrAttrValues{$i};[CD]Double_CD)
        End case
      End for
    End if
  ...
  End case
Until (MyEvent=XML Start Document)
```

XML DOM

いくつかの新しいコマンドが XML DOM テーマに追加されました : DOM Create XML element arrays、DOM REMOVE XML ATTRIBUTE、DOM Insert XML element、DOM Append XML element、DOM Append XML child node、DOM GET XML CHILD NODES、DOM Get XML document ref。さらに XML SET OPTIONS と XML GET OPTIONS コマンドの実装により、既存の DOM SET XML OPTIONS が名称変更され DOM SET XML DECLARATION となり、その動作が変更されました。

DOM Create XML element arrays

DOM Create XML element arrays (elementRef; xPath{; attribNamesArray; attribValuesArray}{; attribNamesArray2; attribValuesArray2; ...; attribNamesArrayN; attribValuesArrayN}) → 文字

引数	型		説明
elementRef	文字	→	XML ルート要素参照
xPath	テキスト	→	作成する XML 要素の XPath パス
attribNamesArray	配列	→	属性名配列
attribValuesArray	配列	→	属性値配列
戻り値	文字	←	作成された XML 要素の参照

新しい DOM Create XML element arrays コマンドを使用して elementRef 要素に新しい要素を追加したり、さらに配列形式で渡された属性とその値も追加できます。

配列をサポートしている以外、このコマンドは DOM Create XML element と同じです。動作についてはこのコマンドの説明を参照してください。

さらに、DOM Create XML element arrays コマンドは attribNamesArray と attribValuesArray 引数に複数の属性とその値のペアを配列として渡すことができます。attribValuesArray にはテキスト、日付、数値、そしてピクチャ型の配列を渡せます。4D は自動で必要な変換を行います。新しい XML SET OPTIONS コマンドを使用してこの変換をコントロールできます。

配列は事前に作成されていなければなあらず、またペアで動作します。必要なだけ配列のペアを渡すことができ、またそれぞれのペアごとに必要なだけ要素を渡すことができます。

- 以下の要素を作成します :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<RootElement>
  <Elem1>
    <Elem2>
      <Elem3 Font="Verdana" Size="10" Style="Bold"></Elem3>
    </Elem2>
  </Elem1>
</RootElement>
```

これを行うには、以下のように記述します :

```
ARRAY TEXT(arrAttNames;3)
ARRAY TEXT(arrAttValues;3)
arrAttNames{1}:="Font"
arrAttValues{1}:="Verdana"
arrAttNames{2}:="Style"
arrAttValues{2}:="10"
arrAttNames{3}:="Style"
arrAttValues{3}:="Bold"
vRootRef:=DOM Create XML ref("RootElement")
vxPath:="/RootElement/Elem1/Elem2/Elem3"
vElementRef:=DOM Create XML element arrays(vRootRef;
    vxPath;arrAttNames;arrAttValues)
```

参照 : DOM Create XML element

DOM REMOVE XML ATTRIBUTE

DOM REMOVE XML ATTRIBUTE (elementRef; attrName)

引数	型		説明
elementRef	文字	→	XML 要素参照
attrName	文字	→	取り除く属性

新しい DOM REMOVE XML ATTRIBUTE コマンドは、attrName で指定された属性が elementRef で指定された XML 要素に存在すれば、そこから取り除きます。

属性が正しく取り除かれると、OK システム変数に 1 が設定されます。elementRef 要素に attrName という名前の属性が存在しない場合、エラーが返され、OK システム変数に 0 が設定されます。

▶ 以下の XML 構造において :

```
<?xml version="1.0" ?>
- <STANZA>
    <LINE N="1">I heard a thousand blended notes,</LINE>
    <LINE N="2">While in grove I sate reclined,</LINE>
    <LINE N="3">In that sweet mood when pleasant thoughts</LINE>
    <LINE N="4">Bring sad thoughts to the mind.</LINE>
  </STANZA>
```

以下のコードは一番目の属性 "N=1" を取り除きます :

```
C_BLOB(myBlobVar)
C_STRING(16;$xml_Parent_Ref;$xml_Child_Ref)
C_LONGINT($LineNum)

$xml_Parent_Ref:=DOM Parse XML variable (myBlobVar)
$xml_Child_Ref:=DOM Get first child XML element ($xml_Parent_Ref)
DOM REMOVE XML ATTRIBUTE($xml_Child_Ref;"N")
```

参照 : DOM GET XML ATTRIBUTE BY INDEX, DOM GET XML ATTRIBUTE BY NAME

DOM Insert XML element

DOM Insert XML element (targetElementRef; sourceElementRef; childIndex) → String

引数	型		説明
targetElementRef	文字	→	親 XML 要素参照
sourceElementRef	文字	→	挿入する XML 要素参照
childIndex	倍長整数	→	新しい要素を挿入するターゲットとなる子要素のインデックス
戻り値	文字	←	新しい XML 要素の参照

新しい DOM Insert XML element コマンドを使用して、targetElementRef 引数に渡された参照を持つ XML 要素の子要素の間に、新しい XML 要素を挿入できます。

sourceElementRef に挿入する要素を渡します。この要素は、DOM ツリーの中の既存の XML 要素の参照として渡さなければなりません。

childIndex 引数は、新しい要素を挿入する、親要素の子要素を指定するために使用します。この引数にはインデックス番号を渡します。番号が有効でない場合 (例えばこのインデックス番号を持つ子要素が存在しない)、新しい要素は親要素の最初の子要素の前に挿入されます。

コマンドは取得した XML 要素の参照を返します。

- 以下の XML 構造で、1 番目と 2 番目の本を入れ替えます :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<NewBooks>
 <Book>
      <Title>Open Source Web Services</Title>
      <Author>Collective</Author>
      <Date>2003</Date>
      <ISBN>2-7440-1507-5</ISBN>
      <Publisher>Wrox</Publisher>
  </Book>
<Book>
      <Title>Building XML Web services</Title>
      <Author>Scott Short</Author>
      <Date>2002</Date>
      <ISBN>2-10-006476-2</ISBN>
      <Publisher>Microsoft Press</Publisher>
  </Book>
</NewBooks>
```

これを行うには、以下のコードを実行します :

```
C_TEXT($rootRef)
$rootRef:=DOM Parse XML source ("") // XML ドキュメントを選択
If (OK=1)
  C_TEXT($newStruct)
  $newStruct:=DOM Create XML Ref ("NewBooks")

  $bookRef:=DOM Find XML element
($rootRef;"/BookCatalogue/Book[1]")
  $newElementRef:=DOM Append XML element ($newStruct;$bookRef)
```

```
   $bookRef:=DOM Find XML element
($rootRef;"/BookCatalogue/Book[2]")
   C_TEXT($newElementRef)
   $newElementRef:=DOM Insert XML element ($newStruct;$bookRef;1)

   DOM CLOSE XML($newStruct)
   DOM CLOSE XML($rootRef)
End if
```

参照 : DOM Append XML element

DOM Append XML element

DOM Append XML element (targetElementRef; sourceElementRef) → 文字

引数	型		説明
targetElementRef	文字	→	XML 親要素参照
sourceElementRef	文字	→	追加する XML 要素参照
戻り値	文字	←	新しい XML 要素参照

新しい DOM Append XML element コマンドを使用して、targetElementRef 引数に渡した参照を持つ XML 要素の子に新しい XML 要素を追加できます。

sourceElementRef には追加する要素を渡します。この要素は、DOM ツリーの中の既存の XML 要素の参照として渡さなければなりません。これは targetElementRef の既存の子要素の最後に追加されます。

- DOM Insert XML element コマンドの例題参照。

参照 : DOM Insert XML element

DOM Append XML child node

DOM Append XML child node (elementRef; childType; childValue) → 文字

引数	型		説明
elementRef	文字	→	XML 要素参照
childType	倍長整数	→	追加する子のタイプ
childValue	文字 | BLOB	→	子ノードとして挿入するテキストまたは変数 (テキストまたは BLOB)
戻り値	文字	←	子 XML 要素の参照

新しい DOM Append XML child node コマンドを使用して 、elementRef で指定したノードに値 childValue を追加できます。

追加されるノードのタイプは childType 引数で指定します。この引数には "XML" テーマの以下の定数を渡します :

子ノードのタイプ	定数 (値)
テキスト	XML DATA (6)
処理命令	XML Processing Instruction (3)
コメント	XML Comment (2)
CDATA	XML CDATA (7)
DOCTYPE	XML DOCTYPE (10)
要素	XML ELEMENT (11)

childValue には挿入する値を渡します。文字または 4D 変数 (文字または BLOB) を渡します。この引数の内容は常にテキストに変換されます。

Note: elementRef 引数がドキュメントノード (トップレベルノード) を指す場合、コマンドは他のノードよりも前に "Doctype" ノードを挿入します。同じことが処理命令とコメントについても言え、常にルートノードの前 (ただし Doctype ノードの後) に挿入されます。

- テキストタイプノードを追加する :

```
Reference := DOM Create XML element (elementRef;"myElement")
DOM SET XML ELEMENT VALUE(Reference ; "Hello")
temp:= DOM Create XML element (Reference ; "br")
temp := DOM Append XML child node (Reference ; XML DATA; "New")
temp := DOM Create XML element (Reference ; "br")
temp := DOM Append XML child node(Reference ; XML DATA; "York")
```

結果 :

```
<myElement>Hello<br/>New<br/>York</myElement>
```

- 処理命令ノードを追加する :

```
$Txt_instruction:="xml-stylesheet type = \"text/xsl\" href=\"style.xsl\""
Reference:= DOM Append XML child node(elementRef;XML Processing Instruction ; $Txt_instruction )
```

結果 (最初の要素の前に挿入される):

```
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
```

- コメントタイプのノードを追加する :

```
Reference := DOM Append XML child node (elementRef;XML Comment; "Hello world")
```

結果 :
<!--Hello world-->

- CDATA タイプのノードを追加する :

  Reference := **DOM Append XML child node** (elementRef; XML CDATA; "12 < 18")

  結果 :
  <element><![CDATA[12 < 18]]></element>

- Doctype 定義タイプのノードを追加または置換する :

  Reference :=**DOM Append XML child node**(elementRef;XML DOCTYPE; "Books SYSTEM \"Book.DTD\"")

  結果 (最初の要素の前に挿入される):
  <!DOCTYPE Books SYSTEM "Book.DTD">

- 要素タイプのノードを追加または置換する
  - childValue 引数が XML の断片の場合、子ノードとして挿入されます :

  Reference:=**DOM Append XML child node**(elementRef;XML ELEMENT; "<child>simon</child><child>eva</child>")

  結果 :
  ```
  <parent>
    <child>simon</child>
    <child>eva</child>
  </parent>
  ```

  - そうでなければ新しい空の子要素が追加されます :

  Reference:=**DOM Append XML child node** (elementRef;XML ELEMENT; "tbreak")

  結果 :
  ```
  <parent>
    <tbreak/>
  </parent>
  ```

  childValue の内容が有効でない場合、エラーが返されます。

  参照 : DOM GET XML CHILD NODES

DOM GET XML CHILD NODES

DOM GET XML CHILD NODES(elementRef; childTypesArr; nodeRefsArr)

引数	型		説明
elementRef	文字	→	XML 要素参照
childTypesArr	倍長整数配列	←	子ノードのタイプ
nodeRefsArr	テキスト配列	←	子ノードの参照または値

新しい DOM GET XML CHILD NODES コマンドは、elementRef で指定された XML 要素のすべての子ノードのタイプと参照または値を返します。

子ノードのタイプは childTypesArr 配列に返されます。コマンドから返される値は、"XML" テーマの以下の定数と比較できます :

定数	型	値
XML Comment	倍長整数	2
XML Processing Instruction	倍長整数	3
XML DATA	倍長整数	6
XML CDATA	倍長整数	7
XML DOCTYPE	倍長整数	10
XML ELEMENT	倍長整数	11

詳細は DOM Append XML child node コマンドの説明を参照してください。

nodeRefsArr 配列には、(内容あるいは命令に応じて) 要素の参照または値が返されます。

- 以下の XML 構造があるとき :
<myElement>Hello
New
York</myElement>

以下のステートメント実行後 :

elementRef:=**DOM Find XML element** ($root;"myElement")
DOM GET XML CHILD NODES(elementRef;$typeArr;$textArr)

$typeArr と $textArr 配列には以下の値が格納されます :
$typeArr{1}=6$textArr{1} = "Hello"
$typeArr{2}=11$textArr{2} = "AEF1233456878977" (要素参照
)
$typeArr{3}=6$textArr{3} = "New"
$typeArr{4}=11$textArr{4} = "AEF1237897734568" (要素参照
)
$typeArr{5}=6$textArr{5} = "York"

参照 : DOM Append XML child node

DOM Get XML document ref

DOM Get XML document ref (elementRef) → 文字

引数	型		説明
elementRef	文字	→	DOM ツリー中の既存の要素の参照
戻り値	文字	←	DOM ツリーの最初の要素の参照 (ドキュメントノード)

新しい DOM Get XML document ref コマンドを使用して、elementRef に渡した DOM ツリーの " ドキュメント " 参照を取得できます。ドキュメント要素は DOM ツリーの最初の要素であり、ルート要素の親です。

ドキュメント要素の参照を使用して "Doctype" や " 処理命令 " ノードを処理できます。これは DOM Append XML child node と DOM GET XML CHILD NODES コマンドでのみ利用できます。
このレベルで、処理命令やコメントを追加したり、Doctype ノードを置換したりすることだけが可能です。ここに CDATA やテキストノードを作成することはできません。

▶ この例題では、XML ドキュメントの DTD 宣言を取得します :

```
C_TEXT($rootRef)
$rootRef:=DOM Parse XML source("")
If (OK=1)
  C_TEXT($documentRef)
     // ドキュメントノードを探します。このノードにはルートノード
     // の前に DOCTYPE が記述されています。
  $documentRef:=DOM Get XML document ref($rootRef)
  ARRAY TEXT($typeArr;0)
  ARRAY TEXT($valueArr;0)
     // このノードの子ノード中で DOCTYPE タイプのノードを探す
  DOM GET XML CHILD NODES($refDocument;$typeArr;$valueArr)
  C_TEXT($text)
  $text:=""
  $pos:=Find in array($typeArr;XML DOCTYPE)
  If ($pos>-1)
     // DTD 宣言を $text に取得
     $text:=$text+"Doctype: "+$valueArr{$pos}+Char(Carriage return)
  End if
  DOM CLOSE XML($rootRef)
End if
```

参照 : DOM GET XML CHILD NODES, DOM Append XML child node

DOM SET XML DECLARATION

DOM SET XML DECLARATION (elementRef; encoding{; standalone{; indentation}}))

引数	型		説明
elementRef	文字	→	XML 要素参照
encoding	文字	→	XML ドキュメントの文字セット (デフォルトは UTF-8)
standalone	ブール	→	True= ドキュメントはスタンドアロン False (デフォルト)= ドキュメントはスタンドアロンではない
indentation	ブール	→	廃止、使用しないでください

互換性に関する注意 :s

- DOM SET XML DECLARATION コマンドは以前のバージョンで DOM SET XML OPTIONS という名称でした。
- DOM オプションは SAX オプションに依存しなくなります。

indentation 引数は以前のバージョンの 4D との互換性のために保持されていますが、4D v12 ではこの引数の利用をお勧めしません。コンバージョンからインデントを設定するには、新しい XML SET OPTIONS コマンドの利用を強く推奨します。

DOM GET XML ELEMENT VALUE

DOM GET XML ELEMENT VALUE(elementRef; elementValue{; cDATA})

今後、elementValue に受け取ったオブジェクトが BLOB である場合、コマンドは自動で Base64 のデコードを試行します。実際 DOM SET XML ELEMENT VALUE コマンドにより BLOB が作成されると、それは自動で Base64 にエンコードされます。BASE64 DECODE コマンドを呼び出す必要はありません。

参照 : SAX GET XML ELEMENT VALUE

XML SAX

XML SAX テーマの 3 つのコマンドが変更されました。XML SET OPTIONS と XML GET OPTIONS コマンドの実相により、既存の SAX SET XML OPTIONS コマンドは SAX SET XML DECLARATION に名称変更され、動作も変更されました。

SAX SET XML DECLARATION

SAX SET XML DECLARATION (document; encoding{; standalone{; indentation}})

引数	型		説明
document	DocRef	→	開かれたドキュメントの参照
encoding	文字	→	XML ドキュメント文字セット (デフォルトで UTF-8)
standalone	ブール	→	True= ドキュメントはスタンドアロン False (デフォルト)= ドキュメントはスタンドアロンではない
indentation	ブール	→	廃止、使用しないでください

互換性に関する注意 :

- SAX SET XML DECLARATION コマンドは以前のバージョンで SAX SET XML OPTIONS という名称でした。
- SAX オプションは開かれたドキュメントのみに有効となり、ドキュメントを閉じた後は保持されません。

indentation 引数は以前のバージョンの 4D との互換性のために保持されていますが、4D v12 ではこの引数の利用をお勧めしません。コンバージョンからインデントを設定するには、新しい XML SET OPTIONS コマンドの利用を強く推奨します。

SAX OPEN XML ELEMENT ARRAYS

SAX OPEN XML ELEMENT ARRAYS (document; tab{; attribNamesArray; attribValuesArray}{; attribNamesArray2; attribValuesArray2; ...; attribNamesArrayN; attribValuesArrayN})

テキスト型の配列に加え、SAX OPEN XML ELEMENT ARRAYS コマンドは日付、数値、ブール、そしてピクチャ配列を attribValuesArray 引数に受け入れるようになりました。4D は自動で必要な変換を行います。この変換は新しい XML SET OPTIONS コマンドで変更できます。

SAX GET XML ELEMENT VALUE

SAX GET XML ELEMENT VALUE(document; value)

DOM GET XML ELEMENT VALUE コマンドのように、4D は取得した値を引数に渡した変数の方への変換を試みます。型が BLOB の場合、コマンドは自動で Base64 のデコードを試みます。

参照 : DOM GET XML ELEMENT VALUE

新しいシステム変数

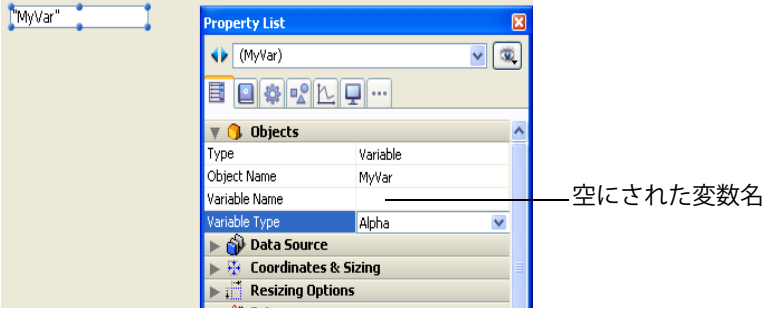
4D v12 で新しいシステム変数が 2 つ使用できるようになりました。これらの変数エラー処理メソッドのフレームワークで、エラーの発生元を特定するために使用されます :

- Error method: テキスト型のシステム変数です。この変数にはエラーが発生したメソッドの名前が格納されます。
- Error line: 倍長整数型のシステム変数です。エラーが発生したメソッドの発生元の行番号が格納されます。

Error システム変数と同様、これらのシステム変数も ON ERR CALL コマンドでインストールされるエラー処理メソッド中でのみ利用できます。これらの変数に、エラーの発生元となったメソッド内からアクセスしたい場合、値をプロセス変数にコピーする必要があります。

ダイナミック変数

フォームオブジェクト (ボタン、入力可変数、チェックボックス等) に割り当てる変数を、必要に応じて、動的に 4D に作成させることができるようになりました。これを行うには、オブジェクトのプロパティリストの " 変数名 " を空にします :



変数に名前が付けられていないと、フォームがロードされたとき、4D はオブジェクト用に新しい変数を作成して、インタープリターのプロセス変数空間でユニークな名称を計算します (つまりこのメカニズムはコンパイルモードでも利用できます)。この一時的な変数はフォームが閉じられるときに破棄されます。

この原理がコンパイルモードでも動作するようにするために、変数の型をプロパティリストの " 変数タイプ " メニューで設定しなければなりません。ダイナミック変数を選択し (変数名なし)、" 変数タイプ " メニューで**なし**を選択すると、コンパイラから型未指定のエラーが返されます。

Note: 以前のバージョンの 4D のように、変数に名前が付けられていると、" 変数タイプ " メニューは実際には変数の型を決定しません。ただプロパティリストのオプションを方針するだけです (ピクチャ変数を除く)。名前を付けた変数の型を指定するには、ランゲージを使用しなければなりません。

4D コードで、ダイナミック変数には OBJECT Get pointer コマンドで取得したポインタを使用してアクセスできます。例えば :

