

# 4D v15への変換

"4D v15 への変換"のマニュアルへようこそ。ここでは、4D v14のデータベースを4D v15用に変換する前、変換中、そして変換後にチェックすべきさまざまな点についての説明がなされています。

4D v14のデータベースの変換は4D v15ではスムーズに行われるはずですが、全てが上手く行くためのいくつかの推奨点が"**変換の原則**"の章にまとめられています。しかしながら、変換が終わった後にもいくつかチェックすべき事柄があります。それは"**新しい互換性オプション**"と"**振る舞いの変更**"であり、どちらもアプリケーションレベルと4Dコマンドレベルで関わるので、4D v15の新機能を完全に使いこなすためにはどちらも理解しておく必要があります。

最後に、このマニュアルでは**4D v15で廃止予定の機能**をまとめてあります。デベロッパの方は、新しい機能をセットアップするために、ここを見る事で廃止予定の機能を探す手間が省ける事でしょう。

**注:** このマニュアルに記載されている新機能・変更の一部は、4D v14の"R-リリース"プログラムで既に紹介・導入されているものです。

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

- **4D v14:** "[Conversion to 4D v14](#)" と "[Deprecated features 4D v14 and higher](#)".
- **4D v13:** "[Conversion to 4D v13](#)" と "[Deprecated features 4D v13 and higher](#)".
- **4D v12:** "[Deprecated features 4D v12 and higher](#)" (このバージョンには"変換"ドキュメントはありませんでした)
- **4D v11:** "[Conversion to 4D v11 SQL](#)".

- 📖 変換の原則
- 📖 新しい互換性オプション
- 📖 振る舞いの変更(全体)
- 📖 振る舞いの変更(ランゲージ)
- 📖 名前の変更、テーマの変更
- 📖 廃止予定の機能
- 📖 無効化された機能
- 📖 付録: 変換に有用なメソッド

## 変換の原則

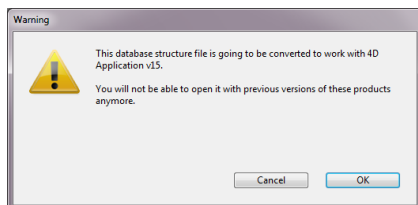
### 変換の前にはしておくべきこと

- 変換を行うためには、データベースの"インタープリター"版(ストラクチャーのxxxx.4DB ファイル)と、デザイナーパスワードが必要になります。
- 変換の前に、必ずデータベースのコピー(バックアップ)を作成して下さい。
- シンタックスチェックを実行します。データベースをコンパイルしない場合も、このチェックによって起こり得るエラーを検知することができます。
- **Maintenance and Security Center(MSC)** を使用してストラクチャーとデータの検査と修復を行って下さい。
- 4D Pack **AP Is Picture Deprecated** コマンド(v13.2から採用)を使用して、データベース内にPICTファイルがないかどうかをチェックし、あった場合には**TRANSFORM PICTURE** コマンドを使用して変換します(4D v14では、**SET DATABASE PARAMETER** コマンドの新しいセクターによってQuickTimeがまだ使用されている可能性があります)。
- (任意)データのジャーナリングが必要な場合、プライマリーキー(v13.4以降実装)を実装することができます(デザインリファレンス マニュアルの**主キーを設定、削除する**を参照して下さい)。
- v13.5以降のデータベースにおいて、**重複不可**属性を持つフィールドは**インデックスが必須**となりました。今後はインデックスがついてない**重複不可**属性のフィールド内では一切レコードを作成/編集することはできなくなります。レコードを保存しようとする、エラーが生成されます(-9998 重複不可のレコードが存在します・1088 インデックスが無効または未設定です)。存在しないインデックスを作成する場合、またはインデックスがなされていないフィールドを全てまとめたディスクファイルを生成する場合、付録"**変換に使用されるメソッド**"を参照して下さい。

### 変換の方法

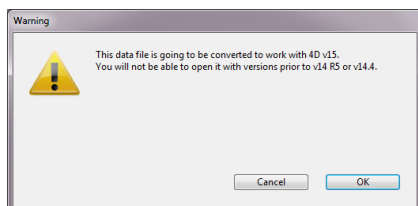
v14の4D または4D Server (v11、v12、v13 も同様)を使用して作成されたデータベースは4D v15において互換性があります(ストラクチャーとデータファイル)。どんなインタープリタ版のストラクチャーファイルも変換することができます。変換するためには、4D v15を起動し、そのストラクチャーファイル(xxx.4DB ファイル)をインタープリタモードで開くだけです。