```
// "hstart" オブジェクトの変数に時間 12:00:00 を代入する
$p: = Object Get pointer(Object named;"hstart")
$p->: = ?12:00:00?
```

この新しいメカニズムの利点は二つあります :

- 一つめは、このメカニズムにより、同じホストフォーム上で複数個使用される " サブフォーム " タイプのコンポーネントの開発が可能になります。例として開始日と終了日を設定するために、ホストフォームに 2 個の datepicker サブフォームを配置することを考えてみましょう。このサブフォームは年月日を選択するオブジェクトを使用します。開始日と終了日ではこれらのオブジェクトが異なる変数に対して動作する必要があります。4D がユニークな変数名を作成することで、困難さが解決されます。
- もう一つは、メモリの利用量を減らせることです。フォームオブジェクトはプロセスおよびインタープロセス変数とともにのみ動作します。しかしコンパイルモードでは、それぞれのプロセス変数のインスタンスが、サーバプロセスを含むすべてのプロセスで作成されます。このインスタンスは、たとえフォームがセッション中で使用されなくても、メモリを占有します。フォームのロード時に 4D が動的に変数を作成することで、メモリを節約できます。

Note:s • 変数名がないとき、フォームエディタ中でオブジェクト名はクォーテーションマーク付きで表示されます。
• このメカニズムは変数名を割り当てられるすべてのオブジェクトで使用できます (リストボックスを除く)。

定数

この節で説明する新しい定数を使用して追加の機能にアクセスできます。

Open window

新しい Has toolbar button Mac OS 定数を使用して、Mac OS でウィンドウ中にツールバー管理ボタンを表示できます (116 ページの " ツールバーボタン (Mac OS)" 参照)。
Mac OS インタフェースの変更に伴い、Metal Look 定数は Texture appearance に名称変更されました (フォームエディタのプロパティリストのオプション " メタルルック " も " テクスチャ (Mac OS)" に変更されました)。

定数	型	値
Has toolbar button Mac OS	倍長整数	8192
Texture appearance	倍長整数	2048

▶ 使用例 :

$NewWin:=**Open window**(10;10;1010;810;Plain window+Has toolbar button Mac OS)

Note: Has toolbar button Mac OS 定数は **Open form window** コマンドでも使用できます。

System Documents

"System Documents" テーマに新しい定数が追加されました : Folder separator。この定数は呼び出された OS により異なる値を返します :

定数	型	値
Folder separator	文字	\ (Windows) または : (Mac OS)

この定数を使用して、プラットフォームの違いにかかわらず正しいパス名を構築できます。

Form Events

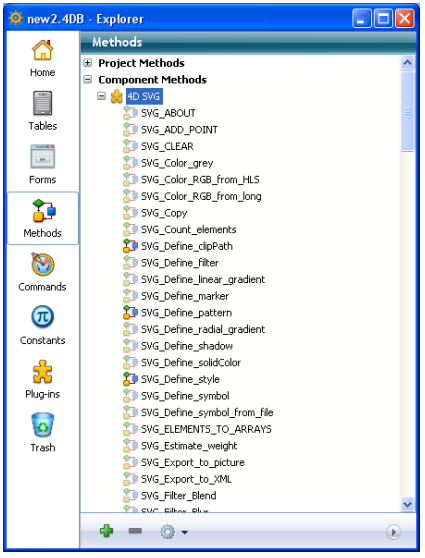
"Form Events" テーマに新しいフォームイベントが 2 つ追加されました :

定数	型	値	説明
On bound variable change	倍長整数	54	サブフォームにバインドされた変数が変更されたときに生成されます。74 ページの " バインドした変数の管理 " 参照。
On Mac toolbar button	倍長整数	55	Mac OS で、ユーザがツールバー管理ボタンをクリックすると生成されます。116 ページの " ツールバーボタン (Mac OS)" 参照。

4D Widget

4D v12 では、4D Widget と呼ばれる新しいコンポーネントの形式で提供されるハイレベルなフォームオブジェクトが統合されています。これらのオブジェクトは 82 ページの " ウィジェット " で説明しています。

このコンポーネントの設定は、エクスプローラのメソッドページにリストされる特定のメソッドを使用して行われます :



この節ではこれらのメソッドのシンタックスを説明します。

DatePicker

DatePicker はフォームに日付を入力、表示するための直感的なグラフィックインタフェースオブジェクトを提供します。詳細は 84 ページの "DatePicker" を参照してください。

コンポーネントメソッドを使用してこれらのオブジェクトの動作をカスタマイズできます。DatePicker Display Dialog コマンドを使用して DatePicker カレンダーオブジェクトをポップアップウィンドウに表示します。

DatePicker Display Dialog

DatePicker Display Dialog {(top; left)} → Date

引数	型		説明
top	倍長整数	→	ウィンドウの上位置
left	倍長整数	→	ウィンドウの左位置
戻り値	日付	←	ユーザが選択した日付

DatePicker Display Dialog コマンドは DatePicker カレンダをポップアップウィンドウに表示します (ポップアップタイプのウィンドウはユーザがウィンドウの外をクリックするか、**Enter** または **Esc** キーが押されると自動で閉じられます)。

2 つのオプションの引数を使用して、ウィンドウを開く左と上の座標を指定できます。これら 2 つの引数はペアで渡さなければなりません。1 つしか渡されない場合、それは無視されます。
これらの引数が省略されると、ウィンドウはクリックされた場所で開かれます。

この関数は、ユーザが DatePicker カレンダ内で選択した日付を返します。日付が選択されずにウィンドウが閉じられたとき、コマンドは空の日付を返します。

▶ この例題はボタンがクリックされると DatePicker カレンダを表示します :

```
OBJECT GET CORDINATES (*;"MyCalendarButton";$x1;$y1;$x2;$y2)
$MyLocalDate:=DatePicker Display Dialog ($x1;$y1)
If($MyLocalDate #!00/00/00!)
   [Event]DateRV:=$MyLocalDate
End if
```

DatePicker SET MIN DATE

DatePicker SET MIN DATE (objectName; minDate)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
minDate	日付	→	入力可能な日付の下限

DatePicker SET MIN DATE コンポーネントメソッドを使用して、DatePicker カレンダに入力可能な日付の下限を設定できます (下限より過去の日付はカレンダ中で灰色に表示されます)。

objectName 引数では、コマンドを適用するサブフォームのインスタンスを指定します。この引数にはカレントフォーム中に表示されるサブフォームの名前を渡さなければなりません。

minDate はシステム言語に基づくデフォルト入力フォーマットで表現されなければなりません。空の日付 (!00/00/00!) を渡すと、今日の日付より前の日付はすべて入力可能になります。

Note: 日付の下限が上限より大きい場合 (DatePicker SET MAX DATE 参照)、日付の入力はできなくなります。

▶ フォームの 2 つのサブフォーム "DP1" と "DP2" に 2 つの DatePicker カレンダが置かれています。

```
// 一番目のカレンダで 2009/01/01 以前を無効にする
DatePicker SET MIN DATE("DP1";!2009/01/01!)

// 二番目のカレンダで 2009/03/01 以前を無効にする
DatePicker SET MIN DATE("DP2";!2009/03/01!)
```

参照 : DatePicker SET MAX DATE

DatePicker SET MAX DATE

DatePicker SET MAX DATE (objectName; maxDate)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
maxDate	日付	→	入力可能な日付の上限

DatePicker SET MAX DATE コンポーネントメソッドを使用して、DatePicker カレンダに入力可能な日付の上限を設定できます (上限より後の日付はカレンダ中で灰色に表示されます)。

objectName 引数では、コマンドを適用するサブフォームのインスタンスを指定します。この引数にはカレントフォーム中に表示されるサブフォームの名前を渡さなければなりません。

maxDate はシステム言語に基づくデフォルト入力フォーマットで表現されなければなりません。空の日付 (!00/00/00!) を渡すと、今日の日付より後の日付はすべて入力可能になります。

Note: 日付の上限が下限より小さい場合 (DatePicker SET MIN DATE 参照)、日付の入力はできなくなります。

▶ オブジェクト名 "ReturnDate" で、2009/12/31 より後を無効にする :

DatePicker SET MAX DATE("ReturnDate";!2009/12/31!)

参照 : DatePicker SET MIN DATE

DatePicker SET WEEK FIRST DAY

DatePicker SET WEEK FIRST DAY (objectName; dayNum)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
dayNum	倍長整数	→	先頭に表示する曜日の番号

DatePicker SET WEEK FIRST DAY コンポーネントメソッドを使用して DatePicker カレンダの最左に表示する曜日を指定できます。デフォルトで最初の曜日は月曜日です。

objectName 引数では、コマンドを適用するサブフォームのインスタンスを指定します。この引数にはカレントフォーム中に表示されるサブフォームの名前を渡さなければなりません。

dayNum 引数には "Days and Months" テーマの 4D 定数を渡します。

▶ 日曜日からの表示に設定する :

DatePicker SET WEEK FIRST DAY("mycalender";Sunday)

▶ 木曜日からの表示に設定する :

DatePicker SET WEEK FIRST DAY("mycalender";Thursday)

参照 : DatePicker SET DAYS OFF

DatePicker SET DAYS OFF

DatePicker SET DAYS OFF (objectName; dayType; ptrHolidaysArray)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
dayType	倍長整数	→	休日タイプ
ptrDaysOffArray	ポインタ	→	休日の日付またはブールタイプ配列へのポインタ

DatePicker SET DAYS OFF コンポーネントメソッドを使用して DatePicker カレンダに表示する休日を設定できます。これらの日付はボールドおよびイタリックで表示され、ユーザは依然選択が可能です。

objectName 引数では、コマンドを適用するサブフォームのインスタンスを指定します。この引数にはカレントフォーム中に表示されるサブフォームの名前を渡さなければなりません。

このコンポーネントメソッドでは毎週や毎年の休日や、特別な休日を設定できます。dayType 引数を使用して休日タイプを指定します :

- 0 = 毎週の休日 (デフォルトで土曜と日曜)
- 1 = 毎年の休日 (1 月 1 日など)
- 2= 特別な休日でその年のみに設定

配列を作成して、ptrDaysOffArray 引数にその配列へのポインタを渡すことで、休日を設定します。配列のタイプは dayType に渡す値により異なります :

- dayTypeに0 (毎週の休日) を渡した場合、ptrDaysOffArrayには7要素のブール配列へのポインタを渡します。True はその曜日が休日であることを示します。

- dayType に 1 (毎年の休日) または 2 (特別な休日) を渡した場合、ptrDaysOffArray には日付配列へのポインタを渡します。この配列では、それぞれの要素に休日を示す有効な日付が格納されていなければなりません。日付ははシステム言語に基づくデフォルトフォーマットで表現されなければなりません。1 (毎年の休日) を渡した場合、年は無視されます。有効な日付であればどんな値でも使用できます

▶ デフォルトの土曜日と日曜日に代えて金曜日を毎週の休みに設定する :

```
ARRAY BOOLEAN ($arrbDaysOff;7)
   // デフォルトですべてのブール配列要素は False です。なので
   // 初期化コードを追加する必要はありません。
$arrbDaysOff{Friday }:=True
DatePicker SET DAYS OFF (->$arrbDaysOff)
```

▶ 特別な休日を設定 :

```
ARRAY DATE($arrdUniqueDays;0)
   // 年も設定される
APPEND TO ARRAY($arrdUniqueDays;!2008/02/15!)
APPEND TO ARRAY($arrdUniqueDays;!2009/02/12!)
APPEND TO ARRAY($arrdUniqueDays;!2010/02/17!)
DatePicker SET DAYS OFF (1;->$arrdUniqueDays)
```

参照 : DatePicker SET WEEK FIRST DAY, DatePicker SET DEFAULT DAYS OFF

DatePicker SET DEFAULT DAYS OFF

DatePicker SET DEFAULT DAYS OFF (DayType; ptrDaysOffArray)

引数	型		説明
dayType	倍長整数	→	休日タイプ
ptrDaysOffArray	ポインタ	→	休日の日付またはブール配列へのポインタ

DatePicker SET DEFAULT DAYS OFF コンポーネントメソッドを使用して DatePicker コンポーネントのすべてのカレンダーに現れる休日を設定できます。これらの日付は太字および斜体で表示され、ユーザーは選択が可能です。

この設定は、メソッド呼び出し後に表示されるカレンダーに対し有効で、すでに表示さたカレンダーには適用されません。既存のカレンダーに適用させたい場合は DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

このコンポーネントメソッドでは毎週および毎年、さらには特別な休日も設定できます。dayType 引数を使用して休日のタイプを指定します :

- 0 = 毎週の休み (デフォルトで土日)
- 1 = 毎年の休み (1 月 1 日など)
- 2 = その年だけに設定する特別な休日

配列を作成して ptrDaysOffArray にポインタ渡しすることで休日を設定します。配列の型は dayType に渡す値に基づきます :

- *dayType* に 0 を渡す (毎週の休み) 場合、7 要素を持つブール配列へのポインタを渡さなければなりません。True を渡された要素は毎週の休みに設定されます。
- *dayType* に 1 または 2 を渡す (毎年あるいは特別な休日) 場合、*ptrDaysOffArray* には日付配列へのポインタを渡します。この配列にはそれぞれの要素に休日を示す有効な日付が格納されていなければなりません。日付はシステムランゲージのデフォルトフォーマットで表現されていなければなりません。1 を渡した場合、年は無視されます。有効な日付であれば、どんな値でも渡すことができます。

▶ 毎年の休みを設定する (USA の例):

```
ARRAY DATE($arrdRepeatedDays;0)
   // 年は無視されるので、仮に 2000 を使用
APPEND TO ARRAY($arrdRepeatedDays;!2000/01/01!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/02/02!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/02/14!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/03/17!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/04/01!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/10/31!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/11/11!)
APPEND TO ARRAY($arrdRepeatedDays;!2000/12/25!)
DatePicker SET DEFAULT DAYS OFF (0;->$arrdRepeatedDays)
```

参照 : DatePicker SET DAYS OFF

DatePicker SET DEFAULT MIN DATE

DatePicker SET DEFAULT MIN DATE (minDate)

引数	型		説明
minDate	日付	→	入力可能な日付の下限

DatePicker SET DEFAULT MIN DATE コンポーネントメソッドを使用して、DatePicker コンポーネントのすべてのカレンダの 、入力可能な日付の下限を設定できます。

このパラメタは後で作成されるカレンダに適用され、既存のカレンダには影響を与えない点に留意してください。既存のカレンダに適用したい場合は DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

▶ 日付の下限を 2000 年 1 月 1 日にする :

DatePicker SET DEFAULT MIN DATE (!2000/01/01!)

参照 : DatePicker SET MIN DATE, DatePicker SET DEFAULT MAX DATE

DatePicker SET DEFAULT MAX DATE

DatePicker SET DEFAULT MAX DATE (maxDate)

引数	型		説明
maxDate	Date	→	入力可能な日付の上限

DatePicker SET DEFAULT MAX DATE コンポーネントメソッドを使用して、DatePicker コンポーネントのすべてのカレンダの 、入力可能な日付の上限を設定できます。

このパラメタは後で作成されるカレンダに適用され、既存のカレンダには影響を与えない点に留意してください。既存のカレンダに適用したい場合は DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : DatePicker SET MAX DATE, DatePicker SET DEFAULT MIN DATE

DatePicker SET DEFAULT 1ST DAY

DatePicker SET DEFAULT 1ST DAY(dayNum)

引数	型		説明
dayNum	倍長整数	→	最初の曜日

DatePicker SET DEFAULT 1ST DAY コンポーネントメソッドを使用して、すべての DatePicker カレンダで、デフォルトで左に表示される曜日を設定できます。

このパラメタは後で作成されるカレンダに適用され、既存のカレンダには影響を与えない点に留意してください。既存のカレンダに適用したい

場合は DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : DatePicker SET MAX DATE, DatePicker SET DEFAULT MIN DATE

DatePicker APPLY DEFAULT VALUES

DatePicker APPLY DEFAULT VALUES(objectName)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクトの名前

DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用して、objectName サブフォームオブジェクトのすべての DatePicker パラメタを、デフォルト値に再設定できます。

これらのデフォルト値は初期値またはコンポーネントの SET DEFAULT コマンド で設定された値です。
このコマンドは即座に効果を及ぼします。objectName のデフォルト値は即座に更新されます。オブジェクトに割り当てられた変数は、新しい値を有効にするために更新されるかもしれないことに留意してください。例えば新しい最小日付のデフォルト値が 2000/01/01 に設定され、objectName に割り当てられた変数が 1995/05/05 のとき、その値は自動で 2000/01/01 に変更されます。

DatePicker のパラメタには以下があります :

- 入力可能な最小および最大日付
- 週の最初の曜日
- 毎週毎年の休日および特別な休日

▶ この例題では Date1 オブジェクトのパラメタをデフォルト設定に設定します :

DatePicker APPLY DEFAULT VALUES ("Date1")

DatePicker RESET DEFAULT VALUES

DatePicker RESET DEFAULT VALUES

引数	型	説明
このメソッドは引数を必要としません		

DatePicker RESET DEFAULT VALUES コンポーネントメソッドは DatePicker コンポーネントのデフォルト値を初期値にリセットします。このコンポーネントメソッド実行後は以下のように設定されます :

- 最小および最大入力可能日付は 00/00/00 (制限なし)
- 最初の曜日は 2 (月曜)
- 毎週の休日は土曜と日曜

- 毎年の休日と特別な休日はなし

このパラメタは後で作成されるカレンダに適用され、既存のカレンダには影響を与えない点に留意してください。既存のカレンダに適用したい場合は DatePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

SearchPicker

SearchPicker コンポーネントを使用して、フォームに標準の検索エリアを含めることができます。詳細は 82 ページの "SearchPicker" を参照してください。

このコンポーネントのメソッドはエリアのヘルプテキストを設置するために使用されます。

SearchPicker SET HELP TEXT

SearchPicker SET HELP TEXT (helpText)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
helpText	テキスト	→	表示するテキスト

SearchPicker SET HELP TEXT コンポーネントメソッドを使用して、objectName で指定した検索エリアのバックグランドに、入力不可のグレイアウトされたテキストを表示できます。このテキストはユーザがエリア内をクリックすると消えます。

▶ エリア内に "Country" と表示します。これにより検索がこの変数に関連するものだということを示すことができます :

```
C_TEXT(vCountry)
SearchPicker SET HELP TEXT("vSearch";"Country")
```

TimePicker

TimePicker コンポーネントはフォーム中に時間を入力あるいは表示するためのグラフィカルなオブジェクトを提供します。詳細は 87 ページの "TimePicker" を参照してください。

このコンポーネントのメソッドは、これらのオブジェクトの動作のカスタマイズに使用されます。

TimePicker SET MIN TIME TimePicker SET MIN TIME (objectName; minTime)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
minTime	時間	→	入力可能な時間の下限

TimePicker SET MIN TIME コンポーネントメソッドを使用して、objectName で指定したオブジェクトが受け入れる時間の下限を設定できます。それよりも小さな時間が入力されると、受け入れられません。

参照 : TimePicker SET MAX TIME

TimePicker SET MAX TIME TimePicker SET MAX TIME (objectName; maxTime)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
maxTime	時間	→	入力可能な時間の上限

TimePicker SET MAX TIME コンポーネントメソッドを使用して、objectName で指定したオブジェクトが受け入れる時間の上限を設定できます。それよりも大きな時間が入力されると、受け入れられません。

参照 : TimePicker SET MIN TIME

TimePicker SET STEP TimePicker SET STEP (objectName; step)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
step	時間	→	2 つの時間値の間隔

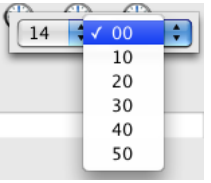
TimePicker SET STEP コンポーネントメソッドを使用して、objectName で指定したオブジェクトで利用される時間と時間の間のステップを設定できます。このパラメタはポップアップメニューとして表示される TimePicker にのみ適用されます。

step 値は 1 分から 1 時間の間で、60 分を割り切れる整数値のみを指定できます。つまり実際 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 そして 60 が利用可能です。他の値は自動でこの原則に沿うよう丸められます。

- "time1" という名前のポップアップメニュー形式の TimePicker を設定します。入力可能な時間は 8:30 から 16:30 で、間隔は 10 分です :