ストラクチャーファイルが変換されることを警告するダイアログが表示されます:



ストラクチャーファイルが4D v15に変換されたあとは、その前のバージョンで開くことはできなくなります。

その後二つ目のダイアログが表示されます:



次にデータファイルがv15用に変換されますが、こちらは4D v14R5またはv14.4を使用すれば引き続き開くことができます。

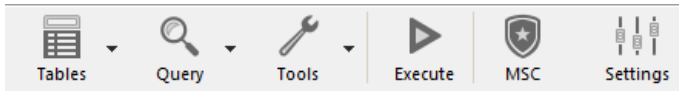


## 新しい互換性オプション

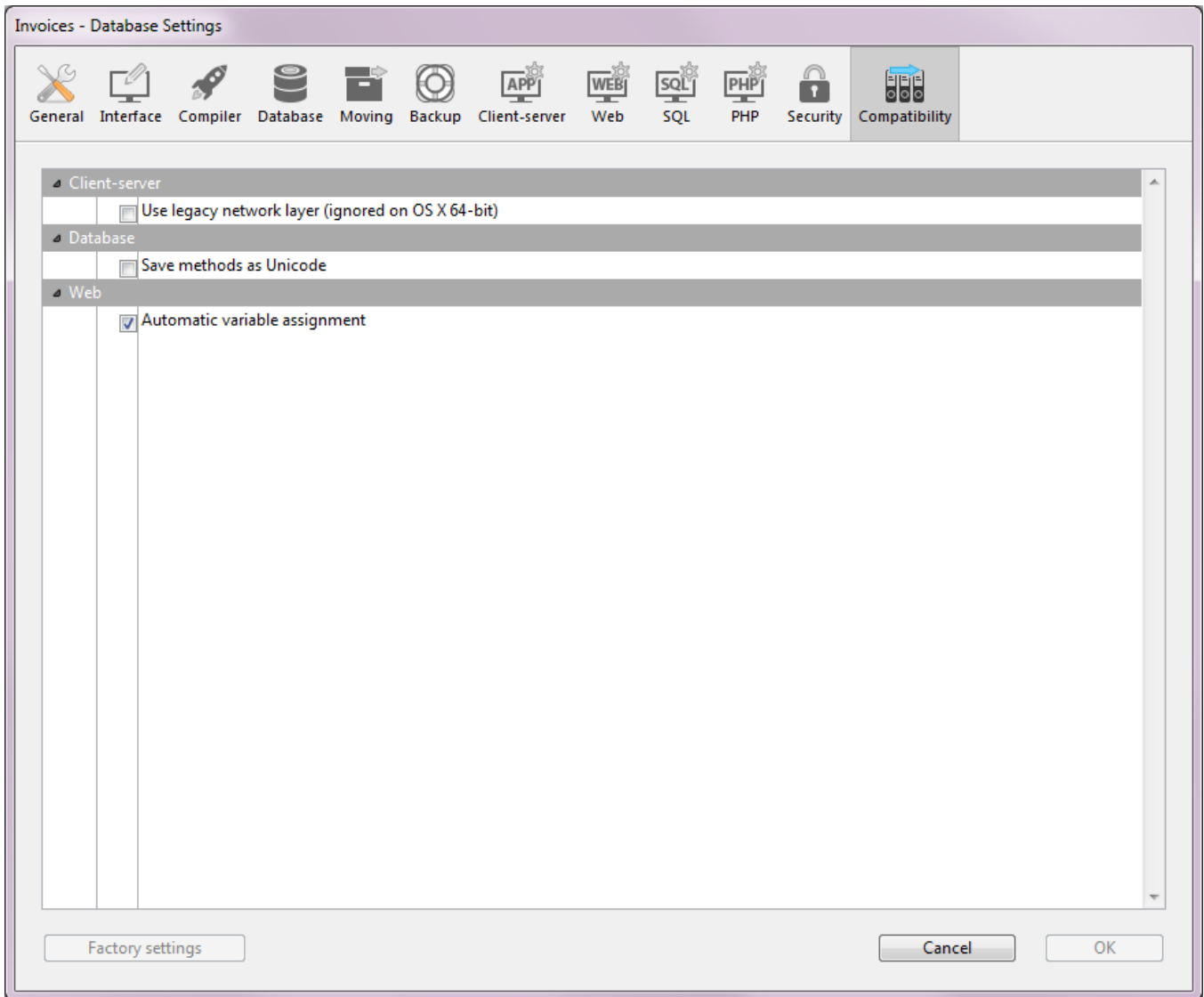
### 互換性タブ

4D v15では、互換性タブに二つの新しいオプションが追加されました。

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



その後"互換性"のタブをクリックします:



新しいオプションとは、以下の二つです:

#### 1 - メソッドをUnicodeとして保存

4D v15でデータベースを作成すると、その中ではメソッドは自動的にUnicodeモードで保存されます。変換されたデータベースでもUnicodeで保存するためには、データベース設定の"互換性"ページの**メソッドをUnicodeとして保存**オプションをチェックする必要があります。

メソッドをUnicode対応にするためには、ポインター表現の仕組みが拡張される必要がありました。ポインターは最適化され、二次元配列要素等の更なる要素をサポートするようになりました。以前コンパイルされたコンポーネントとプラグインとの互換性を保つために、4Dによって透過的に管理される新しいポインターデータ型がランゲージに追加されました。

これにより二つのコマンドが影響を受けます:

- **RESOLVE POINTER** コマンドは、変数または一次元配列へのポインターに対しての第四引数には0ではなく-1を返す

ようになりました。

- **Get pointer** コマンドは、振る舞いが変わりました：
  - 式を使用しているものを含め、2次元配列へのポインターが可能になりました。
  - 変数の無効な名前はエラー77("不正な変数名です")を返すようになりました。以前のバージョンでは、これは受け入れられていました。
  - 余分な空白はエラーではなくなりました。

## 2 - 旧式ネットワークレイヤーを使用

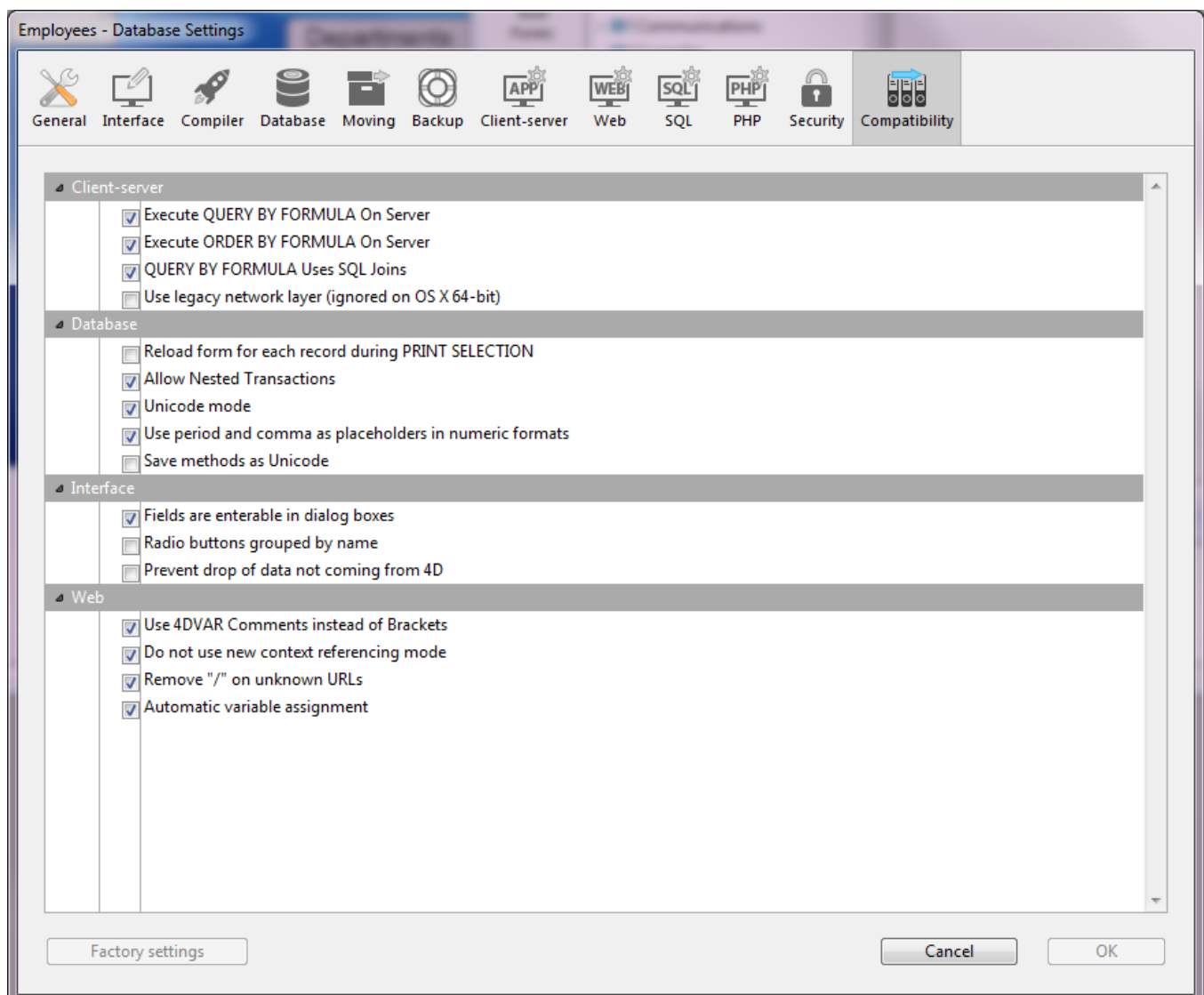
以前の"旧式"のネットワークレイヤーは廃止予定となりましたが、既存のデータベースとの互換性を確保するために残されています。4D v14 R5以降、4Dアプリケーションは4D Serverとリモートの4Dマシンとの間での通信を管理するために、**ServerNet** という新しいネットワークレイヤーを含めるようになりました。**ServerNet** レイヤーは、現代的で堅牢なAPIに基づいています。