```
TimePicker SET MIN TIME ("time1";?08:30:00?)
TimePicker SET MAX TIME ("time1";?16:30:00?)
```

TimePicker SET STEP ("time1";?00:10:00?)



TimePicker SET LABEL AM

TimePicker SET LABEL AM (objectName; label)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
label	テキスト	→	AM に使用するラベル

TimePicker SET LABEL AM コンポーネントメソッドを使用して、AM/PM フォーマットで表示される TimePicker オブジェクトの "AM" ラベルを変更できます。コマンドは objectName で指定されたオブジェクトに適用されます。デフォルトでシステムの am/pm ラベルが使用されます。

- AM のとき、システムラベルの代わりにデフォルトで "in the morning" を使用する :

TimePicker SET LABEL AM("clock"; "in the morning")

参照 : TimePicker SET LABEL PM

TimePicker SET LABEL PM

TimePicker SET LABEL PM (objectName; label})

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名
label	テキスト	→	PM に使用するラベル

TimePicker SET LABEL PM コンポーネントメソッドを使用して、AM/PM フォーマットで表示される TimePicker オブジェクトの "PM" ラベルを変更できます。コマンドは objectName で指定されたオブジェクトに適用されます。デフォルトでシステムの am/pm ラベルが使用されます。

- PM のとき、システムラベルの代わりにデフォルトで "in the evening" を使用する :

TimePicker SET LABEL PM("clock"; "in the evening")

参照 : TimePicker SET LABEL AM

TimePicker SET DEFAULT MIN TIME

TimePicker SET DEFAULT MIN TIME (minTime)

引数	型		説明
minTime	時間	→	入力可能な時間の下限

TimePicker SET DEFAULT MIN TIME コンポーネントメソッドを使用して、すべての TimePicker オブジェクトで使用される入力可能な時間の下限を設定できます。

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : TimePicker SET MAX TIME

TimePicker SET DEFAULT MAX TIME

TimePicker SET DEFAULT MAX TIME (maxTime)

引数	型		説明
maxTime	時間	→	入力可能な時間の上限

TimePicker SET DEFAULT MAX TIME コンポーネントメソッドを使用して、すべての TimePicker オブジェクトで使用される入力可能な時間の上限を設定できます。

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : TimePicker SET MIN TIME

TimePicker SET DEFAULT STEP

TimePicker SET DEFAULT STEP (step)

引数	型		説明
step	時間	→	2 つの時間値の間隔

TimePicker SET DEFAULT STEP コンポーネントメソッドを使用して、すべての TimePicker オブジェクトの時間と時間の間隔を設定できます。

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : TimePicker SET STEP

TimePicker SET DEFAULT LABEL AM

TimePicker SET DEFAULT LABEL AM (label)

引数	型		説明
label	テキスト	→	AM に使用されるラベル

TimePicker SET DEFAULT LABEL AM コンポーネントメソッドを使用して、AM/PM フォーマットで表示されるすべての TimePicker オブジェクトのデフォルトの "AM" ラベルを変更できます。

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : TimePicker SET LABEL AM

TimePicker SET DEFAULT LABEL PM

TimePicker SET DEFAULT LABEL PM (label)

引数	型		説明
label	テキスト	→	PM に使用されるラベル

TimePicker SET DEFAULT LABEL PM コンポーネントメソッドを使用して、AM/PM フォーマットで表示されるすべての TimePicker オブジェクトのデフォルトの "PM" ラベルを変更できます。

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

参照 : TimePicker SET LABEL PM

TimePicker APPLY DEFAULT VALUES

TimePicker APPLY DEFAULT VALUES (objectName)

引数	型		説明
objectName	テキスト	→	サブフォームオブジェクト名

TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用して、objectName サブフォームオブジェクトのすべての TimePicker パラメタを現在のデフォルト値に再設定できます。

これらのデフォルト値は初期値またはコンポーネントの SET DEFAULT コマンド で設定された値です。
このコマンドは即座に効果を及ぼします。objectName のデフォルト値は即座に更新されます。オブジェクトに割り当てられた変数は、新しい値

を有効にするために更新されるかもしれないことに留意してください。例えば新しい最小時間のデフォルト値が 07:00:00 に設定され、objectName に割り当てられた変数が 06:00:00 のとき、その値は自動で 07:00:00 に変更されます。

TimePicker パラメタには以下があります :

- 入力可能時間の下限と上限
- AM および PM ラベル
- 分単位のステップ

参照 : TimePicker RESET DEFAULT VALUES

TimePicker RESET DEFAULT VALUES

TimePicker RESET DEFAULT VALUES

引数	型	説明
このコンポーネントメソッドは引数を必要としません		

TimePicker RESET DEFAULT VALUES コンポーネントメソッドは TimePicker コンポーネントのパラメタを初期値にリセットします。このコンポーネントメソッド実行後は以下のように設定されます :

- 入力可能な時間の下限は 08:00:00
- 入力可能な時間の上限は 20:00:00
- AM と PM ラベルはシステムラベル
- 分単位のステップは 00:15:00

この設定は後で作成されるオブジェクトに適用され、既存のオブジェクトには影響を与えない点に留意してください。既存のオブジェクトに適用したい場合は TimePicker APPLY DEFAULT VALUES コンポーネントメソッドを使用します。

4D SVG

バージョン 12 の 4D SVG コンポーネントにはいくつかの新しいコマンドおよび更新されたコマンドが含まれています。この節ではこれらのコマンドについて説明します。

属性

SVG_SET_CLASS

SVG_SET_CLASS (svgObject; class)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
class	テキスト	→	クラス名

SVG_SET_CLASS は svgObject に渡されたオブジェクトのクラスを設定します。svgObject が有効な参照でない場合エラーが生成されます。

▶ SVG_Define_style コマンドの例題参照

参照 : http://www.w3.org/TR/SVG/styling.html#ClassAttribute

SVG_SET_CLIP_PATH

SVG_SET_CLIP_PATH (svgObject; clipPathID)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
clipPathID	テキスト	→	クリップパス名

SVG_SET_CLIP_PATH コマンドは、svgObject に渡されたオブジェクトで clipPathID という名称のクリップパスを設定します。svgObject が有効な参照でない場合やクリップパスが定義されていない場合エラーが生成されます。

▶ 円形のクリップパスを定義し、画像に割り当てます。

```
// 円形のクリップパスを定義
$Dom_clipPath:=SVG_Define_clip_path ($Dom_SVG;"theClip")
$Dom_circle:=SVG_New_circle ($Dom_clipPath;150;100;100)

// グループを作成
$Dom_g:=SVG_New_group ($Dom_SVG)

// 画像を挿入
$Txt_path:= Get 4D folder(6)+"logo.svg"
READ PICTURE FILE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image ($Dom_g;$Pic_buffer)
SVG_SET_ID ($Dom_picture;"MyPicture")
```

```
// クリップパスをグループに適用
SVG_SET_CLIP_PATH ($Dom_g;"theClip")
```

- 角の丸い四角のクリップパスの例 :

```
// 四角のクリップパスを定義
$Dom_clipPath:=SVG_Define_clip_path ($Dom_SVG;"theClip")
$Dom_rect:=SVG_New_rect ($Dom_clipPath;5;10;320;240;10;10)

// グループを作成
$Dom_g:=SVG_New_group ($Dom_SVG)

// 画像を挿入
$Txt_path:= Get 4D folder(6)+"logo.svg"
READ PICTURE FILE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image ($Dom_g;$Pic_buffer)
SVG_SET_ID ($Dom_picture;"MyPicture")

// グループにクリップパスを適用
SVG_SET_CLIP_PATH ($Dom_g;"theClip")
```

参照 : http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath

SVG_SET_FILL_RULE

SVG_SET_FILL_RULE (svgObject; fillRule)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
fillRule	テキスト	→	塗りつぶしオブジェクトのモード

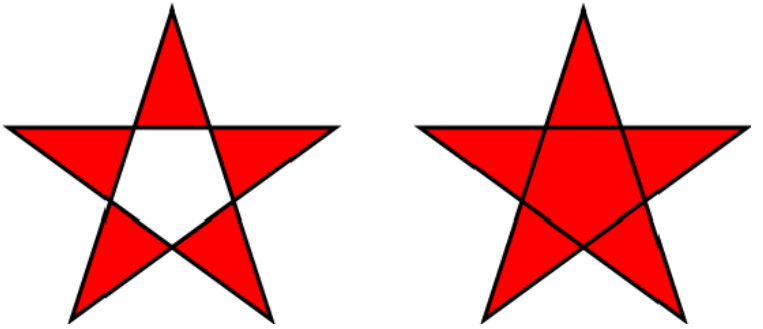
SVG_SET_FILL_RULE コマンドは svgObject で指定された SVG オブジェクトの塗りつぶしルールを指定するために使用します。

fillRule 引数は以下のいずれかの値でなければなりません : "nonzero"、"evenodd"、または "inherit"。そうでない場合エラーが生成されます。

▶ 塗りつぶしモードの例 :

```
//'evenodd' 塗りつぶしルールでパスを作成する
$Dom_path:=SVG_New_path ($Dom_SVG;250;75)
SVG_PATH_LINE_TO ($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE ($Dom_path)
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_WIDTH ($Dom_path;3)
SVG_SET_FILL_RULE ($Dom_path;"evenodd")

//'nonzero' 塗りつぶしルールで同様のオブジェクトを作成する
$Dom_path:=SVG_New_path ($Dom_SVG;250;75)
SVG_PATH_LINE_TO ($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE ($Dom_path)
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_WIDTH ($Dom_path;3)
SVG_SET_FILL_RULE ($Dom_path;"nonzero")
// 水平方向に移動
SVG_SET_TRANSFORM_TRANSLATE ($Dom_path;300)
```



参照 : http://www.w3.org/TR/SVG/painting.html#FillRuleProperty

SVG_SET_SHAPE_RENDERING

SVG_SET_SHAPE_RENDERING (svgObject; rendering)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
rendering	テキスト	→	レンダリングのタイプ

SVG_SET_SHAPE_RENDERING コマンドは、svgObject で指定したオブジェクトのグラフィック要素のレンダリングに関して、どのトレードオフを適用するかを設定するために使用します。svgObject が有効な SVG オブジェクトでない場合、エラーが生成されます。

rendering 引数は以下のいずれかの値でなければなりません : "auto"、"optimizeSpeed"、"crispEdges"、"geometricPrecision"、または "inherit"。そうでなければエラーが生成されます。

参照 : http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#ShapeRenderingProperty

SVG_SET_STROKE_DASHARRAY

SVG_SET_STROKE_DASHARRAY (svgObject; dash ;value1{ ;...; valueN})

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
dash	実数	→	先頭のダッシュの長さ
value1...N	倍長整数	→	スペースとダッシュの長さ

SVG_SET_STROKE_DASHARRAY コマンドは、svgObject に渡された SVG オブジェクトの、パスのアウトラインに使用されるダッシュとギャップのパターンを設定するために使用されます。svgObject が有効な SVG オブジェクトでない場合、エラーが生成されます。

dash 引数の値はドットパターンの最初のダッシュの長さを示します。value1 以降の引数が省略されると、ドットラインはその長さのダッシュと、同じ長さのギャップで構成されます。

dash 引数の小数部は、その値がヌルでない場合、パターン中でダッシュが開始されるまでの距離を示します。

dash が 0 の場合、ドットパターンは取り除かれます。

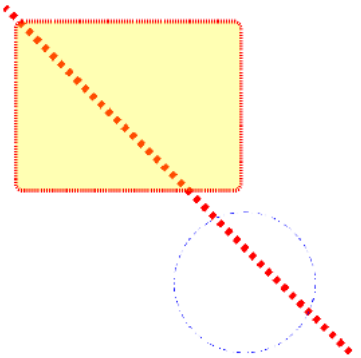
value1 以降の引数は、最初のダッシュに続くギャップとダッシュの長さを指定します。(最初のダッシュを含み) 奇数の value が渡されると、偶数の value が生成されるまで value リストが繰り返されます。

▶ ドットラインの例 :

```
// 線
$Dom_line:=SVG_New_line ($Dom_SVG;10;10;500;500)
SVG_SET_STROKE_WIDTH ($Dom_line;10)
SVG_SET_STROKE_DASHARRAY ($Dom_line;8,099)
SVG_SET_STROKE_BRUSH ($Dom_line;"red")

// 四角
$Dom_rect:=SVG_New_rect ($Dom_SVG;25;30;320;240;10;10;"red";
                                                "yellow:30")
SVG_SET_STROKE_WIDTH ($Dom_rect;5)
SVG_SET_STROKE_DASHARRAY ($Dom_rect;2)

// 円
$Dom_circle:=SVG_New_circle ($Dom_SVG;350;400;100;"blue";"none")
SVG_SET_STROKE_DASHARRAY ($Dom_circle;2;4;6;8)
```



参照 : http://www.w3.org/TR/SVG/painting.html#StrokeProperties

SVG_SET_STROKE_MITERLIMIT

SVG_SET_STROKE_MITERLIMIT (svgObject; join)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
join	倍長整数	→	結合の値

SVG_SET_STROKE_MITERLIMIT コマンドは、svgObject で指定された SVG オブジェクトの、パスとアウトライン間の留め接ぎの長さ制限を設定するために使用します。svgObject が有効な SVG オブジェクトでない場合、エラーが生成されます。

join 引数が -1 の場合、値はデフォルト値 (4) になります。join 引数が 0 の場合、属性定義が取り除かれます。0 未満の他の値はエラーとなります。

参照 : http://www.w3.org/TR/SVG/painting.html#StrokeProperties

カラーとグラデーション

SVG_Color_RGB_from_CMYK

SVG_Color_RGB_from_CMYK (cyan ; magenta ; yellow ; black {; format}) → 戻り値

引数	型		説明
cyan	倍長整数	→	シアン値
magenta	倍長整数	→	マゼンタ値
yellow	倍長整数	→	イエロー値
black	倍長整数	→	ブラック値
format	倍長整数	→	カラーフォーマット
戻り値	テキスト	←	カラー文字列

SVG_Color_RGB_from_CMYK コマンドは、引数に渡したシアン、マゼンタ、イエロー、そしてブラックの 4 色に対応するカラーを表す文字列を返します。返される文字列はデフォルトで、SVG のレンダリングエンジンで認識される "RGB(red,greeen,blue)" の形式です。

cyan、magenta、yellow、そして black は 0 から 100% の倍長整数値です。

オプションの format 引数を使用して返されるカラー文字列のフォーマットを指定できます。使用できる値は以下の通りです :

値	フォーマット
1 (デフォルト)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_HLS

SVG_Color_RGB_from_HLS (hue ; luminosity ; saturation {; format}) → 戻り値

引数	型		説明
hue	倍長整数	→	色相値
luminosity	倍長整数	→	明度値
saturation	倍長整数	→	彩度値
format	倍長整数	→	カラーフォーマット
戻り値	テキスト	←	カラー文字列

SVG_Color_RGB_from_HLS コマンドは色相、明度、および彩度に対応するカラーを表す文字列を返します。返される文字列はデフォルトで、SVG のレンダリングエンジンで認識される "RGB(red,greeen,blue)" の形式です。

色相は 0 から 360° の倍長整数値です。

明度と彩度は 0 から 100% の倍長整数値です。

オプションの format 引数を使用して返されるカラー文字列のフォーマットを指定できます。使用できる値は以下の通りです :

値	フォーマット
1 (デフォルト)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_GET_COLORS_ARRAY

SVG_GET_COLORS_ARRAY (colorNamesArrayPointer)

引数	型		説明
colorNamesArrayPointer	ポインタ	→	カラー名を受け取るテキスト配列へのポインタ

SVG_GET_COLORS_ARRAY コマンドはテキスト配列へのポインタ colorNamesArrayPointer に SVG で認識されるカラー名を返します。

参照 : http://www.w3.org/TR/SVG/types.html#ColorKeywords

描画

SVG_Add_object

SVG_Add_object (targetSVGObject ; sourceSVGObject)→ 戻り値

引数	型		説明
targetSVGObject	SVG_Ref	→	親要素の参照
sourceSVGObject	SVG_Ref	→	追加されるオブジェクトの参照
戻り値	SVG_Ref	←	SVG オブジェクトの参照

SVG_Add_object コマンドは、targetSVGObject で指定された SVG コンテナ内に、sourceSVGObject で指定された SVG オブジェクトを配置し、その参照を返します。SVG コンテナはドキュメントのルートあるいはこのタイプの要素を含むことのできる他の SVG オブジェクトへの参照です。

ストラクチャおよび定義

SVG_Define_clip_path

SVG_Define_clip_path (parentSVGObject; clipPathID) → 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素の参照
clipPathID	テキスト	→	クリップパス名
戻り値	SVG_Ref	←	クリップパスの参照

SVG_Define_clip_path コマンドは、parentSVGObject で指定された SVG コンテナ内に新しいクリップパスを定義し、その参照を返します。親 SVGObject が SVG ドキュメントでないか、SVG ドキュメントに属していない場合、エラーが生成されます。

clipPathID 引数でクリップパス名を指定します。この名前はクリップパスをオブジェクトに関連付けるために使用されます。ドキュメント中に同じ名前の要素が既に存在する場合、エラーが生成されます。

- SVG_SET_CLIP_PATH コマンドの例題参照。

参照 : http://www.w3.org/TR/2001/REC-SVG-20010904/masking.html#EstablishingANewClippingPath

SVG_Define_pattern

SVG_Define_pattern (parentSVGObject; patternID{; width {; height {; x {; y {; units {; viewBox}}}}}}) → 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素の参照
patternID	Text	→	パターン名
width	倍長整数	→	パターンの幅
height	倍長整数	→	パターンの高さ
x	倍長整数	→	パターンの x 位置
y	倍長整数	→	パターンの y 位置
unit	テキスト	→	長さと位置の単位
viewBox	テキスト	→	ビューボックス
戻り値	SVG_Ref	←	パターンの参照

SVG_Define_pattern コマンドは parentSVGObject で指定した SVG コンテナに新しいカスタムパターンを設定し、その参照を返します。parentSVGObject が SVG ドキュメントでないか、SVG ドキュメントに属していない場合、エラーが生成されます。

patternID 引数でパターン名を指定します。この名前はパターンをオブジェクトに関連付けるために使用されます。ドキュメント中に同じ名前の要素が既に存在する場合、エラーが生成されます。

オプションの width、height、x、y、unit、そして viewBox 引数はパターンの参照四角、言い換えれば どのようにパターンタイルが配置されるかを指定します。

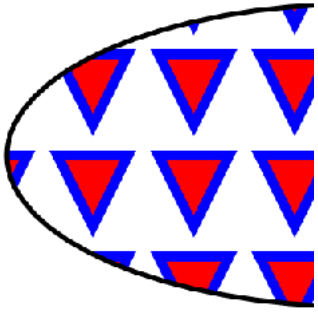
カラー表現が期待されているとき、パターンは "url(#id)" 文字列を値として渡すことで、塗りつぶしやストロークペイントとして割り当てられます。

- パターンを設定し、楕円の塗りつぶしに使用する :

```
// パターンの定義
$Dom_pattern:=SVG_Define_pattern ($Dom_SVG;"MyPattern";100;100;
                                  0;0;"";"0 0 10 10")
$Dom_path:=SVG_New_path ($Dom_pattern;0;0)

SVG_PATH_MOVE_TO ($Dom_path;0;0)
SVG_PATH_LINE_TO ($Dom_path;7;0)
SVG_PATH_LINE_TO ($Dom_path;3,5;7)
SVG_PATH_CLOSE ($Dom_path)
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_BRUSH ($Dom_path;"blue")

// パターンで塗りつぶされた楕円を描画する
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150;"black";
                              "url(#MyPattern)";5)
```

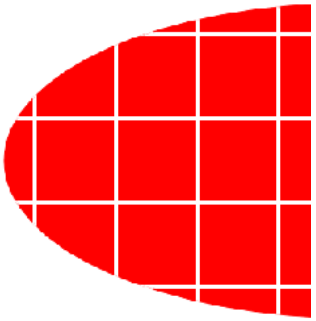


- パターンを設定し、楕円の塗りつぶしと外周のストロークに使用する :

```
// パターンの定義
$Dom_pattern:=SVG_Define_pattern ($Dom_SVG;"MyPattern ";80;80;0;
                                  0;"";"0 0 20 20")
$Dom_rect:=SVG_New_rect ($Dom_pattern;0;0;20;20;0;0;"white";"red")
```

```
// 楕円を描画
$Dom_ellipse:=SVG_New_ellipse ($Dom_SVG;400;200;350;150)