**ServerNet** は新規に作成されたデータベースでは自動的に採用されます。

デフォルトでは、このオプションは以下のようになっています：

- 4D v14 R5以降で新規に作成されたデータベースではチェックされていません。これらのデータベースは新しい**ServerNet** レイヤーを使用します。以前のネットワークレイヤーに戻すためには、このオプションをチェックして下さい。
- 変換されたデータベースではチェックが入れています(以前のネットワークレイヤーを使用します)。このオプションはOS X 64-bitでは効力を持たないことに注意して下さい。

互換性に関するオプションはこのダイアログに含まれています。これらはバージョンごとに徐々に増えて行くので、データベースを作成したバージョンが古ければ古いほど、よりたくさんのオプションが追加されているはずで



これらのオプションに対してのより詳細な情報については、[互換性ページ](#)を参照して下さい。



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

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

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



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

このコードがサーバー上で実行された場合でも、*myvariable*変数値はクライアントマシン上のものが使用されます。他方この原則はフォーミュラにメソッドが使用され、そのメソッド内で変数が参照されている場合には当てはまりません。この場合サーバー上で変数が解釈されます。

変換されたデータベースではこの点が考慮されていない可能性があるため、デフォルトでこれらのコマンドはクライアントマシン上でフォーミュラを実行します。新しいモードを使用したい場合はこれらのオプションを明示的に選択します。

**注:** このオプションは**SET DATABASE PARAMETER**コマンドで設定することもできます。

- **QUERY BY FORMULAでSQL JOINを使用:** 4D v11より**QUERY BY FORMULA**や**QUERY SELECTION BY FORMULA**コマンドはSQLの結合モデルに基づくJOINを実行するようになりました。これによりストラクチャーエディターで自動リレーションが設定されていなくても[Table\_A ]field\_X=[Table\_B ]field\_Yのようなフォーミュラを使用できるようになりました。

既存のデータベースでこの動作が考慮されていない場合、予期しない動作となることがあるため、変換されたデータベースではこの機能がデフォルトで無効にされています。データベースコードを見直した後、このモードを有効にすることを推奨します。

**注:**

- "SQL JOIN"モードが有効な場合でも、以下のケースでは**QUERY BY FORMULA**や**QUERY SELECTION BY FORMULA**コマンドはストラクチャーエディターで設定された自動リレーションを使用します:
  - フォーミュラを{field ;comparator ;value}形式に分解できない場合
  - 同じテーブルの2つのフィールドが比較されている場合

- **SET DATABASE PARAMETER**コマンドを使用してプロセス毎にこのオプションを設定できます。

- **トランザクションのネストを許可する:** マルチレベルトランザクションのサポートを有効にします。4D v11以降、マルチレベルのトランザクションがサポートされるようになりました。既存のデータベースでこの動作が考慮されていない場合、予期しない動作となることがあるため、変換されたデータベースではこの機能がデフォルトで無効にされています(トランザクションは1レベルに制限されます)。マルチレベルのトランザクションを使用したい場合、このオプションを選択します。

**注:** このオプションはSQLエンジンで実行されるトランザクションには影響しません。SQLのトランザクションは常にマルチレベルです。

- **Unicodeモード:** カレントデータベースのUnicodeモードの有効/無効を切り替えます。Unicodeモードではデータベースエンジン、言語、メニューなどでネイティブにUnicode文字が処理されます。非Unicodeモードでは日本語環境の場合Shift\_JISが使用されます。

この設定に関わらずデータファイルはUnicodeが使用され、文字列の評価にはICUが使用されます。

v2004以前から変換されたデータベースではこのオプションが選択されていません。しかしながらこのオプションを選択し、必要なコードの修正を適用することを強く推奨します。非Unicodeモードを使用しても、過去のバージョンとの完全な互換性は提供されません。

**注:**

- このオプションの範囲はデータベースごとです。インタープリターモードではUnicodeモードのデータベースに非Unicodeモードのコンポーネントをインストールしたり、あるいはその逆を行ったりすることが可能です。
- **SET DATABASE PARAMETER**コマンドを使用してUnicodeモードを設定することができます。

4DにおけるUnicodeサポートについては**ASCIIコード**を参照してください。

- **ピリオドとカンマを数値フォーマットのプレースホルダーとして使用する:** v11以降、4Dは数値の表示フォーマットにシステムの地域設定パラメーターを使用するようになりました(**表示フォーマット**の"数値フォーマット"参照)。4Dは自動で数値表示フォーマット中の","を千の位区切り文字、"."を小数点として解釈し、システムに設定された記号で置き換えます。以前のバージョンでは数値表示フォーマットでシステムの地域設定は考慮されていませんでした。例えば"###,##0.00"フォーマットは日本語システムでは有効ですが、フランス語システムでは結果が異なっていました。変換されたデータベースでは互換性保持のためこの新しいメカニズムが無効になっています。

- **Web変数に値を自動的に代入する:** 以前のバージョンの4Dでは、Webサーバーの標準機構によって、HTTPフォームやGET type URL を使用して送信された変数の値を自動的に4Dプロセス変数へと代入をしていました。インタープリターモードでは、変数同士が同じ名前であれば、受け取った変数の値はどんな値でも4Dプロセス変数へとコピーされました。コンパイルモードでは、変数は事前に**COMPILER\_WEB** プロジェクトメソッドを使用して宣言しておく必要がありました。

v13.4以降この機構は廃止予定となり、新しいデータベースでは使用できなくなりました。変換されたデータベースではこの機能は互換性のために、残されていますが、互換性のオプションでチェックを外すことによって無効化することができます。今後は代わりに **WEB GET VARIABLES** または**WEB GET BODY PART** コマンドの使用が推奨されます。



- **旧式ネットワークレイヤーを使用する(OS X用64-bit版では使用不可):** v14 R5以降、4Dアプリケーションは4D Serverとリモートの4Dマシン間の通信に、ServerNetという新しいネットワークレイヤーを使い始めました。以前のネットワークレイヤーは廃止予定となりますが、既存のデータベースとの互換性を保つために保持はされます。このオプションを使用すると、変換された4D Serverアプリケーションにおいて、必要に応じていつでも以前のネットワークレイヤーを有効化・無効化することができます。例えば、クライアントアプリケーションを移行させるとき(設定 (環境設定) の章を参照して下さい)などに使えます。ServerNet は新規に作成されたデータベースにおいては自動的に使用され、変換されたデータベースにおいては無効化されます(このオプションがチェックされます)。この設定を変更する際には、その変更が反映されるためにはアプリケーションを再起動する必要があるという点に注意して下さい。接続していたクライアントアプリケーションも新しいネットワークレイヤーで接続するためには全て再起動しなければなりません(ServerNet を使用するために必要な最小限のクライアントのバージョンはあ4D v14 R4です。設定 (環境設定) の章を参照して下さい)。