// 塗りつぶしとアウトラインにパターンを適用
SVG_SET_FILL_BRUSH ($Dom_ellipse;"url(#MyPattern)")
SVG_SET_STROKE_BRUSH ($Dom_ellipse;"url(#MyPattern)")
```



参照 : http://www.w3.org/TR/SVG/pservers.html#Patterns

SVG_Define_style

SVG_Define_style (parentSVGObject ; style { ; type {; media}}) → 戻り値

引数	型		説明
parentSVGObject	SVG_Ref	→	親要素の参照
style	テキスト	→	スタイル定義、または 使用するファイルのパス名
type	テキスト	→	内容のタイプ
media	テキスト	→	メディア記述子
戻り値	SVG_Ref	←	スタイル参照

SVG_Define_style コマンドは、parentSVGObject で指定された SVG コンテナに新しいスタイルシートを設定し、その参照を返します。parentSVGObject が SVG ドキュメントでなく、また SVG ドキュメントに属してもいない場合、エラーが生成されます。

style 引数は SVG のコンテンツに直接スタイルシートを組み込むために使用します :

- style 引数が有効な CSS ファイルへのパス名を含む場合、スタイル定義は外部スタイルシートを参照するスタイルで行われます。パスが "#" または "file:" から始まる場合、それはデータベースの "Resources" フォルダからの相対パスを表します。
- style 引数には "http://… " の URL タイプも指定できます。この場合スタイルシートは外部リソースとして参照されます。

オプションの type 引数は要素内容に使用されるスタイルシートのランゲージを指定します。デフォルト値は "text/css" です。

オプションの media 引数はスタイル情報が適用されるメディアを指定するために使用します。この引数を省略すると、使用されるデフォルト値は "all" です。値が CSS2 で認識されるメディアタイプのリストに含まれない場合、エラーが生成されます。

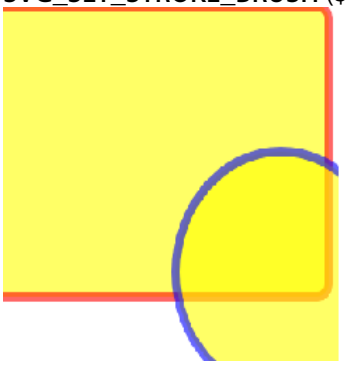
- 要素の一つに組み込みスタイルと重ねを設定する :

```
// スタイルの定義
$Txt_style:=".colored {fill: yellow; fill-opacity: 0.6; stroke: red;
                                                    stroke-width:8; stroke-opacity: 0.6}"
SVG_Define_style ($Dom_SVG;$Txt_style)

// グループを作成し、デフォルトスタイルを割り当てる
$Dom_g:=SVG_New_group ($Dom_SVG)
SVG_SET_CLASS ($Dom_g;"colored")

// 四角を描画
$Dom_rect:=SVG_New_rect ($Dom_g;25;30;320;240;10;10;"";"")

// 円を描画し、オーバーレイと枠線カラーを適用
$Dom_circle:=SVG_New_circle ($Dom_g;300;250;100;"";"")
SVG_SET_STROKE_BRUSH ($Dom_circle;"blue")
```



- "Resources" フォルダ内の "dev" フォルダに配置された "mystyle.css" ファイルを参照します :

```
// スタイル定義
SVG_Define_style ($Dom_svg;"#dev/monstyle.css")

// グループを作成しデフォルトスタイルを割り当てる
$Dom_g:=SVG_New_group ($Dom_SVG)
SVG_SET_CLASS ($Dom_g;"colored")
```

```
// 四角を描画
$Dom_rect:=SVG_New_rect ($Dom_g;25;30;320;240;10;10;"";"")
```

mystyle.css ファイル :

```
.colored {fill: red; fill-opacity: 0.6; stroke: blue; stroke-width:8; stroke-opacity: 0.6}
```

参照 : http://www.w3.org/TR/SVG/styling.html#StyleElement

SVG_DELETE_OBJECT

SVG_DELETE_OBJECT (svgObject)

引数	型		説明
svgObject	SVG_Ref	→	削除する SVG 要素参照

SVG_DELETE_OBJECT コマンドは svgObject で指定された SVG オブジェクトを、それが属するドキュメントから削除します。svgObject が有効な参照でない場合、エラーが生成されます。

SVG_Get_default_encoding

SVG_Get_default_encoding → 戻り値

引数	型		説明
戻り値	テキスト	←	文字セット

SVG_Get_default_encoding コマンドは、新しいドキュメントが作成されるときに使用されるエンコーディングを返します。

SVG_SET_DEFAULT_ENCODING

SVG_SET_DEFAULT_ENCODING {(encoding)}

引数	型		説明
encoding	テキスト	→	文字セット

SVG_SET_DEFAULT_ENCODING コマンドは、新しいドキュメントを作成するときに使用されるエンコーディングを設定するために使用します。

encoding 引数が省略された場合、コマンドはデフォルトエンコーディングである "UTF-8" を設定します。

SVG_SET_PATTERN_CONTENT_UNITS

SVG_SET_PATTERN_CONTENT_UNITS (patternObject; sysCoord)

引数	型		説明
patternObject	SVG_Ref	→	変更するパターンの参照
sysCoord	テキスト	→	使用される座標のシステム

SVG_SET_PATTERN_CONTENT_UNITS コマンドは、patternObject で指定されたパターンの内容で使用する座標システムを設定します。patternObject がパターンでない場合、エラーが生成されます。

sysCoord 引数は使用されるシステムの名前を指定します。"userSpaceOnUse" または "objectBoundingBox" のいずれかを指定します。そうでなければエラーが生成されます。

参照 : http://www.w3.org/TR/SVG/pservers.html#Patterns

SVG_SET_PATTERN_UNITS

SVG_SET_PATTERN_UNITS (patternObject; sysCoord)

引数	型		説明
patternObject	SVG_Ref	→	変更するパターンの参照
sysCoord	テキスト	→	使用する座標のシステム

SVG_SET_PATTERN_UNITS コマンドは、patternObject で指定されたパターンの x、y、width、そして height 属性の座標システムを設定するために使用します。patternObject がパターンでない場合、エラーが生成されます。

sysCoord 引数は使用されるシステムを指定します。"userSpaceOnUse" または "objectBoundingBox" のいずれかを指定します。そうでなければエラーが生成されます。

参照 : http://www.w3.org/TR/SVG/pservers.html#Patterns

テキスト

SVG_APPEND_TEXT_TO_TEXTAREA

SVG_APPEND_TEXT_TO_TEXTAREA (svgObject; addedText)

引数	型		説明
svgObject	SVG_Ref	→	要素テキスト参照
addedText	テキスト	→	追加するテキスト

SVG_APPEND_TEXT_TO_TEXTAREA コマンドを使用して、svgObject で指定したテキストオブジェクトの内容にテキストを追加します。svgObject が "textArea" オブジェクトでない場合、エラーが生成されます。

改行は自動で "<tbreak/>" 要素に置き換えられます。

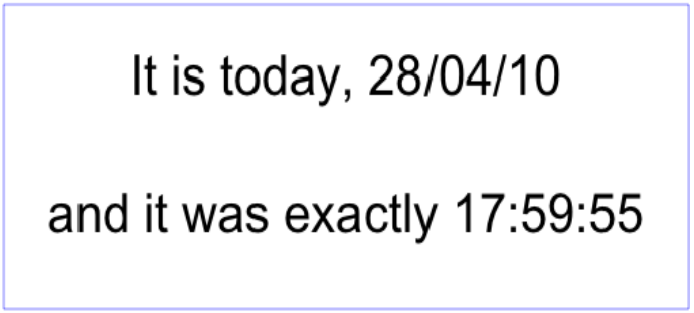
▶ テキストを追加

```
//'rect' 要素を使用して外枠を表示
$Dom_rect:=SVG_New_rect ( $Dom_SVG;10;10;500;200;0;0;"blue:50";
                                                    "none")

// テキストを作成
$Dom_text:=SVG_New_textArea ($Dom_SVG;"It is today, ";10;30;500;200;
                                                    "'Arial'";36;0;3)

// 日付と 2 つの改行を追加
SVG_APPEND_TEXT_TO_TEXTAREA ($Dom_text; String(Current date)+
"\r\r")

// 時刻を追加
SVG_APPEND_TEXT_TO_TEXTAREA ($Dom_text;"and it was exactly "+
                                                    String(Current time))
```



SVG_Get_text

SVG_Get_text (svgObject) → 戻り値

引数	型		説明
svgObject	SVG_Ref	→	要素テキスト参照
戻り値	テキスト	←	テキストの内容

SVG_Get_text コマンドは svgObject で指定した要素のテキスト内容を返します。svgObject がテキストオブジェクト参照 ('text', 'textArea' または 'tspan') でない場合、エラーが生成されます。

textArea オブジェクトの場合、<tbreak/> 要素は CR に変換されます。

SVG_SET_TEXT_KERNING

SVG_SET_TEXT_KERNING (svgObject; kerning {; unit })

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
kerning	Real	→	文字詰め
unit	テキスト	→	値の単位

SVG_Get_text コマンドは 、svgObject で指定したテキストオブジェクトの文字詰めを変更するために使用します。svgObject が SVG テキストオブジェクトでない場合、エラーが生成されます。

オプションの unit 引数を使用して文字詰め値の単位を指定できます。デフォルトは "%" です。

kerning が -1 の場合、文字詰め値は 'auto' に設定されます。

Note: Windows では、左から右と上から下のテキスト (右から左へのテキストは無効)、および 'text' と 'tspan' 要素にに実装が制限されています。Mac OS のサポートに制限はありません。

- さまざまな文字詰めの例 :

```
// 参照
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;2)
```

```
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0)
```

Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !
Hello world !

参照 : http://www.w3.org/TR/SVG/text.html#KerningProperty

SVG_SET_TEXT_LETTER_SPACING

SVG_SET_TEXT_LETTER_SPACING (svgObject; spacing {; unit})

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
spacing	Real	→	文字間隔
unit	テキスト	→	値の単位

SVG_SET_TEXT_LETTER_SPACING コマンドは svgObject で指定されたテキストオブジェクトに対し、'kerning' 属性による間隔に加えて、文字の間隔を変更するために使用します。svgObject が SVG テキストオブジェクトでない場合、エラーが生成されます。

オプションの unit 引数を使用して文字詰め値の単位を指定できます。デフォルトは "%" です。

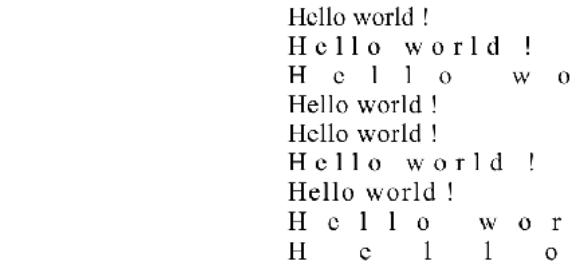
spacing が -1 の場合、間隔値は 'normal' に設定されます。

▶ さまざまな間隔の例 :

```
// 参照
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;120;"";36)
```

```
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"em")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"px")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"pt")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"pc")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"mm")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"cm")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"in")
```



参照 : http://www.w3.org/TR/SVG/text.html#LetterSpacingProperty

SVG_SET_TEXT_RENDERING

SVG_SET_TEXT_RENDERING (svgObject; rendering)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
rendering	テキスト	→	レンダリングの値

SVG_SET_TEXT_RENDERING コマンドは svgObject で指定したテキストオブジェクトのテキストレンダリングに関するトレードオフを設定するために使用します。svgObject が SVG テキストオブジェクトでない場合、エラーが生成されます。

rendering 引数には以下の値を指定できます : "auto"、"optimizeSpeed"、"optimizeLegibility"、"geometricPrecision"、または "inherit"。これ以外の場合エラーが生成されます。

参照 : http://www.w3.org/TR/2001/REC-SVG-20010904/painting.html#TextRenderingProperty

SVG_SET_TEXT_WRITING_MODE

SVG_SET_TEXT_WRITING_MODE (svgObject; writingMode)

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
writingMode	テキスト	→	テキストの方向

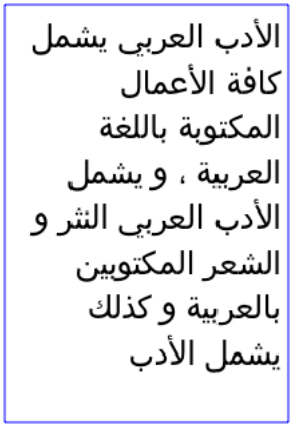
SVG_SET_TEXT_WRITING_MODE コマンドは svgObject で指定したテキストオブジェクトの方向が、左から右、右から左、または上から下かを指定するために使用します。svgObject が SVG テキストオブジェクトでない場合、エラーが生成されます。

writingMode 引数には以下のいずれかの値を渡します : "lr-tb"、"rl-tb"、"tb-rl"、"lr"、"rl"、"tb"、または "inherit "。これ以外の場合エラーが生成されます。

▶ 右から左に設定する :

```
// フレーム
SVG_New_rect ($Dom_SVG;5;5;210;310;0;0;"blue";"none")

// テキスト
$Dom_text:=SVG_New_textArea ($Dom_SVG;$Txt_sample;10;10;200;
                                    300;"Baghdad 'Arial Unicode
MS'";25)
SVG_SET_TEXT_WRITING_MODE($Dom_text;"rl")
```



SVG_SET_TEXTAREA_TEXT

SVG_SET_TEXTAREA_TEXT (svgObject ; theText)

引数	型		説明
svgObject	SVG_Ref	→	要素テキスト参照
theText	テキスト	→	設定するテキスト

SVG_SET_TEXTAREA_TEXT コマンドは、svgObject で指定されたテキストオブジェクトのテキスト内容を設定 / 置換するために使用します。svgObject が "textArea" オブジェクトでない場合、エラーが生成されます。

改行文字は自動で "<tbreak/>" 要素に置き換えられます。

更新されたコマンド

SVG_New_textArea

SVG_New_textArea コマンドは <tbreak/> 要素により返される行を置換するようになりました。

参照 : http://www.w3.org/TR/SVGTiny12/text.html#tbreakElement

SVG_SET_FONT_FAMILY

SVG_SET_FONT_FAMILY (svgObject; font1{ ;...; fontN})

引数	型		説明
svgObject	SVG_Ref	→	SVG 要素参照
font1...N	文字列	→	フォント名

SVG_SET_FONT_FAMILY コマンドは 1 つ以上のフォント名をサポートするようになりました。複数の名称を渡すと、コマンドは自動でフォントファミリー名のリストまたは汎用のファミリー名を構築します。

- 複数のフォント名を渡す :

```
SVG_SET_FONT_FAMILY(svgObject ; "Lucida grande" ; "Sans-serif")
// 以下のリストを構築します : "'Lucida grande', 'Sans-serif'"
```

ユーティリティ

SVG_ABOUT

SVG_ABOUT

引数	型	説明
このコマンドは引数を必要としません。		

SVG_ABOUT コマンドは 4D SVG ロゴとコンポーネントのバージョン番号を示すダイアログを表示します :

SVGTool_SET_VIEWER_CALLBACK

SVGTool_SET_VIEWER_CALLBACK (methodName)

引数	型		説明
methodName	テキスト	→	4D メソッド名

SVGTool_SET_VIEWER_CALLBACK コマンドを使用して、SVGTool_SHOW_IN_VIEWER コマンドを使用して表示されたイメージ上で On Clicked や On Mouse Move イベントが発生したときに呼び出されるプロジェクトメソッド methodName をインストールできます。

このメソッドは、4D の SVG Find element ID by coordinates によって提供される、クリックやマウス移動が発生した要素の ID をテキスト引数に受け取ります。この引数はホストデータベースの methodName プロジェクトメソッド内で C_TEXT($1) コードを使用して宣言しなければなりません。

methodName 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する " プロパティを有効にしなければなりません。

メソッドが存在しないか共有されていない場合、エラー -10508 が生成されます。

更新されたコマンド

SVG_References_array

SVG_References_array (refsArrayPointer) → 戻り値

引数	型		説明
refsArrayPointer	ポインタ	→	ドキュメント参照の配列
戻り値	倍長整数	←	参照の数

SVG_References_array コマンドは見つかった参照の数を返すようになりました。

▶ 要素数だけループする

```
ARRAY TEXT (MyArray ; 0 )
For ($i ; 1 ; SVG_References_array ( ->MyArray ))
   …
End for
```

6 SQL

この章では 4D v12 の SQL エンジンに追加された新機能と変更について説明します :

- SQL を使用したデータベース複製の新機能
- 主キーの直接定義
- OUTER 結合のサポート
- エクスターナルデータベースの利用
- UUID フィールドのサポート
- 新しいコマンドと変更されたコマンド

Note: コードエディタ中での新しい SQL 文の利用が、マクロコマンドにより容易になりました。このマクロを利用するためには "Macros.xml" を更新する必要があります。この点に関する情報は、17 ページの "Macros.xml ファイルの更新 " を参照してください。

SQL による複製

4D v12 では SQL を使用して 2 つ以上の 4D データベース間でデータを同期する新しいメカニズムを提供します。このメカニズムを使用して 1 つ以上のミラーデータベースをセットアップし、データの可用性を保証することができます。

これは以下のように動作します : ローカルデータベースがリモートソースデータベースのデータをローカルに複製します。更新はローカルデータベースが定期的にリモートデータベースからデータを取得することで実行されます。複製はテーブルレベルで行われます。リモートデータベーステーブルのデータをローカルデータベースのテーブルに複製します。

これはスタンプおよび新しい SQL コマンドの追加によって可能となりました。
新しいテーブルプロパティを使用してリモートおよびローカルデータベースで複製メカニズムを有効にできます。ローカル側では新しい SQL REPLICATE コマンドを使用してリモートデータベースのテーブルからデータを取得し、このデータをローカルデータベースのテーブルに統合します。最後に新しい SQL SYNCHRONIZE コマンドを使用すれば 2 つのテーブルを同期できます。

新しい仮想フィールド

4D データベースのそれぞれのテーブルに 3 つの新しい " 仮想 " フィールド、__ROW_ID、__ROW_STAMP、そして __ROW_ACTION が割り当てられることがあります。これらのフィールドは特別なプロパティを持つため、" 標準 " のフィールドと区別するために " 仮想 " と呼ばれます。これらのフィールドは自動で値が割り当てられ、ユーザーは値を読むことはできますが書き込むことはできず、そしてデータベースのシステムテーブルには現れません。以下の表はこれらのフィールドと利用モードを説明しています :

仮想フィールド	型	内容	利用
__ROW_ID	Int32	レコードの ID	REPLICATE や SYNCHRONIZE を除くすべての SQL ステートメント
__ROW_STAMP	Int64	レコード複製情報	すべての SQL ステートメント
__ROW_ACTION	Int16	レコードに対して実行されたアクション : 1 = 追加または更新 2 = 削除	REPLICATE または SYNCHRONIZE のみ

複製メカニズムが有効になると、レコードが作成、更新、削除されるたびに対応する情報が自動でこのレコードの仮想フィールドに保存されます。

複製を有効にする

複製を可能にするメカニズムはデフォルトで無効になっています。複製や同期を行うテーブルごとに、リモートおよびローカルのデータベース両方で、明示的にこれを有効にしなければなりません。

メカニズムを有効にしても複製自身が実行されるわけではない点に留意してください。データが実際にローカルまたは同期データベースに複製されるためには、新しい REPLICATE や SYNCHRONIZE コマンドを使用しなければなりません。

(ローカルおよびリモートデータベース上で) 内部的な複製メカニズムをテーブル毎に有効にするためには、テーブルインスペクタ内の新しい**複製を有効にする**テーブルプロパティを使用します :

複製メカニズムが動作するようにするためには、リモートおよびローカルデータベース側でテーブルの主キーを指定しなければなりません。このキーはストラクチャエディタ (23 ページの " 主キーの設定 " 参照) または SQL コマンド (282 ページの " 列作成時に主キーを指定する " 参照) で指定できます。

このオプションがチェックされると、4D はテーブルのレコードを複製するために必要な情報 (特にテーブルの主キーに基づき) を生成します。この情報は仮想 __ROW_STAMP と __ROW_ACTION フィールドに格納されます。

Note: SQL CREATE TABLE と ALTER TABLE コマンドを使用して複製情報の作成を有効にしたり無効にしたりできます。これを行うために新しい ENABLE REPLICATE と DISABLE REPLICATE キーワードを使用できます。詳細は 300 ページの " 変更された SQL コマンド " にあるこれらのコマンドの詳細説明を参照してください。

ローカルデータベース側の更新

データベースごとのテーブルごとに複製メカニズムが有効になると、ローカルデータベースから新しい SQL REPLICATE コマンドを使用できます。このコマンドは 291 ページの " 新しい SQL コマンド " で説明しています。

列作成時に主キーを指定する

テーブル作成時 (CREATE TABLE コマンドを使用) あるいは列追加時 (ALTER TABLE コマンドを使用) に直接主キー (PRIMARY KEY) を指定することができます。以前のバージョンの 4D では既存の列に ALTER TABLE を使用する必要がありました。

このためこれらのコマンドのシンタックスが変更され、列作成時に PRIMARY KEY キーワードが受け入れられるようになりました。

詳細はこれらのコマンドの説明を参照してください。

Note: 4D ではストラクチャエディタを使用して主キーを管理することもできます。詳細は 23 ページの " 主キーの設定 " を参照してください。

結合のサポート

4D v12 の SQL エンジンは結合のサポートを拡張し、特に OUTER 結合が実行できるようになりました。

結合操作は INNER または OUTER、そして明示または暗黙です。SELECT コマンドで実行される暗黙の INNER 結合はすでに 4D v11 SQL でサポートされていました (後述参照)。4D v12 では SQL **JOIN** キーワードを使用して生成される明示的な INNER 結合と OUTER 結合が使用できます。

Note: 4D SQL エンジンの結合の現在の実装には以下が含まれていません :
- NATURAL 結合
- INNER 結合の USING 構成

概要

結合操作は 2 つ以上のテーブル間のレコードの接続を作成し、結果を結合と呼ばれる新しいテーブルに一体化させるために使用します。

結合条件を指定した SELECT ステートメントを使用して結合を生成します。明示的な結合では、これらの条件を複合にすることができますが、常に結合に含まれる列間の等価比較に基づくものでなければなりません。例えば >= 演算子を明示的な結合条件に使用することはできません。
暗黙的な結合ではすべてのタイプの比較を使用できます。
内部的に、等価比較は 4D エンジンにより直接実行されるので、素早く実行されます。

Note: 通常データベースエンジンでは、テーブルの順番は検索の間に指定された順番です。しかし結合を使用する際、テーブルの順番はテーブルリストで決定されます。以下の例題で :
SELECT * FROM T1 RIGHT OUTER JOIN T2 ON T2.depID = T1.depID;
テーブルの順番は (テーブルリストに現れる通り) T1 そして T2 です。(結合条件に現れる順である) T2 そして T1 ではありません。

明示的な INNER 結合

INNER 結合は 2 つの列間で一致する値を探す比較に基づきます。例えば以下の 3 つのテーブルがあるとき :

- **Employees**

name	depID	cityID
Alan	10	30
Anne	11	39
Bernard	10	33
Mark	12	35
Martin	15	30
Philip	NULL	33
Thomas	10	NULL

- **Departments**

depID	depName
10	Program
11	Engineering
NULL	Marketing
12	Development
13	Quality

- **Cities**

cityID	cityName
30	Paris
33	New York
NULL	Berlin

Note: この例のストラクチャはこのセクションを通して使用されます。

まず (すでに 4D v11 SQL でサポートされている) 暗黙的な INNER 結合の例は以下の通りです :

```
SELECT *
   FROM employees, departments
   WHERE employees.DepID = departments.DepID;
```

4D v12 では、JOIN キーワードを使用して明示的な INNER 結合を指定できます :

```
SELECT *
   FROM employees
   INNER JOIN departments
      ON employees.DepID = departments.DepID;
```

このクエリを以下のように 4D コードに挿入できます :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aDepID;0)
Begin SQL
  SELECT *
  FROM employees
       INNER JOIN departments
          ON employees.depID = departments.depID
       INTO :aName, :aEmpDepID, :aDepID, :aDepName;
End SQL
```

この結合結果は以下のとおりとなります :

aName	aEmpDepID	aDepID	aDepName
Alan	10	10	Program
Anne	11	11	Engineering
Bernard	10	10	Program
Mark	12	12	Development
Thomas	10	10	Program

従業員名が Philip と Martin、また部署名が Marketing と Quality は結合の結果に現れません。なぜなら :

- Philip に部署は割り当てられていません (NULL 値)。
- Martin の部署 ID は Departments テーブルに存在しません。
- Quality 部署に割り当てられた従業員はいません (ID 13)。
- Marketing 部署に ID が割り当てられていません (NULL 値)。

CROSS 結合

CROSS 結合あるいはデカルト積は WHERE も ON 句も指定されていない INNER 結合です。これはあるテーブルのそれぞれの行を他のテーブルのそれぞれの行に関連付けるものです。
CROSS 結合の結果はテーブルのデカルト積であり、2 つのテーブルの行数の積が総行数として含まれます。この積は両テーブルの行を結合した場合のすべての組み合わせを表します。

以下のシンタックスはすべて同等です :

```
SELECT * FROM T1 INNER JOIN T2
SELECT * FROM T1, T2
SELECT * FROM T1 CROSS JOIN T2;
```

以下は CROSS 結合を行う 4D コードの例です :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aDepID;0)
Begin SQL
  SELECT *
  FROM employees CROSS JOIN departments
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
End SQL
```

以下は例題データベースにおけるこの結合の結果です :

aName	aEmpDepID	aDepID	aDepName
Alan	10	10	Program
Anne	11	10	Program
Bernard	10	10	Program
Mark	12	10	Program
Martin	15	10	Program
Philip	NULL	10	Program
Thomas	10	10	Program
Alan	10	11	Engineering
Anne	11	11	Engineering
Bernard	10	11	Engineering
Mark	12	11	Engineering
Martin	15	11	Engineering
Philip	NULL	11	Engineering
Thomas	10	11	Engineering
Alan	10	NULL	Marketing
Anne	11	NULL	Marketing
Bernard	10	NULL	Marketing
Mark	12	NULL	Marketing
Martin	15	NULL	Marketing
Philip	NULL	NULL	Marketing
Thomas	10	NULL	Marketing
Alan	10	12	Development
Anne	11	12	Development
Bernard	10	12	Development
Mark	12	12	Development
Martin	15	12	Development
Philip	NULL	12	Development

Thomas	10	12	Development
Alan	10	13	Quality
Anne	11	13	Quality
Bernard	10	13	Quality
Mark	12	13	Quality
Martin	15	13	Quality
Philip	NULL	13	Quality
Thomas	10	13	Quality

Note: パフォーマンスの理由から、CROSS 結合は注意して使用しなければなりません。

OUTER 結合

4D v12 で OUTER 結合を生成できるようになりました。OUTER 結合では、結合させるテーブルの行間で値が一致する必要はありません。結果のテーブルには、たとえ一致する行がなくても、テーブル (または少なくとも 1 つの結合されたテーブル) のすべての行が含まれます。これは、たとえ異なる結合されたテーブル間でぎょうが完全に値を持っていない場合でも、あるテーブルのすべての情報が使用できることを意味します。

OUTER 結合には LEFT、RIGHT、そして FULL キーワードで指定される 3 つのタイプがあります。LEFT と RIGHT は (JOIN キーワードの左あるいは右どちらに位置するかで) すべてのデータが処理されるテーブルを示すために使用されます。FULL は両側の OUTER 結合を示します。

Note: 4D v12 では明示的な OUTER 結合だけがサポートされます。

LEFT OUTER 結合

LEFT OUTER 結合 (または LEFT 結合) の結果には、たとえ結合条件に一致するレコードがキーワードの右側のテーブルにない場合でも、常にキーワードの左にあるテーブルのすべてのレコードが含まれます。これは検索が右側のテーブル内で一致する行を見つけることができない左テーブルの各行についても、結合はそれらを結果に含めることを意味します。しかしこの場合右テーブルの各列には NULL 値が入ります。言い換えれば LEFT OUTER 結合は左側のテーブルのすべての行と、結合条件に一致する右テーブルの行 (または一致しない場合 NULL) を作成します。左テーブルの 1 つの行に対し、結合述部に一致する右テーブルの行が複数ある場合、左テーブルの値は右テーブルの各行に対し繰り返される点に留意してください。

以下は LEFT OUTER 結合を行う 4D コードの例です :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aDepID;0)
```

```
Begin SQL
   SELECT *
      FROM employees
      LEFT OUTER JOIN departments
         ON employees.DepID = departments.DepID;
         INTO :aName, :aEmpDepID, :aDepID, :aDepName;
End SQL
```

以下は例題データベースでこの結合を実行したときの結果です (赤は追加の行を示します):

aName	aEmpDepID	aDepID	aDepName
Alan	10	10	Program
Anne	11	11	Engineering
Bernard	10	10	Program
Mark	12	12	Development
Thomas	10	10	Program
Martin	15	NULL	NULL
Philip	NULL	NULL	NULL

RIGHT OUTER 結合

RIGHT OUTER 結合は完全に LEFT OUTER 結合の反対です (前の段落参照)。たとえ結合条件に一致するレコードが左のテーブルに見つからない場合でも、結果には常に JOIN キーワードの右側のテーブルのすべてのレコードが含まれます。

以下は RIGHT OUTER 結合を行う 4D コードの例です :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aDepID;0)
Begin SQL
   SELECT *
      FROM employees
      RIGHT OUTER JOIN departments
         ON employees.DepID = departments.DepID;
         INTO :aName, :aEmpDepID, :aDepID, :aDepName;
End SQL
```

以下は例題データベースでこの結合を実行したときの結果です (赤は追加の行を示します):

aName	aEmpDepID	aDepID	aDepName
Alan	10	10	Program
Anne	11	11	Engineering
Bernard	10	10	Program
Mark	12	12	Development
Thomas	10	10	Program

NULL	NULL	NULL	Marketing
NULL	NULL	13	Quality

FULL OUTER 結合

FULL OUTER 結合は単純に LEFT OUTER 結合と RIGHT OUTER 結合の結果を組み合わせたものです。結果の結合テーブルには左および右のレコードがすべて含まれます。それぞれの側で失われたフィールドは NULL 値になります。

以下は FULL OUTER 結合を行う 4D コードの例です :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aDepID;0)
Begin SQL
  SELECT *
    FROM employees
    FULL OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
End SQL
```

以下は例題データベースでこの結合を実行したときの結果です (赤は追加の行を示します):

aName	aEmpDepID	aDepID	aDepName
Alan	10	10	Program
Anne	11	11	Engineering
Bernard	10	10	Program
Mark	12	12	Development
Thomas	10	10	Program
Martin	15	NULL	NULL
Philip	NULL	NULL	NULL
NULL	NULL	NULL	Marketing
NULL	NULL	13	Quality

1 つのステートメント内で複数の結合

1 つの SELECT ステートメント内で複数の結合を組み合わせることができます。また暗黙的または明示的な INNER 結合と明示的な OUTER 結合を組み合わせることもできます。

以下は複数の結合を行う 4D コードの例です :

```
ARRAY TEXT (aName;0)
ARRAY TEXT (aDepName;0)
ARRAY TEXT (aCityName;0)
ARRAY INTEGER (aEmpDepID;0
ARRAY INTEGER (aEmpCityID;0)
```

```
ARRAY INTEGER (aDepID;0)
ARRAY INTEGER (aCityID;0)
Begin SQL
  SELECT *
    FROM employees
    FULL OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
    FROM (employees RIGHT OUTER JOIN departments
          ON employees.depID = departments.depID)
          LEFT OUTER JOIN cities
          ON employees.cityID = cities.cityID
    INTO :aName, :aEmpDepID, :aEmpCityID, :aDepID, :aDepName;
                                          :aCityID, :aCityName;
End SQL
```

以下は例題データベースでこの結合を実行したときの結果です :

aName	aEmpDepID	aEmpCityID	aDepID	aDepName	aCityID	aCityName
Alan	10	30	10	Program	30	Paris
Anne	11	39	11	Engineering	0	
Bernard	10	33	10	Program	33	New York
Mark	12	35	12	Developmen t	0	
Thomas	10	NULL	10	Program	0	

エクスターナルデータベースの利用

4D v12 を使用して SQL ランゲージ経由で、" エクスターナルデータベース " を作成、更新、および利用できます。

エクスターナルデータベースとはメインの 4D データベースから独立していますが、メインのデータベースから使用できる 4D データベースです。エクスターナルデータベースの利用は、このデータベースを一時的にカレントデータベース、言い換えれば 4D により実行される SQL クエリのターゲットデータベースとして指定することを意味します。

エクスターナルデータベースは新しい SQL コマンド CREATE DATABASE を使用してメインデータベースから作成できます。データベースはディスク上に標準ファイル (.4db と .4dd ファイル) として格納されます。メインデータベースからいくつでもエクスターナルデータベースを作成できます。
エクスターナルデータベースを作成したら、それを SQL の USE DATABASE コマンドを使用してカレントデータベースとして指定できます。そして標準の SQL コマンド (CREATE TABLE や ALTER TABLE 等) でス

トラクチャを変更し、データを格納できます。新しい DATABASE_PATH 関数を使用していつでもカレントデータベースを知ることができます。

エクスターナルデータベースの主たる関心は、4D コンポーネントから作成および処理が可能であるということです。この新しい機能により、必要に応じてテーブルやフィールドを作成するコンポーネントが開発できるようになります。

Note: エクスターナルデータベースは標準の 4D データベースです。4D や 4D Server アプリケーションでそれをメインデータベースとして開き、操作することができます。逆に、標準の 4D データベースをエクスターナルデータベースとして使用することができます。しかしエクスターナルデータベースに (Designer パスワードを設定することで) アクセス管理システムを有効化することはできません。有効にすると、SQL の USE DATABASE を使用してアクセスすることができなくなってしまいます。

UUID フィールドのサポート

4D v12 では UUID の生成と格納がサポートされています (19 ページの "UUID のサポート " 参照)。

UUID フィールドの作成

ストラクチャエディタでは、文字フィールドに特別なプロパティを指定することで、UUID フィールドを設定します (21 ページの "UUID フォーマット " 参照)。
4D SQL では、UUID フィールドを新しいデータタイプ **UUID** で定義します。このデータタイプはフィールドを作成あるいは更新するコマンドで使用できます。例えば :

```
Begin SQL
  CREATE TABLE ACTORS_FAN
     (ID       UUID,
     Name   VARCHAR(30));
End SQL
```

または :

```
Begin SQL
  CREATE TABLE ACTORS_FAN
     (Name  VARCHAR(30));
  ALTER TABLE ACTORS_FAN
     ADD ID UUID;
End SQL
```

UUID を自動で生成する

4D では、" 自動 UUID" プロパティを使用することで、レコードの UUID フィールドに自動で UUID を生成することができます (21 ページの " 自動 UUID" 参照)。
4D SQL では、このプロパティは新しいキーワード、**AUTO_GENERATE** でサポートされています。このキーワードはフィールドを作成あるいは更新するコマンドで使用できます。例えば :

```
Begin SQL
   CREATE TABLE ACTORS_FAN
      (ID       UUID AUTO_GENERATE,
      Name   VARCHAR(30));
End SQL
```

新しい SQL コマンド

4D v12 でいくつかの新しい SQL コマンドが追加されました :

- エクスターナルデータベースの管理に関連するコマンド : CREATE DATABASE, USE DATABASE
- 新しい実装に関連するコマンド : ALTER DATABASE, REPLICATE と SYNCHRONIZE

CREATE DATABASE

CREATE DATABASE [**IF NOT EXISTS**] **DATAFILE**<Complete pathname>

新しい CREATE DATABASE コマンドを使用して新しいエクスターナルデータベース (.4db と .4dd ファイル) を指定した場所に作成できます。

IF NOT EXISTS 制約が渡されていると、指定した場所に同じ名前のデータベースが存在すれば、データベースは作成されずエラーも生成されません。
IF NOT EXISTS 制約が渡されていない場合、指定した場所に同じ名前のデータベースが既に存在すれば、データベースは作成されず、" データベースは既に存在します。CREATE DATABASE コマンドの実行に失敗しました。" エラーメッセージが表示されます。

DATAFILE 句に新しいエクスターナルデータベースの完全名 (完全パス名 + 名前) を指定します。ストラクチャファイルの名前を渡さなければなりません。プログラムは (指定されていなければ) 自動で ".4db" 拡張子をファイルに追加し、データファイルを作成します。パス名は POSIX シンタックスあるいはシステムシンタックスで指定します。また絶対、あるいはメインの 4D データベースのストラクチャファイルからの相対で指定できます。

- POSIX シンタックス (URL 型): 使用するプラットフォームに関係なく、フォルダ名はスラッシュ ("/") で区切られます。
  例 : .../extdatabases/myDB.4db
- システムシンタックス: 現在のプラットフォームに基づくシンタックスのパス名。
  例 :
  - (Mac OS) Disk:Applications:myserv:extdatabases:myDB.4db
  - (Windows) C:\Applications\myserv\extdatabases\myDB.4db

CREATE DATABASE コマンドの実行に成功しても、作成された新しいデータベースは自動ではカレントデータベースにはなりません。これを行うには、新しい USE DATABASE コマンドを使用して明示的にカレントデータベースとして宣言しなければなりません。

▶ C:/MyDatabase/ にエクスターナルデータベース ExternalDB.4DB と ExternalDB.4DD を作成する :

```
Begin SQL
   CREATE DATABASE IF NOT EXISTS DATAFILE 'C:/MyDatabase/ExternalDB';
End SQL
```

▶ メインデータベースのストラクチャファイルと同階層にエクスターナルデータベース TestDB.4DB と TestDB.4DD を作成する :

```
Begin SQL
   CREATE DATABASE IF NOT EXISTS DATAFILE 'TestDB';
End SQL
```

▶ ユーザが指定した場所にエクスターナルデータベース External.4DB と External.4DD を作成する :

```
C_TEXT($path)
$path:=Select folder("Destination folder of external database:")
$path:= $path+"External"
Begin SQL
   CREATE DATABASE DATAFILE <<$path>>;
End SQL
```

参照 : USE DATABASE

USE DATABASE

USE [LOCAL | REMOTE] DATABASE
{DATAFILE <Complete pathname> | **SQL_INTERNAL | DEFAULT}**
[AUTO_CLOSE]

新しい USE DATABASE コマンドはエクスターナルデータベース (289 ページの " エクスターナルデータベースの利用 " 参照) をカレントデータベー

スとして指定するために、いいかえればカレントプロセスで次の SQL リクエストの送信先を指定するために使用します。すべてのタイプの SQL クエリが関連します : Begin SQL/End SQL 構造に含まれるクエリ、SQL EXECUTE または SQL EXECUTE SCRIPT コマンド等

シングルユーザで動作しているとき、エクスターナルデータベースは 4D と同じマシン上になければなりません。
リモートモードの場合、エクスターナルデータベースはローカルマシンまたは 4D Server マシンに存在できます。

4D をリモートモードで使用している場合、**REMOTE** キーワードを使用して 4D Server 上にあるエクスターナルデータベースを指定できます。
セキュリティのため、このメカニズムはネイティブなリモート接続のみ、言い換えると、4D Server に接続したリモート 4D のコンテキストのみでで動作します。ODBC やパススルー接続では使用できません。
キーワードが指定されない場合、**LOCAL** オプションがデフォルトで使用されます。4D をローカルモードで使用している場合、**REMOTE** および **LOCAL** キーワードは 無視されます。接続は常にローカルです。

使用するエクスターナルデータベースを指定するには、**DATAFILE** 句に完全パス (パス名 + 名前) を渡します。パス名は POSIX シンタックスあるいはシステムシンタックスで指定できます。また絶対パスあるいはメインの 4D データベースからの相対パスで表すことができます。
リモートモードで **REMOTE** キーワードが渡されると、この引数はサーバマシンからのデータベースパスを指定します。これが省略されるか **LOCAL** キーワードが渡されると、この引数はローカル 4D マシンのデータベースを指定します。

アクセス制御が有効化されていない (Designer にパスワードが設定されていない) 、有効なエクスターナルデータベースを指定しなければなりません。そうでなければエラーが生成されます。

メインデータベースをカレントデータベースにリセットするには、コマンドに **SQL_INTERNAL** または **DEFAULT** キーワードを渡して実行します。

使用後、つまりカレントデータベースを変更する際、物理的にエクスターナルデータベースを閉じたい場合は **AUTO_CLOSE** を渡します。実際のところ、エクスターナルデータベースのオープン処理は時間のかかる処理です。最適化のため、4D はユーザセッション中に開かれたエクスターナルデータベースに関する情報をメモリに保持します。この情報は 4D アプリケーションが起動されている間保持されます。このため同じエクスターナルデータベースを引き続き開く際には、素早く処理が行われます。しかし最初にエクスターナルデータベースを開いたアプリケーションが読み / 書きモードでアクセスしているため、複数の 4D アプリ