**注:**

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

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

**注:**

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

## 振る舞いの変更(全体)

### サブレコードの変換

4D v15において(実際には4D v14 R3から)は、v11以前のバージョンで作成されたデータベースを変換する際に4Dによって自動的に追加されていた"id\_added\_by\_converter"特殊フィールドの値を自分で割り当てることが出来るようになりました。以前のバージョンでは、この値は4Dによってのみ割り当て可能だったため、変換されたサブテーブルで新しいレコードを追加するためにはデベロッパは、**\_o\_CREATE SUBRECORD** などの廃止予定のコマンドを使用しなければなりませんでした。

この新しいプロパティを使用することにより、サブテーブルを使用している古いデータベースを、より洗練された方法で変換することができます。特殊な"subtable relation"リンクを保ったまま、あたかも標準のリンクであったかの様に、リレートしたレコードを追加、あるいは編集することができます。全てのメソッドがアップデートされたあと、コードを何も変えることなく特殊なリレーションを通常のものに取り換えることができます。

例えば、以下のように記述することができます:

```
CREATE RECORD ([Employees])
[Employees]Last Name:="Jones"
CREATE RECORD ([Employees_Children])
[Employees_Children]First Name:="Natasha"
[Employees_Children]Birthday:=!12/24/2013!
```

```
[Employees_Children]id_added_by_converter:=4 //以前のリリースではここで型のミスマッチを引き起こしていました
SAVE RECORD ([Employees_Children])
SAVE RECORD ([Employees])
```

このコードは特殊なリレーション・通常のリレーションの両方において動作します。

## 重複不可属性: 自動インデックス

---

4D では、**重複不可属性**を持つフィールドは必ずインデックスが付いていなければなりません。4D v15以降、ストラクチャーエディター内にてインデックスを持っていない重複不可属性を定義することはできなくなりました。以前のリリースでは、メンテナンス目的のために、そのような設定の維持は可能でした。

## 使用不可の画像: 新アイコン

---

4D v14 R3 以降、マシン上でレンダリングが出来ないフォーマットで保存されたピクチャーに対しては新しいアイコンが表示されるようになりました。存在しないフォーマットの拡張子がアイコンの下部に表示されています:



このアイコンは同様の画像が使用されているところであればどこでも表示されます。

これはこのピクチャーがローカルでは表示または処理できないことを意味しています。ただし他のマシンでの表示のために保存することは可能です。これは例えば、WindowsプラットフォームでのPDFピクチャーや、OS X用64-bit版の4D Server でのPICTベースピクチャーなどがそれにあたります。

## メソッドエディター

---

### メソッドエディターはメソッドをUnicodeで保存する

4D v15で作成されたデータベースでは、メソッドは自動的にUnicodeで保存されます。変換されたデータベースにおいてUnicodeで保存するためには、データベース設定の“互換性”ダイアログの**メソッドをUnicodeで保存**オプションを選択する必要があります。