ケーションがエクスターナルデータベースを共有することができません。複数の 4D アプリケーションが同じエクスターナルデータベースを同時に使用できるようにするためには、**AUTO_CLOSE** キーワードを渡すことで、利用後、エクスターナルデータベースを物理的に開放するようにします。

この制限は同じアプリケーションのプロセスには適用されません。一つのアプリケーションの異なるプロセスは、閉じることを強制しなくても、常に同じエクスターナルデータベースアクセスできます。
複数のプロセスが同じエクスターナルデータベースを使用する際、**AUTO_CLOSE** が指定されている場合でも、エクスターナルデータベースを使用する最後のプロセスが終了したのちに、エクスターナルデータベースが物理的に解放される点に留意してください。エクスターナルデータベースをアプリケーション間で共有したり削除したりするような処理を行う際は、この動作を考慮に入れなければなりません。

- エクスターナルデータベースにリクエストを行い、メインデータベースに戻る :

```
Begin SQL
   USE DATABASE DATAFILE 'C:/MyDatabase/Names'
   SELECT Name FROM emp INTO :tNames1
   USE DATABASE SQL_INTERNAL
End SQL
```

参照 : CREATE DATABASE, DATABASE_PATH

ALTER DATABASE

ALTER DATABASE {ENABLE | DISABLE} {INDEXES | CONSTRAINTS}

新しい ALTER DATABASE を使用してカレントセッションのカレントデータベースの SQL オプションを有効あるいは無効にできます。

このコマンドは、多くのリソースを必要とするような特定の処理の動作を向上させるため、一時的に SQL オプションを無効にするために使用します。例えば大量のデータを読み込む前にインデックスや制約を無効にすると、読み込み時間を大きく削減できます。制約には主キーや外部キーに加え、重複不可やヌル属性も含まれる点に留意してください。

警告 : このコマンドはよく注意して使用されなければなりません。読み込み時にトリガを無効にすればデータの不整合や破壊を招く恐れがあります。このリスクを十分考慮に入れ、必要に応じてデータの整合性を確保する方法を実装しなければなりません。

ALTER DATABASE は データベース全体に適用されます。言い換えればユーザがオプションを変更すると、その変更はデータベースの他のユーザにも適用されます。

▶ 一時的にすべての SQL オプションを無効にして読み込みを行う例 :

```
Begin SQL
   ALTER DATABASE DISABLE INDEXES;
   ALTER DATABASE DISABLE CONSTRAINTS;
End SQL
SQL EXECUTE SCRIPT ("C:\Exported_data\Export.sql", SQL On error continue)
Begin SQL
   ALTER DATABASE ENABLE INDEXES;
   ALTER DATABASE ENABLE CONSTRAINTS;
End SQL
```

REPLICATE

REPLICATE replicated_list
FROM table_reference
[**WHERE** search_condition]
[**LIMIT** {int_number | 4d_language_reference}]
[**OFFSET** {int_number | 4d_language_reference}]
FOR REMOTE [**STAMP**] {int_number | 4d_language_reference}
[, **LOCAL** [STAMP] {int_number | 4d_language_reference}]
[{**REMOTE OVER LOCAL** | **LOCAL OVER REMOTE**}]
[**LATEST REMOTE** [**STAMP**]: 4d_language_reference]
[, LATEST LOCAL [STAMP]: 4d_language_reference]
INTO {target_list | table_reference(sql_name_1;...;sql_name_N)};

新しい REPLICATE コマンドを使用して、データベース A のテーブルのデータをデータベース B のテーブルに複製できます。用語としてコマンドが実行されるデータベースを " ローカルデータベース "、データ複製元のデータベースを " リモートデータベース " 呼びます。

このコマンドはデータベースの複製システムのフレームワークでのみ使用することができます。システムが動作するためには、複製がローカルデータベースとリモートデータベースで有効にされ、関連するそれぞれのテーブルが主キーを持たなくてはなりません。このシステムに関する詳細は 279 ページの "SQL による複製 " を参照してください。

Note: 完全な同期システムを実装したい場合は新しい SYNCHRONIZE コマンドを参照してください。

replicated_list にはコンマで区切った (仮想あるいは標準の) フィールドリストを渡します。フィールドはリモートデータベースの table_reference テーブルに属していなければなりません。

FROM 句には、replicated_list フィールドのデータを複製するリモートデータベースのテーブルを指定する、table_reference タイプの引数が続かなくてはなりません。

Note: リモートデータベースの仮想フィールドはローカルデータベースの配列にのみ格納できます。

リモートデータベース側 オプションの WHERE 句を使用して、リモートデータベースのテーブルのレコードに予備的なフィルタを適用することができます。これにより search_condition に合致するレコードのみがこのコマンドの処理対象となります。

そして 4D は FOR REMOTE STAMP 句で指定されたすべてのレコードの replicated_list フィールドの値を取り出します。この句に渡すことにできる値は以下のいずれかです :

- **0 より大きい倍長整数値** : この場合、__ROW_STAMP の値がこの値以上のレコードのみが複製されます。
- **0**: この場合、__ROW_STAMP の値が 0 でないすべてのレコードが複製されます。複製を有効にする前に既に存在したレコードは複製の対象とならない点に留意してください (これらのレコードの __ROW_STAM の値は 0 のためです)。
- **-1**: この場合、リモートテーブルのすべてのレコード、言い換えると __ROW_STAMP の値が 0 以上のすべてのレコードが複製されます。前のケースと異なり、複製が有効になる前に既に存在したレコードも含め、すべてのレコードが対象となります。
- **-2**: この場合、(複製が有効になった後に) リモートテーブルから削除された、言い換えると __ROW_ACTION の値が 2 のすべてのレコードが複製されます。

最後にオプションの OFFSET かつ / または LIMIT 句を取得したセレクションに適用できます :

- OFFSET 句が渡されると、セレクションの最初の X レコードが無視されます (X は句に渡された値)。
- LIMIT 句が渡されると、この値は最初の Y レコードにセレクションを制限するために使用されます (Y は句に渡された値)。OFFSET 句も渡されている場合、LIMIT 句は OFFSET 実行後に適用されます。

両方の句が適用されると、結果のセレクションがローカルデータベースに送信されます。

ローカルデータベース側　取り出した値は直接ローカルデータベースの target_list または table_reference テーブルの sql_name で指定した標準フィールドに書き込まれます。target_list 引数は標準のフィールドまたはリモートフィールドと同じ型の配列リストを含むことができます (両方を組み合わせることはできません)。データの格納先がフィールドリストの場合、仮想 __ROW_ACTION フィールドに格納されたアクションに基づき、ターゲットのレコードが自動で作成、更新、または削除されます。

コンフリクト (同じ主キー値) はコンフリクト管理句 (REMOTE OVER LOCAL または LOCAL OVER REMOTE オプション) を使用して解決されます :

- REMOTE OVER LOCAL オプションを渡すかコンフリクト管理句を省略すると、コンフリクトが発生した場合、元のレコード (リモートデータベース) が常にターゲットレコード (ローカルデータベース) を置き換えます。このオプションを使用する場合、LOCAL STAMP は渡しても無視される点に留意してください。
- LOCAL OVER REMOTE オプションを渡すと、コンフリクトが発生した場合、ターゲットレコード (ローカルデータベース) は更新されません。コマンドは LOCAL STAMP を考慮します。LOCAL STAMP 句を使用してローカルテーブル中の複製を、スタンプの値がこの句に渡した値よりも小さいか等しいコンフリクトレコードに制限できます。この句を使用してローカルテーブルに複製されるコンフリクトレコードのセレクションを少なくすることができます。
- LATEST REMOTE STAMP かつ / または LATEST LOCAL STAMP 句を渡すと、4D はリモートおよびローカルテーブルの最新のスタンプの値を、対応する 4d_language_reference 変数に返します。この情報は同期プロシージャの管理をカスタマイズしたい場合に有用です。これらの値は複製処理が完了した直後のスタンプ値に対応します。これらを続く REPLICATE または SYNCHRONIZE ステートメントで使用するとき、それらは REPLICATE コマンドから返される前に自動でインクリメントされるため、開発者がインクリメントする必要はありません。

  複製が正しく実行されると OK システム変数に 1 が設定されます。4D メソッドでこの値をチェックできます。
  複製処理中にエラーが発生すると、処理は最初に発生したエラーで中断されます。最新のソース変数は (指定されていれば) エラーが発生したレコードのスタンプに設定されます。OK システム変数は 0 に設定されます。生成されたエラーは ON ERR CALL コマンドでインストールされるエラー処理メソッドでとらえることができます。

Note: REPLICATE コマンドで実行される処理はデータ整合性制約を考慮に入れません。これは例えば外部キー、重複不可等を管理するルールが検証されないことを意味します。受信したデータがデータ整合性を壊す可能性がある場合、複製処理終了後にデータを検証する必要があります。もっとも簡単な方法は 4D または SQL ランゲージを使用して更新されるレコードをロックすることです。

SYNCHRONIZE

SYNCHRONIZE
[LOCAL] TABLE table_reference (sql_name_1;...;sql_name_N)
WITH
[REMOTE] TABLE table_reference (sql_name_1;...;sql_name_N)
FOR REMOTE [**STAMP**] {int_number | 4d_language_reference}
LOCAL [STAMP] {int_number | 4d_language_reference}
{**REMOTE OVER LOCAL** | **LOCAL OVER REMOTE**}
LATEST REMOTE [**STAMP**] 4d_language_reference
LATEST LOCAL [STAMP] 4d_language_reference;

新しい SYNCHRONIZE コマンドを使用して異なる 2 つの 4D SQL サーバ上に存在する 2 つのテーブルを同期できます。いずれかのテーブルに対して行われた変更は他方のテーブルに対しても実行されます。コマンドを実行する 4D SQL サーバはローカルサーバと呼ばれ、他方のサーバはリモートサーバと呼ばれます。

Note: SYNCHRONIZE コマンドは REPLICATE コマンドを内部的に 2 回呼び出したものです。一回目の呼び出しでリモートサーバからのデータをローカルサーバにレプリケートし、二回目の呼び出しでローカルサーバのデータをリモートサーバにレプリケートします。なので同期されるテーブルは複製用に設定されていなければなりません。
- テーブルは主キーを持っていなければなりません。
- " 複製を有効にする " オプションが各テーブルのインスペクタウィンドウでチェックされていなければなりません。
詳細は REPLICATE コマンドの説明を参照してください。

SYNCHRONIZE コマンドは 4 つのスタンプ 2 つの入力スタンプと 2 つの出力スタンプ (最新の更新) を " 引数 " として受け入れます。入力スタンプはそれぞれのサーバ上での最新の同期の時期を示すために使用されます。この時期はコード化された情報として、継続時間なしで表現されます。出力スタンプは最新の更新直後のそれぞれのサーバ上での更新スタンプの値を返します。この原則により、SYNCHRONIZE コマンドが定期的

に呼ばれるとき、次回の入力スタンプとして、最新の同期の出力スタンプを使用できます。

Note: 入力および出力スタンプは数値として表現され、タイムスタンプとはなりません。これらのスタンプについては REPLICATE コマンドの説明を参照してください。

エラーが発生すると、関連するサーバの出力スタンプには、エラーのもととなったレコードのスタンプが戻されます。エラーが同期以外の原因で引き起こされた場合 (例えばネットワークの問題など)、スタンプは 0 となります。
異なる 2 つのエラーコードがあります。1 つはローカルサイトでの同期エラーを示すもので、もう 1 つはリモートサイトでの同期エラーを示します。
エラーが発生すると、データの状態はローカルサーバのトランザクションの状態に依存します。リモートサーバ上で、同期は常にトランザクション内で実行されるので、処理によりデータが変更されることはありません。しかしローカルサーバ上では、同期処理は開発者の制御下にあります。**Auto-commit Transactions** 環境設定が選択されていない場合、トランザクションの外で処理が実行されます (選択されていればトランザクションコンテキストが自動で作成されます)。
開発者はトランザクションを開始するか決定でき、データの同期後にこのトランザクションを有効にするかキャンセルするかも開発者に任されています。

コンフリクトは、両側で同じレコードが更新されていた場合にどちらのサーバーが優先されるかを明示的に示す、REMOTE OVER LOCAL と LOCAL OVER REMOTE 句を使用して解決されます。実装メカニズムについては REPLICATE コマンドの説明を参照してください。

Note: SYNCHRONIZE コマンドで実行される処理はデータ整合性制約を考慮に入れません。これは例えば外部キー、重複不可等を管理するルールが検証されないことを意味します。受信したデータがデータ整合性を壊す可能性がある場合、複製処理終了後にデータを検証する必要があります。
もっとも簡単な方法は 4D または SQL ランゲージを使用して更新されるレコードをロックすることです。

新しい関数

DATABASE_PATH

DATABASE_PATH()

DATABASE_PATH 関数はカレントデータベースの完全パス名を返します。カレントデータベースは SQL の USE DATABASE コマンドで変更できます。デフォルトでカレントデータベースはメインの 4D データベースです。

▶ カレントのエクスターナルデータベースが TestBase.4DB で、それはフォルダ "C:\MyDatabases" に存在するとします。以下のコードを実行後 :

```
C_TEXT($vCrtDatabasePath)
Begin SQL
  SELECT DATABASE_PATH()
     FROM _USER_SCHEMA
     LIMIT 1
     INTO :$vCrtDatabasePath
End SQL
```

vCrtDatabasePath 変数には "C:\MyDatabases\TestBase.4DB" が格納されます。

参照 : CREATE DATABASE, USE DATABASE

変更された SQL コマンド

この節では 4D の既存の SQL コマンドに対して行われた変更について説明します。変更点は太字で示しています。

CREATE TABLE

CREATE TABLE [IF NOT EXISTS] {sql_name.}sql_name({column_definition |table_constraint} **[PRIMARY KEY]**, ... , {column_definition |table_constraint} **[PRIMARY KEY]**) **[{ENABLE | DISABLE} REPLICATE]**

CREATE TABLE コマンドは 2 つの新しいキーワードを受け入れます :

- PRIMARY KEY: テーブルが作成される際に主キーを指定するために使用します (282 ページの " 列作成時に主キーを指定する " 参照)。

Note: PRIMARY KEY キーワードは 4D バージョン 11.4 より受け入れられます。

- {ENABLE | DISABLE} REPLICATE: テーブルの複製メカニズムを有効あるいは無効にするために使用します (280 ページの " 複製を有効にする " 参照)。

ALTER TABLE

ALTER TABLE sql_name
{ADD column_definition [**PRIMARY KEY**]|
DROP sql_name |
ADD primary_key_definition |
DROP PRIMARY KEY |
ADD foreign_key_definition |
DROP CONSTRAINT sql_name |
[{**ENABLE | DISABLE**} **REPLICATE**] |
SET SCHEMA sql_name}

ALTER TABLE コマンドは 2 つの新しいキーワードを受け入れます :

- PRIMARY KEY: 列を追加する際に主キーを指定するために使用します (282 ページの " 列作成時に主キーを指定する " 参照)。

Note: PRIMARY KEY キーワードは 4D バージョン 11.4 より受け入れられます。

- {ENABLE | DISABLE} REPLICATE: テーブルの複製メカニズムを有効あるいは無効にするために使用します (280 ページの " 複製を有効にする " 参照)。

7 4D Server 管理

64-bit 4D Server について

Windows 版の 4D Server v12 では 32-bit (標準) バージョンおよび 64-bit バージョンを利用できます。64-bit バージョンは 64-bit の Windows OS 上で利用されるものです。

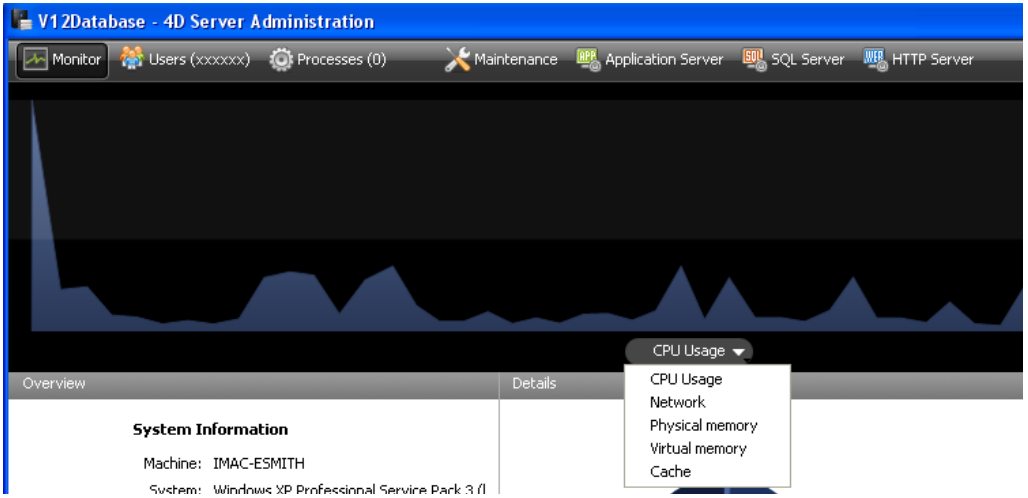
64-bit バージョンの 4D Server は現在開発中であり、ベータテストプログラムが行われています。4D Web サイト (http://www.4d.com/) に接続することにより、このバージョンを評価することができます。近日中に最終公開バージョンとなる予定です。

64-bit バージョンの 4D Server に関する技術情報は、4D のドキュメントサイト (http://doc.4d.com/) で 64-bit 4D Server マニュアルを参照してください。

4D Server 管理ウィンドウ

新しいメモリ情報

4D Server 管理ウィンドウの**モニター**ページに、アプリケーションが使用するメモリに関する情報が追加で表示されるようになりました。この新しい情報はウィンドウの中央にあるメニューから利用できます :



このメニューには以下の新しい項目があります :

- **物理メモリ** : 4D Server が使用する、マシンにインストールされた RAM メモリの量を表します。以前のバージョンの 4D Server でこのオプションは " メモリ " エリアにありました。
- **仮想メモリ** : 4D Server アプリケーションが使用する仮想メモリの量をグラフエリアに表示します。このメモリはアプリケーションの要求に応じてシステムが割り当てます。エリアの右下に現在使用されているメモリ量が表示されます。左上には利用可能な仮想メモリの最大値が表示されます。最大値はアプリケーションの一般メモリ設定に基づき、動的に計算されます。
- **キャッシュ** : 4D Server アプリケーションが使用するキャッシュメモリの量をグラフエリアに表示します。エリアの右下に現在使用されているメモリ量が表示されます。左上にはデータベース設定で設定されたキャッシュメモリの総量が表示されます。
このオプションが選択されている場合、グラフエリアのスクロールは遅くなります。これはキャッシュの効果的な解析を実行するために、長めの調査間隔が必要だからです。

" 技術的な " ユーザを非表示

以前のバージョンの 4D Server では、サーバー上で 1 つでもストアドプロシージャが起動され、クライアントが接続されると、ユーザが自動でユーザリストに追加されました。1 つでも Web プロセスが起動されたときにも同じメカニズムが実装されていました。技術的な理由で作成されるこのタイプのユーザは、もちろんアプリケーションが使用するライセンスにはカウントされません。とはいえ、この種のユーザが表示されることが混乱のもととなることがありました。

このような混乱を避けるために、4D Server v12 の管理ウィンドウには上記のような技術的なユーザが表示されないようになりました。この変更は 4D Server の動作に影響を与えません。

Mac OS でサービスの非サポート

技術的な理由から、4D v12 より 4D Server データベースをサービスとして登録する機能は、Mac OS でサポートされなくなりました。対応するメニューコマンドは Mac OS の 4D Server インタフェースで利用できなくなります。

Windows においては、このメカニズムに変更ありません。

A PHP モジュール

この appendix では、4D v12 の PHP モジュール実装について説明します。以下の項目について説明します :

- 4D の PHP インタプリタでデフォルトで提供される標準 PHP モジュールのリスト
- 4D が保持しない標準 PHP モジュールのリスト
- 追加モジュールのインストール

デフォルトで提供されるモジュール

以下の表は 4D v12 でデフォルトで提供される PHP モジュールの詳細です。

汎用モジュール

名称	Web サイト	説明
BCMath	http://php.net/bc	文字列として表された任意の大きさおよび 精度の数をサポートするバイナリ計算機
	例 : // ($myValue*$myValue*$myValue) の結果を返す **PHP Execute("";"bcpow";$result;String($myValue);"3";)**	
カレンダー	http://php.net/calendar	異なったカレンダーフォーマット間の変換を 簡単に行う関数の集まり。ユリウス積算日を標準とする。
	例 : **PHP Execute**("";"cal_days_in_month";Number of days in Febrary;1;2;Year())	
Ctype	http://php.net/ctype	現在のロケールに基づき文字または文字列がある文字クラスに含まれるかどうかを調べる関数。
	例 : // 提供された文字列のすべての文字が句読点かチェックする **PHP Execute**("";"ctype_punct";isPunct;$myString)	

名称	Web サイト	説明
日付・時刻	http://php.net/datetime	PHP スクリプトを実行するサーバから日付と時間を取得する。
	例 : // ポルトガルのリスボンにおける日の出時刻を計算する。緯度 : 38.4 North、経度 : 9 West、天頂 ~= 90、時差 : +1 GMT **PHP Execute**("";"eval";$resultType;"SUNFUNCS_RET_STRING") **PHP Execute**("";"date_sunrise";SunriseTime;$timestamp;$resultType;38.4l;-9; 90; 1)	
DOM (Document Object Model)	http://php.net/dom	PHP5 の DOM API による XML ドキュメントの処理。
Exif	http://php.net/exif	画像のメタデータを扱う。
ファイル情報 (*)	http://php.net/fileinfo	ファイルの content-type とエンコーディングを推測する。
Filter	http://php.net/filter	安全でないソース、例えばユーザ入力などによるデータの検証や除去を行う。
	例 : **PHP Execute**("";"filter_id";$filterId;"FILTER_VALIDATE_EMAIL") **ON ERR CALL**("notValidEmail") validEmail:=$emailToValidate $ok:=**PHP Execute**("";"filter_var";$test;$filterId;$emailToValidate;$resultType) **ON ERR CALL**("")	
FTP (File Transfer Protocol)	http://php.net/ftp	FTP サーバへの詳細なアクセスを提供。
Hash	http://php.net/hash	メッセージダイジェストエンジン。さまざまなハッシュアルゴリズムを使用して、任意の長さのメッセージに対する直接的あるいは段階的な処理を可能とする。
	例 : **PHP Execute**("";"hash";$md5Result;"md5";$myString)	
GD (Graphics Draw) ライブラリ	http://php.net/gd	画像処理。
Iconv	http://php.net/iconv	様々な文字セット間でのファイル変換。
JSON (JavaScript Object Notation)	http://php.net/json	JSON データ交換形式の実装

名称	Web サイト	説明
LDAP	http://php.net/ldap	LDAP は、" ディレクトリサーバ " にアクセスするために使用されるプロトコル。ディレクトリとは、ツリー構造に情報を保持している特殊なデータベース。
LibXML	http://php.net/libxml	XML 関数や定数のライブラリ
マルチバイト文字列	http://php.net/mbstring	複数バイト文字エンコーディングの処理や文字列の変換に使用できる、文字処理のための一連の関数。
OpenSSL	http://php.net/openssl	OpenSSL の関数を使用して署名の生成、そして、データのシール (暗号化) およびオープン (復号化) を行う。
PCRE (Perl Compatible Regular Expressions)	http://php.net/pcre	Perl 5 と同じシンタックスおよび語義を使用する一連の正規表現関数。
	例 : // この例は文字列から不要なスペースを取り除きます。 $myString:="foo o bar" **PHP Execute**("";"preg_replace";$myString;"/\\s\\s+/";" ";$myString) // $myString からスペースの繰り返しが削除され "foo o bar" になる。	
PDO (PHP Data Objects)	http://php.net/pdo	データベースアクセスのインタフェース。データベース毎の PDO ドライバが必要。
PDO_SQLITE	http://php.net/pdo_sqlite	SQLite 3 に PHP からのアクセスを可能にする、PHP Data Objects (PDO) インタフェースを実装したドライバ。
リフレクション	http://php.net/reflection	完全なリフレクション API で、クラス、インターフェイス、関数、メソッド、そしてエクステンションのリバースエンジニアリングを可能にする。
Phar (PHP Archive)	http://php.net/phar	PHP アプリケーション全体をひとつの "phar" (PHP Archive) ファイルにまとめてしまい、配布やインストールを容易にする。

名称	Web サイト	説明
Session	http://php.net/session	PHP セッションのサポート
	例 : セッションは Web アプリケーションにおいて、それぞれのリクエスト間でコンテキストを保持するために使用されます。4D で PHP Execute を実行すると、PHP スクリプトはセッションを開始し、コンテキストとして保持すべき情報を $_SESSION 配列に格納することが可能となります。 PHP スクリプトがセッションを使用する場合、PHP GET FULL RESPONSE コマンドを使用して PHP から返されるセッション ID を取得し、PHP Execute を呼び出す都度、事前に SET ENVIRONMENT VARIABLE コマンドを使用してセッション ID を指定しなければなりません。 // "PHP Execute with context" メソッド **If** (<>PHP_Session # "") **SET ENVIRONMENT VARIABLE** ("HTTP_COOKIE";<>PHP_Session) **End if** **If** (**PHP Execute**($1)) **PHP GET FULL RESPONSE** ($0;$errorInfos;$errorValues;$headerFields; $headerValues) $idx:=**Find in array**($headerFields; "Set-Cookie") **If** ($idx>0) <>PHP_Session:=$headerValues{$idx} **End if** **End if**	
SimpleXML	http://php.net/simpleXML	とても簡単かつ容易に使用できるツールで、XML をプロパティや配列反復子で処理可能なオブジェクトに変換するために使用します。
ソケット	http://php.net/sockets	BSD ソケットに基づくソケット通信機能の低レベルインタフェースを実装し、クライアントとしてだけでなく、ソケットサーバとして動作させることが可能となります。
SPL (Standard PHP Library)	http://php.net/spl	標準的な問題を解決するためのインターフェイスやクラスを集めたもの。
SQLite	http://php.net/sqlite	SQLite データベースエンジン用の拡張。このエンジンは埋め込み可能。
SQLite3	http://php.net/sqlite3	SQLite version 3 データベースをサポート。

名称	Web サイト	説明
Tokenizer	http://php.net/tokenizer	字句解析レベルの言語処理を行うことなく、PHP ソースを解析 / 修正するツールを作成することを可能にする関数。
XML (eXtensible Markup Language)	http://php.net/xml	XML ドキュメントの解析。
XMLreader	http://php.net/xmlreader	プル型の XML パーサー。
XMLwriter	http://php.net/xmlwriter	XML 形式のストリームやファイルを生成。
Zlib	http://php.net/zlib	gzip (.gz) 圧縮ファイルの読み込みと書き出し。
	例 : **GET HTTP HEADER**($names;$values) $pos:=**Find in array**($names;"Accept-Encoding") **If**($pos>0) **Case of** :(**Position**($values{$pos};"gzip")>0) **SET HTTP HEADER**("Content-Encoding: gzip") **PHP Execute**("";"gzencode";$html;$html) :(**Position**($values{$pos};"deflate")>0) **SET HTTP HEADER**("Content-Encoding: deflate") **PHP Execute**("";"gzdeflate";$html;$html) **End case** **End if** **SEND HTML TEXT**($html)	
Zip	http://php.net/zip	ZIP 圧縮アーカイブやその中のファイルの読み込みと書き出し。

(*) 現バージョンの 4D v12 では、Windows 上でこれらを利用できません。

Windows でのみ利用可能なモジュール

構造的な理由から、以下の PHP モジュールは Windows 上でのみ利用可能です。

名称	Web サイト	説明
COM & .NET	http://php.net/com	COM (Component Object Model) は Windows プラットフォーム上でアプリケーションやコンポーネントが通信を行うおもな方法です。 さらに、4D は COM レイヤからの .Net アセンブリのインスタンス化や作成をサポートします。
ODBC (Open DataBase Connectivity)	http://php.net/odbc	標準の ODBC サポートに加え、PHP の統合 ODBC 関数により、独自の API 実装に ODBC API の構文を利用したいくつかのデータベースへのアクセスが提供されます。
WDDX (Web Distributed Data eXchange)	http://php.net/wddx	プラットフォームにかかわらず、Web を利用した Web アプリケーション間のデータ交換を容易にします。

無効にされたモジュール

以下の PHP モジュールは 4D v12 に実装されていません。一番右の列にその理由が記載されています :

名称	Web サイト	理由 - 代わりのソリューション
Mimetype	http://php.net/mime-magic	廃止 (廃止予定) - Fileinfo を使用。
POSIX (Portable Operating System Interface)	http://php.net/posix	廃止 (廃止予定)。
Regular Expression (POSIX Extended)	http://php.net/regex	廃止 (廃止予定) - PCRE を使用。
Crack	http://php.net/crack	**ライセンスの制限**。
ffmpeg	http://ffmpeg-php.sourceforge.net/	ライセンスの制限 - LAUNCH EXTERNAL PROCESS で、コマンドラインで ffmpeg を使用
Image Magick	http://php.net/manual/book.imagick.php	ライセンスの制限 - GD 2 を使用。
IMAP (Internet Message Access Protocol)	http://php.net/imap	ライセンスの制限 - 4D Internet Commands に統合されたコマンドを使用。
PDF (Portable Document Format)	http://php.net/pdf	ライセンスの制限 - Haru PDF を利用。

名称	Web サイト	理由 - 代わりのソリューション
Mysqlnd (MySQL Native Driver)	http://dev.mysql.com/downloads/connector/php-mysqlnd/	4D 環境に適切でない

追加モジュールのインストール

PHP インタプリタに追加のモジュールをインストールすることが可能です。これによりデフォルトで提供されていない機能を使用できます。

以下のタイプの拡張をインストールできます :

- PECL (PHP Extension Community Library) 拡張
- PEAR (PHP Extension and Application Repository) フレームワークの拡張
- Zend フレームワークの拡張
- Symphony フレームワークの拡張
- JELIX フレームワークの拡張
- eZ コンポーネント

各タイプのインストール情報は以下の通りです。

Note: 4D v12 で提供される PHP バージョンは以下の特徴を持ちます :
- バージョン 5.3.2
- Windows と Mac OS で 32-bit コンパイル
- Windows と Mac OS で " スレッドセーフ " コンパイル

PECL 拡張

Web サイト : http://pecl.php.net

▶ PECL 拡張を追加するには :

1 目的の PECL 拡張をダウンロードしてビルドする。
または
PHP 5.3 VC9 Non Thread Safe Windows バイナリパッケージからビルド済みの拡張を取り出す。(http://windows.php.net/download/#php-5.3-nts-VC9-x86)

2 extension フォルダに拡張を追加する。

3 php.ini で拡張を有効にする。

警告 : PECL Web サイトで利用可能な拡張が制限の緩い PHP ライセンスであったとしても、そのうちのいくつかはよりライセンスに制限のあるライブラリを必要とする場合があります。

他の Web サイトで PHP 拡張を入手できる場合もありますが、この場合 PHP グループの検証によりに提供されるセキュリティ保証はありません。

PEAR 拡張

Web Site: http://pear.php.net

PEAR は完全なオブジェクト指向のフレームワークです。

▶ PEAR 拡張を追加するには :

1 PEAR パッケージをダウンロード (http://pear.php.net/package/PEAR/download) して、"pear" フォルダに解凍する。

2 この "pear" フォルダを "php.ini" ファイルで指定した "include_path" に追加する。

Zend 拡張

1 Zend フレームワークをダウンロード (http://framework.zend.com/download/latest) して、"zend" フォルダに解凍する。

2 この "zend" フォルダを "php.ini" ファイルで指定した "include_path" に追加する。

3 Zend フレームワークコンポーネントのドキュメントを読む : http://framework.zend.com/manual/en

Symphony 拡張

Web サイト : http://www.symfony-project.org

Symphony は Web コントローラを含む Web アプリケーションのフレームワークです。

1 以下のアドレスからフレームワークとサンドボックスをダウンロードし、解凍する : http://www.symfony-project.org/installation/1_2.

JELIX 拡張

1. JELIX フレームワークをダウンロード (http://jelix.org/articles/en/download/stable) し、解凍する。
2. 出来上がった "jelix" フォルダを "php.ini" ファイルで指定した "include_path" に追加する。
3. JELIX フレームワークコンポーネントのドキュメントを読む : http://jelix.org/articles/en/manual-1.1/components

eZ Components

1. eZ Components をダウンロード (http://www.ezcomponents.org/download) し、"ez" フォルダに解凍する。
2. 出来上がった "ez" フォルダを "php.ini" ファイルで指定した "include_path" に追加する。
3. eZ Components のドキュメントを読む : http://www.ezcomponents.org/docs/api/latest

B スタイルタグ

この appendix では、4D v12 でサポートされる、" マルチスタイル " 属性が設定されたテキストエリアのスタイルタグをリストします。個の属性値についての説明は 94 ページの " リッチテキスト " を参照してください。

これらのタグを使用してカスタマイズされたスタイル管理を行うことができます。ここに示されたタグのみがスタイルバリエーションとして 4D で解釈されます。

フォント名

<SPAN STYLE="font-family: DESDEMONA"> ... </SPAN>

フォントサイズ

<SPAN STYLE="font-size: 20pt"> ... </SPAN>

フォントスタイル

- **太字**

  <SPAN STYLE="font-weight: bold"> ... </SPAN>

- **イタリックまたは通常**

  <SPAN STYLE="font-style: italic"> ... </SPAN>

  ***<SPAN STYLE="font-style: normal"> ... </SPAN>

- **下線**

  <SPAN STYLE="text-decoration: underline"> ... </SPAN>

- **取り消し線**

  <SPAN STYLE="text-decoration:line-through">...</SPAN>

Note: "strikethrough" は Mac OS でサポートされていません。しかし対応するタグをプログラミングで使用することはできます。

フォントカラー

<SPAN STYLE="color:green"> ... </SPAN>

または

<SPAN STYLE="color:#006CCC">...</SPAN>

背景色 (Windows のみ)

<SPAN STYLE="background-color:green"> ... </SPAN>

または

<SPAN STYLE="background-color:#006CCC">...</SPAN>

Note: Mac OS では、この設定は無視されます。オブジェクトが更新されると、この設定は取り除かれます。

カラー値

フォントカラーと背景色属性には、RGB の 16 進表現、または W3C の CSS 標準で定義された 16 の HTML カラーの名前をカラー値として指定できます :

Color								
Name	Aqua	Black	Blue	Fuchsia	Gray	Green	Lime	Maroon
RGB	#00FFFF	#000000	#0000FF	#FF00FF	#808080	#008000	#00FF00	#800000

Color								
Name	Navy	Olive	Purple	Red	Silver	Teal	White	Yellow
RGB	#000080	#808000	#800080	#FF0000	#C0C0C0	#008080	#FFFFFF	#FFFF00

C メタデータ定数

ここではピクチャやそれらに含まれるデータタイプ中で、4D v12 によりサポートされるメタデータタグのリストを記載します。これらのタグに関する詳細は SET PICTURE　METADATA や GET PICTURE METADATA コマンドの説明を参照してください。

ピクチャメタデータ名

EXIF

"Picture metadata names" テーマの定数 (metaName 引数)	値	利用可能な値の型または割り当てられた "Picture metadata values" テーマの定数 (contents 引数)
EXIF Aperture Value	EXIF/ApertureValue	実数 (APEX value)
EXIF Brightness Value	EXIF/BrightnessValue	実数 (APEX value)
EXIF Color Space	EXIF/ColorSpace	EXIF Adobe RGB
		EXIF s RGB
		EXIF Uncalibrated
EXIF Components Configuration	EXIF/ComponentsConfiguration	EXIF B
		EXIF Cb
		EXIF Cr
		EXIF G
		EXIF R
		EXIF Unused
		EXIF Y
EXIF Compressed Bits Per Pixel	EXIF/CompressedBitsPerPixel	実数
EXIF Contrast	EXIF/Contrast	EXIF High
		EXIF Low
		EXIF Normal
EXIF Custom Rendered	EXIF/CustomRendered	EXIF Normal
		EXIF Custom

EXIF Date Time Digitized	EXIF/DateTimeDigitized	日付またはテキスト (XML Datetime)
EXIF Date Time Original	EXIF/DateTimeOriginal	日付またはテキスト (XML Datetime)
EXIF Digital Zoom Ratio	EXIF/DigitalZoomRatio	実数
EXIF Exif Version	EXIF/ExifVersion	整数配列 (4 つの値)
EXIF Exposure Bias Value	EXIF/ExposureBiasValue	実数
EXIF Exposure Index	EXIF/ExposureIndex	実数
EXIF Exposure Mode	EXIF/ExposureMode	EXIF Auto
		EXIF Auto Bracket
		EXIF Manual
EXIF Exposure Program	EXIF/ExposureProgram	EXIF Manual
		EXIF Action
		EXIF Aperture Priority AE
		EXIF Creative
		EXIF Landscape
		EXIF Exposure Portrait
		EXIF Program AE
		EXIF Shutter Speed Priority AE
EXIF Exposure Time	EXIF/ExposureTime	実数
EXIF File Source	EXIF/FileSource	EXIF Digital Camera
		EXIF Film Scanner
		EXIF Reflection Print Scanner
EXIF Flash	EXIF/Flash	EXIF Auto Mode
		EXIF Compulsory Flash Firing
		EXIF Compulsory Flash Suppression
		EXIF Unknown
		EXIF Detected
		EXIF No Detection Function
		EXIF Not Detected
		EXIF Reserved
EXIF Flash Energy	EXIF/FlashEnergy	実数
EXIF Flash Pix Version	EXIF/FlashPixVersion	整数配列 (4 つの値)
EXIF Flash Fired	EXIF/Flash/Fired	ブール
EXIF Flash Function Present	EXIF/Flash/FunctionPresent	ブール
EXIF Flash Mode	EXIF/Flash/Mode	EXIF Auto Mode
		EXIF Compulsory Flash Firing
		EXIF Compulsory Flash Suppression
		EXIF Unknown
EXIF Flash Red Eye Reduction	EXIF/Flash/RedEyeReduction	ブール
EXIF Flash Return Light	EXIF/Flash/ReturnLight	EXIF Detected
		EXIF No Detection Function
		EXIF Not Detected

		EXIF Reserved
EXIF F Number	EXIF/FNumber	実数
EXIF Focal Len In 35 mm Film	EXIF/FocalLenIn35mmFilm	倍長整数
EXIF Focal Length	EXIF/FocalLength	実数
EXIF Focal Plane Resolution Unit	EXIF/FocalPlaneResolutionUnit	倍長整数
EXIF Focal Plane X Resolution	EXIF/FocalPlaneXResolution	実数
EXIF Focal Plane Y Resolution	EXIF/FocalPlaneYResolution	実数
EXIF Gain Control	EXIF/GainControl	EXIF High Gain Down
		EXIF High Gain Up
		EXIF Low Gain Down
		EXIF Low Gain Up
		EXIF None
EXIF Gamma	EXIF/Gamma	実数
EXIF Image Unique ID	EXIF/ImageUniqueID	テキスト
EXIF ISO Speed Ratings	EXIF/ISOSpeedRatings	倍長整数または倍長整数配列
EXIF Light Source	EXIF/LightSource	EXIF Unknown
		EXIF Cloudy
		EXIF Cool White Fluorescent
		EXIF D50
		EXIF D55
		EXIF D65
		EXIF D75
		EXIF Daylight
		EXIF Daylight Fluorescent
		EXIF Day White Fluorescent
		EXIF Fine Weather
		EXIF Flash
		EXIF Light Fluorescent
		EXIF ISOStudio Tungsten
		EXIF Other
		EXIF Shade
		EXIF Standard Light A
		EXIF Standard Light B
		EXIF Standard Light C
		EXIF Tungsten
		EXIF White Fluorescent
EXIF Maker Note	EXIF/MakerNote	テキスト
EXIF Max Aperture Value	EXIF/MaxApertureValue	実数
EXIF Metering Mode	EXIF/MeteringMode	EXIF Other
		EXIF Average
		EXIF Center Weighted Average
		EXIF Multi Segment
		EXIF Multi Spot
		EXIF Partial
		EXIF Spot

EXIF Pixel X Dimension	EXIF/PixelXDimension	倍長整数
EXIF Pixel Y Dimension	EXIF/PixelYDimension	倍長整数
EXIF Related Sound File	EXIF/RelatedSoundFile	テキスト
EXIF Saturation	EXIF/Saturation	EXIF High
		EXIF Low
		EXIF Normal
EXIF Scene Capture Type	EXIF/SceneCaptureType	EXIF Scene Landscape
		EXIF Night
		EXIF Scene Portrait
		EXIF Standard
EXIF Scene Type	EXIF/SceneType	倍長整数
EXIF Sensing Method	EXIF/SensingMethod	EXIF Color Sequential Area
		EXIF Color Sequential Linear
		EXIF Not Defined
		EXIF One Chip Color Area
		EXIF Three Chip Color Area
		EXIF Trilinear
		EXIF Two Chip Color Area
EXIF Sharpness	EXIF/Sharpness	EXIF High
		EXIF Low
		EXIF Normal
EXIF Shutter Speed Value	EXIF/ShutterSpeedValue	実数
EXIF Spectral Sensitivity	EXIF/SpectralSensitivity	テキスト
EXIF Subject Area	EXIF/SubjectArea	倍長整数配列 (2, 3 または 4 つの値)
EXIF Subject Dist Range	EXIF/SubjectDistRange	EXIF Unknown
		EXIF Close
		EXIF Distant
		EXIF Macro
EXIF Subject Distance	EXIF/SubjectDistance	実数
EXIF Subject Location	EXIF/SubjectLocation	倍長整数配列 (2 つの値)
EXIF User Comment	EXIF/UserComment	テキスト
EXIF White Balance	EXIF/WhiteBalance	EXIF Auto
		EXIF Manual

GPS

"Picture metadata names" テーマの定数 (metaName 引数)	値	利用可能な値の型または割り当てられた "Picture metadata values" テーマの定数 (contents 引数)
GPS Altitude	GPS/Altitude	GPS Above Sea Level
		GPS Below Sea Level
GPS Altitude Ref	GPS/AltitudeRef	GPS Above Sea Level
		GPS Below Sea Level

GPS Area Information	GPS/AreaInformation	テキスト
GPS Date Time	GPS/DateTime	日付またはテキスト (XML Datetime)
GPS Dest Bearing	GPS/DestBearing	テキスト (1 文字)
GPS Dest Bearing Ref	GPS/DestBearingRef	テキスト (1 文字)
GPS Dest Distance	GPS/DestDistance	テキスト (1 文字)
GPS Dest Distance Ref	GPS/DestDistanceRef	テキスト (1 文字)
GPS Dest Latitude	GPS/DestLatitude	テキスト
GPS Dest Latitude Deg	GPS/DestLatitude/Deg	実数
GPS Dest Latitude Dir	GPS/DestLatitude/Dir	テキスト (1 文字)
GPS Dest Latitude Min	GPS/DestLatitude/Min	実数
GPS Dest Latitude Sec	GPS/DestLatitude/Sec	実数
GPS Dest Longitude	GPS/DestLongitude	テキスト
GPS Dest Longitude Deg	GPS/DestLongitude/Deg	実数
GPS Dest Longitude Dir	GPS/DestLongitude/Dir	テキスト (1 文字)
GPS Dest Longitude Min	GPS/DestLongitude/Min	実数
GPS Dest Longitude Sec	GPS/DestLongitude/Sec	実数
GPS Differential	GPS/Differential	GPS Correction Applied
		GPS Correction Not Applied
GPS DOP	GPS/DOP	実数
GPS Img Direction	GPS/ImgDirection	GPS Magnetic north
		GPS True north
GPS Img Direction Ref	GPS/ImgDirectionRef	GPS Magnetic north
		GPS True north
GPS Latitude	GPS/Latitude	GPS North
		GPS South
GPS Latitude Deg	GPS/Latitude/Deg	実数
GPS Latitude Dir	GPS/Latitude/Dir	GPS North
		GPS South
GPS Latitude Min	GPS/Latitude/Min	実数
GPS Latitude Sec	GPS/Latitude/Sec	実数
GPS Longitude	GPS/Longitude	GPS West
		GPS East
GPS Longitude Deg	GPS/Longitude/Deg	実数
GPS Longitude Dir	GPS/Longitude/Dir	GPS West
		GPS East
GPS Longitude Min	GPS/Longitude/Min	実数
GPS Longitude Sec	GPS/Longitude/Sec	実数
GPS Map Datum	GPS/MapDatum	テキスト
GPS Measure Mode	GPS/MeasureMode	GPS 2D
		GPS 3D
GPS Processing Method	GPS/ProcessingMethod	テキスト
GPS Satellites	GPS/Satellites	テキスト
GPS Speed	GPS/Speed	GPS km h
		GPS miles h

		GPS knots h
GPS Speed Ref	GPS/SpeedRef	GPS km h
		GPS miles h
		GPS knots h
GPS Status	GPS/Status	GPS Measurement in progress
		GPS Measurement Interoperability
GPS Track	GPS/Track	実数 (0.00..359.99)
GPS Track Ref	GPS/TrackRef	テキスト (1 文字)
GPS Version ID	GPS/VersionID	倍長整数配列 (4 文字)

IPTC

"Picture metadata names" テーマの定数 (metaName 引数)	値	利用可能な値の型または割り当てられた "Picture metadata values" テーマの定数 (contents 引数)
IPTC Byline	IPTC/Byline	テキストまたはテキスト配列
IPTC Byline Title	IPTC/BylineTitle	テキストまたはテキスト配列
IPTC Caption Abstract	IPTC/CaptionAbstract	テキスト
IPTC Category	IPTC/Category	テキスト
IPTC City	IPTC/City	テキスト
IPTC Contact	IPTC/Contact	テキストまたはテキスト配列
IPTC Content Location Code	IPTC/ContentLocationCode	テキストまたはテキスト配列
IPTC Content Location Name	IPTC/ContentLocationName	テキストまたはテキスト配列
IPTC Copyright Notice	IPTC/CopyrightNotice	テキスト
IPTC Country Primary Location Code	IPTC/CountryPrimaryLocationCode	テキスト
IPTC Country Primary Location Name	IPTC/CountryPrimaryLocationName	テキスト
IPTC Credit	IPTC/Credit	テキスト
IPTC Date Time Created	IPTC/DateTimeCreated	日付またはテキスト (XML Datetime)
IPTC Digital Creation Date Time	IPTC/DigitalCreationDateTime	日付またはテキスト (XML Datetime)
IPTC Edit Status	IPTC/EditStatus	テキスト
IPTC Expiration Date Time	IPTC/ExpirationDateTime	日付またはテキスト (XML Datetime)
IPTC Fixture Identifier	IPTC/FixtureIdentifier	テキスト
IPTC Headline	IPTC/Headline	テキスト
IPTC Image Orientation	IPTC/ImageOrientation	テキスト
IPTC Image Type	IPTC/ImageType	テキスト
IPTC Keywords	IPTC/Keywords	テキストまたはテキスト配列
IPTC Language Identifier	IPTC/LanguageIdentifier	テキスト

IPTC Object Attribute Reference	IPTC/ObjectAttributeReference	テキスト
IPTC Object Cycle	IPTC/ObjectCycle	テキスト
IPTC Object Name	IPTC/ObjectName	テキスト
IPTC Original Transmission Reference	IPTC/OriginalTransmissionReference	テキスト
IPTC Originating Program	IPTC/OriginatingProgram	テキスト
IPTC Program Version	IPTC/ProgramVersion	テキスト
IPTC Province State	IPTC/ProvinceState	テキスト
IPTC Release Date Time	IPTC/ReleaseDateTime	日付またはテキスト (XML Datetime)
IPTC Scene	IPTC/Scene	IPTC Action
		IPTC Aerial View
		IPTC Close Up
		IPTC Couple
		IPTC Exterior View
		IPTC Full Length
		IPTC General View
		IPTC Group
		IPTC Half Length
		IPTC Headshot
		IPTC Interior View
		IPTC Movie Scene
		IPTC Night Scene
		IPTC Off Beat
		IPTC Panoramic View
		IPTC Performing
		IPTC Posing
		IPTC Profile
		IPTC Rear View
		IPTC Satellite
		IPTC Single
		IPTC Symbolic
		IPTC Two
		IPTC Under Water
IPTC Source	IPTC/Source	テキスト
IPTC Special Instructions	IPTC/SpecialInstructions	テキスト
IPTC Star Rating	IPTC/StarRating	倍長整数
IPTC Sub Location	IPTC/SubLocation	テキスト
IPTC Subject Reference	IPTC/SubjectReference	倍長整数または倍長整数配列
IPTC Supplemental Category	IPTC/SupplementalCategory	テキストまたはテキスト配列
IPTC Urgency	IPTC/Urgency	倍長整数
IPTC Writer Editor	IPTC/WriterEditor	テキストまたはテキスト配列

TIFF

"Picture metadata names" テーマの定数 (metaName 引数)	値	利用可能な値の型または割り当てられた "Picture metadata values" テーマの定数 (contents 引数)
TIFF Artist	TIFF/Artist	テキスト
TIFF Compression	TIFF/Compression	TIFF Adobe Deflate
		TIFF CCIRLEW
		TIFF CCITT1D
		TIFF DCS
		TIFF Deflate
		TIFF Epson ERF
		TIFF IT8BL
		TIFF IT8CTPAD
		TIFF IT8LW
		TIFF IT8MP
		TIFF JBIG
		TIFF JBIGB&W
		TIFF JBIGColor
		TIFF JPEG
		TIFF JPEG2000
		TIFF JPEGThumbs Only
		TIFF Kodak262
		TIFF Kodak DCR
		TIFF Kodak KDC
		TIFF LZW
		TIFF MDIBinary Level Codec
		TIFF MDIProgressive Transform Codec
		TIFF MDIVector
		TIFF Next
		TIFF Nikon NEF
		TIFF Pack Bits
		TIFF Pentax PEF
		TIFF Pixar Film
		TIFF Pixar Log
		TIFF SGILog
		TIFF SGILog24
		TIFF Sony ARW
		TIFF T4Group3Fax
		TIFF T6Group4Fax
		TIFF Thunderscan
		TIFF Uncompressed
TIFF Copyright	TIFF/Copyright	テキスト

TIFF Date Time	TIFF/DateTime	日付またはテキスト (XML Datetime)
TIFF Document Name	TIFF/DocumentName	テキスト
TIFF Host Computer	TIFF/HostComputer	テキスト
TIFF Image Description	TIFF/ImageDescription	テキスト
TIFF Make	TIFF/Make	テキスト
TIFF Model	TIFF/Model	テキスト
TIFF Orientation	TIFF/Orientation	TIFF Horizontal
		TIFF Mirror Horizontal
		TIFF Mirror Horizontal And Rotate270CW
		TIFF Mirror Horizontal And Rotate90CW
		TIFF Mirror Vertical
		TIFF Rotate180
		TIFF Rotate270CW
		TIFF Rotate90CW
TIFF Photometric Interpretation	TIFF/PhotometricInterpretation	TIFF Black Is Zero
		TIFF CIELab
		TIFF CMYK
		TIFF Color Filter Array
		TIFF ICCLab
		TIFF ITULab
		TIFF Linear Raw
		TIFF Pixar Log L
		TIFF Pixar Log Luv
		TIFF RGB
		TIFF RGBPalette
		TIFF Transparency Mask
		TIFF White Is Zero
		TIFF YCb Cr
TIFF Resolution Unit	TIFF/ResolutionUnit	TIFF CM
		TIFF Inches
		TIFF MM
		TIFF None
		TIFF UM
TIFF Software	TIFF/Software	テキスト
TIFF XResolution	TIFF/XResolution	実数
TIFF YResolution	TIFF/YResolution	実数

ピクチャメタデータの値

以下のリストは "Picture metadata values" テーマの定数とその値です。これらの定数は、metaName 引数に応じて、SET PICTURE METADATA および GET PICTURE METADATA コマンドの contents 引数で利用できます :

定数	型	値
EXIF Adobe RGB	倍長整数	2
EXIF s RGB	倍長整数	1
EXIF Uncalibrated	倍長整数	-1
EXIF B	倍長整数	6
EXIF Cb	倍長整数	2
EXIF Cr	倍長整数	3
EXIF G	倍長整数	5
EXIF R	倍長整数	4
EXIF Unused	倍長整数	0
EXIF Y	倍長整数	1
EXIF High	倍長整数	2
EXIF Low	倍長整数	1
EXIF Normal	倍長整数	0
EXIF Normal	倍長整数	0
EXIF Custom	倍長整数	1
EXIF Auto	倍長整数	0
EXIF Auto Bracket	倍長整数	2
EXIF Manual	倍長整数	1
EXIF Manual	倍長整数	1
EXIF Action	倍長整数	6
EXIF Aperture Priority AE	倍長整数	3
EXIF Creative	倍長整数	5
EXIF Landscape	倍長整数	8
EXIF Exposure Portrait	倍長整数	7
EXIF Program AE	倍長整数	2
EXIF Shutter Speed Priority AE	倍長整数	4
EXIF Digital Camera	倍長整数	3
EXIF Film Scanner	倍長整数	1
EXIF Reflection Print Scanner	倍長整数	2
EXIF Auto Mode	倍長整数	3
EXIF Compulsory Flash Firing	倍長整数	1
EXIF Compulsory Flash Suppression	倍長整数	2
EXIF Unknown	倍長整数	0
EXIF Detected	倍長整数	3
EXIF No Detection Function	倍長整数	0
EXIF Not Detected	倍長整数	2
EXIF Reserved	倍長整数	1
EXIF Auto Mode	倍長整数	3

EXIF Compulsory Flash Firing	倍長整数	1
EXIF Compulsory Flash Suppression	倍長整数	2
EXIF Unknown	倍長整数	0
EXIF Detected	倍長整数	3
EXIF No Detection Function	倍長整数	0
EXIF Not Detected	倍長整数	2
EXIF Reserved	倍長整数	1
EXIF High Gain Down	倍長整数	4
EXIF High Gain Up	倍長整数	2
EXIF Low Gain Down	倍長整数	3
EXIF Low Gain Up	倍長整数	1
EXIF None	倍長整数	0
EXIF Unknown	倍長整数	0
EXIF Cloudy	倍長整数	10
EXIF Cool White Fluorescent	倍長整数	14
EXIF D50	倍長整数	23
EXIF D55	倍長整数	20
EXIF D65	倍長整数	21
EXIF D75	倍長整数	22
EXIF Daylight	倍長整数	1
EXIF Daylight Fluorescent	倍長整数	12
EXIF Day White Fluorescent	倍長整数	13
EXIF Fine Weather	倍長整数	9
EXIF Flash	倍長整数	4
EXIF Light Fluorescent	倍長整数	2
EXIF ISOStudio Tungsten	倍長整数	24
EXIF Other	倍長整数	255
EXIF Shade	倍長整数	11
EXIF Standard Light A	倍長整数	17
EXIF Standard Light B	倍長整数	18
EXIF Standard Light C	倍長整数	19
EXIF Tungsten	倍長整数	3
EXIF White Fluorescent	倍長整数	15
EXIF Other	倍長整数	255
EXIF Average	倍長整数	1
EXIF Center Weighted Average	倍長整数	2
EXIF Multi Segment	倍長整数	5
EXIF Multi Spot	倍長整数	4
EXIF Partial	倍長整数	6
EXIF Spot	倍長整数	3
EXIF High	倍長整数	2
EXIF Low	倍長整数	1
EXIF Normal	倍長整数	0
EXIF Scene Landscape	倍長整数	1
EXIF Night	倍長整数	3

EXIF Scene Portrait	倍長整数	2
EXIF Standard	倍長整数	0
EXIF Color Sequential Area	倍長整数	5
EXIF Color Sequential Linear	倍長整数	8
EXIF Not Defined	倍長整数	1
EXIF One Chip Color Area	倍長整数	2
EXIF Three Chip Color Area	倍長整数	4
EXIF Trilinear	倍長整数	7
EXIF Two Chip Color Area	倍長整数	3
EXIF High	倍長整数	2
EXIF Low	倍長整数	1
EXIF Normal	倍長整数	0
EXIF Unknown	倍長整数	0
EXIF Close	倍長整数	2
EXIF Distant	倍長整数	3
EXIF Macro	倍長整数	1
EXIF Auto	倍長整数	0
EXIF Manual	倍長整数	1

GPS Above Sea Level	倍長整数	0
GPS Below Sea Level	倍長整数	1
GPS Above Sea Level	倍長整数	0
GPS Below Sea Level	倍長整数	1
GPS Correction Applied	倍長整数	1
GPS Correction Not Applied	倍長整数	0
GPS Magnetic north	テキスト	M
GPS True north	テキスト	T
GPS Magnetic north	テキスト	M
GPS True north	テキスト	T
GPS North	テキスト	N
GPS South	テキスト	S
GPS North	テキスト	N
GPS South	テキスト	S
GPS West	テキスト	W
GPS East	テキスト	E
GPS West	テキスト	W
GPS East	テキスト	E
GPS 2D	倍長整数	2
GPS 3D	倍長整数	3
GPS km h	テキスト	K
GPS miles h	テキスト	M
GPS knots h	テキスト	K
GPS km h	テキスト	K
GPS miles h	テキスト	M
GPS knots h	テキスト	K

GPS Measurement in progress	テキスト	A
GPS Measurement Interoperability	テキスト	V

IPTC Action	倍長整数	11900
IPTC Aerial View	倍長整数	11200
IPTC Close Up	倍長整数	11800
IPTC Couple	倍長整数	10700
IPTC Exterior View	倍長整数	11600
IPTC Full Length	倍長整数	10300
IPTC General View	倍長整数	11000
IPTC Group	倍長整数	10900
IPTC Half Length	倍長整数	10200
IPTC Headshot	倍長整数	10100
IPTC Interior View	倍長整数	11700
IPTC Movie Scene	倍長整数	12400
IPTC Night Scene	倍長整数	11400
IPTC Off Beat	倍長整数	12300
IPTC Panoramic View	倍長整数	11100
IPTC Performing	倍長整数	12000
IPTC Posing	倍長整数	12100
IPTC Profile	倍長整数	10400
IPTC Rear View	倍長整数	10500
IPTC Satellite	倍長整数	11500
IPTC Single	倍長整数	10600
IPTC Symbolic	倍長整数	12200
IPTC Two	倍長整数	10800
IPTC Under Water	倍長整数	11300

TIFF Adobe Deflate	倍長整数	8
TIFF CCIRLEW	倍長整数	32771
TIFF CCITT1D	倍長整数	2
TIFF DCS	倍長整数	32947
TIFF Deflate	倍長整数	32946
TIFF Epson ERF	倍長整数	32769
TIFF IT8BL	倍長整数	32898
TIFF IT8CTPAD	倍長整数	32895
TIFF IT8LW	倍長整数	32896
TIFF IT8MP	倍長整数	32897
TIFF JBIG	倍長整数	34661
TIFF JBIGB&W	倍長整数	9
TIFF JBIGColor	倍長整数	10
TIFF JPEG	倍長整数	7
TIFF JPEG2000	倍長整数	34712
TIFF JPEGThumbs Only	倍長整数	6
TIFF Kodak262	倍長整数	262

TIFF Kodak DCR	倍長整数	65000
TIFF Kodak KDC	倍長整数	32867
TIFF LZW	倍長整数	5
TIFF MDIBinary Level Codec	倍長整数	34718
TIFF MDIProgressive Transform Codec	倍長整数	34719
TIFF MDIVector	倍長整数	34720
TIFF Next	倍長整数	32766
TIFF Nikon NEF	倍長整数	34713
TIFF Pack Bits	倍長整数	32773
TIFF Pentax PEF	倍長整数	65535
TIFF Pixar Film	倍長整数	32908
TIFF Pixar Log	倍長整数	32909
TIFF SGILog	倍長整数	34676
TIFF SGILog24	倍長整数	34677
TIFF Sony ARW	倍長整数	32767
TIFF T4Group3Fax	倍長整数	3
TIFF T6Group4Fax	倍長整数	4
TIFF Thunderscan	倍長整数	32809
TIFF Uncompressed	倍長整数	1
TIFF Horizontal	倍長整数	1
TIFF Mirror Horizontal	倍長整数	2
TIFF Mirror Horizontal And Rotate270CW	倍長整数	5
TIFF Mirror Horizontal And Rotate90CW	倍長整数	7
TIFF Mirror Vertical	倍長整数	4
TIFF Rotate180	倍長整数	3
TIFF Rotate270CW	倍長整数	8
TIFF Rotate90CW	倍長整数	6
TIFF Black Is Zero	倍長整数	1
TIFF CIELab	倍長整数	8
TIFF CMYK	倍長整数	5
TIFF Color Filter Array	倍長整数	32803
TIFF ICCLab	倍長整数	9
TIFF ITULab	倍長整数	10
TIFF Linear Raw	倍長整数	34892
TIFF Pixar Log L	倍長整数	32844
TIFF Pixar Log Luv	倍長整数	32845
TIFF RGB	倍長整数	2
TIFF RGBPalette	倍長整数	3
TIFF Transparency Mask	倍長整数	4
TIFF White Is Zero	倍長整数	0
TIFF YCb Cr	倍長整数	6
TIFF CM	倍長整数	3
TIFF Inches	倍長整数	2
TIFF MM	倍長整数	4
TIFF None	倍長整数	1

TIFF UM	倍長整数	5