### メソッドエディターはデフォルトで"English-US"設定に

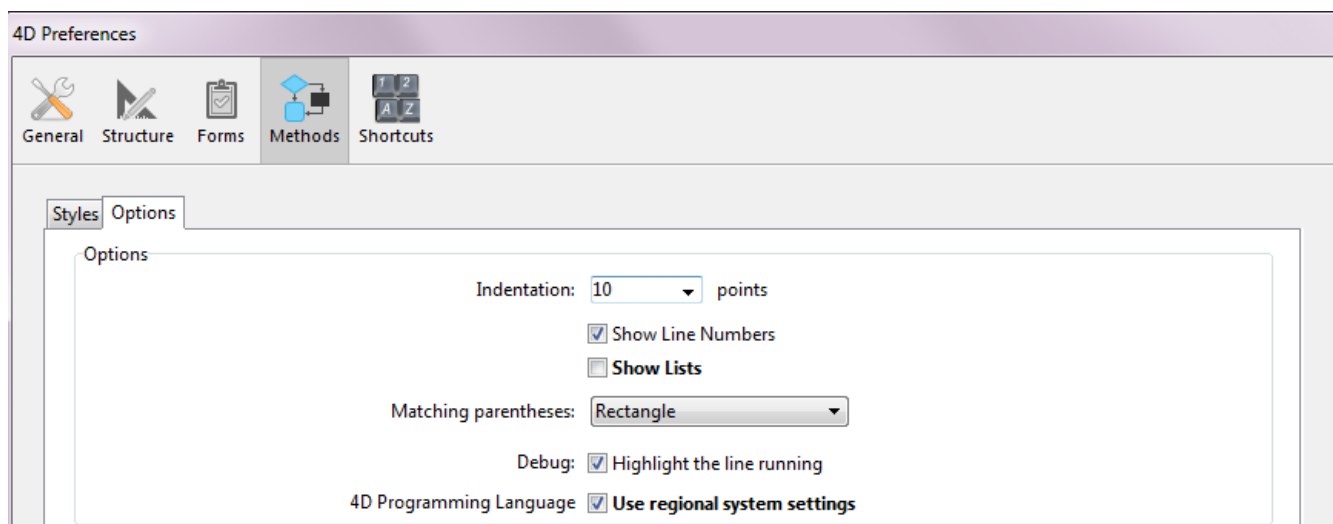
---

4D v15では、4Dのメソッドエディターは4Dのバージョンやローカルシステム設定に関わらず、デフォルトで国際的な"English-US"言語を使用します:

- 全てのバージョンにおいて、実数における小数点はピリオド(".")を使用する必要があります(カンマ(",")は使用できません)。
- 日付定数は、ISOフォーマット(!YYYY-MM-DD!)に準拠する必要があります。
- コマンド名と定数名は英語でなければなりません(他の言語ではすでにこういった仕様だったため、この変更が関係するのはフランス語版の4Dだけです)。

以前のバージョンから変換されたアプリケーションにおいては、**EXECUTE FORMULA** などの式を使用する際には注意が必要です。

この設定は、4D環境設定ダイアログボックスの**メソッド/オプション**ページの**地域特有のシステム設定を使う**オプションを使用して無効化することもできます:



配付: 地域特有のシステム設定を使うオプションはマシンローカルの設定であるため、組み込みアプリケーションには含まれていません。

組み込みアプリケーションを配付する際には、リージョン設定を使用するためには二つのソリューションがあります:

- ソリューション1: 最終的な組み込みアプリをビルドする前に、4D Volume DesktopアプリケーションのResourcesフォルダの第一階層にあるen.lprojディレクトリを削除します。
- ソリューション2: それぞれのローカルのマシン上の4D v15の設定ファイルの中身を編集し、**"use\_localized\_language"** キーを **"true"** に設定します。  
このサンプルコードは付録に記載があります。これを実行した後、この変更が有効になるためには4Dアプリケーションを再起動する必要があるという点を忘れないで下さい。

より詳細な情報については、アップグレードマニュアルの[メソッドエディターでのEnglish-US設定](#)の章を参照して下さい。

## Windowsのテンキーの"."キーの使用について

4D v14 以降、Windows PCにおいて、テンキーの"小数点"キーがピリオド[.]であったとき、数値型フィールドとテキスト型フィールドとでこのキーを使用したときの結果が異なります:

- **4D v13 以前:** フィールドが数値型かテキスト型かに関わらず、テンキーの[.]キーを押すと、システムレベルで定義された小数点を挿入しました(この定義は4Dを起動する前になされている必要があります)。
- **4D v14 と 4D v15:** 実数型のフィールドにおいて、テンキーの[.]キーを押すと、システムレベルで定義された小数点を挿入します。他の型のフィールドにおいては、このキーは単にピリオドを挿入します。

この際は、システムの小数点がピリオドでない場合(例えばほとんどのヨーロッパ系システムなど)に限り認知可能なものでした。

**注:** 変換されたデータベースでの数値のフォーマットについての詳細な情報に関しては、[互換性ページ](#)の章の"数値フォーマットにおいてピリオドとカンマをプレースホルダーとして使用する"を参照して下さい。

## 実数からテキストへの変換: 有効桁数の縮小

4D での実数の小数部分を表すのに使用される有効桁数は、以下の様に削減されました:

- 以前のバージョンでは、この桁数は15桁でした。
- 4D v15以降のバージョンでは**13桁**になりました。

この変更が関係するのは実数をテキストに変換する際に限られ、内部での表現(保存)、または実数同士の計算には関係しません。つまり、実数の正確性に影響はないという事です。この新しい原理は、テキストで表示した際に正確性の保証できない最後の2桁を含めない、というものです。この目的は、実数の演算が誤ったテキストとして結果が返ってくる状況を限定することです。例えば、この原理により、以下の様な場合に有効な結果を得ることができます:

演算	v14 R3以前の4Dでの結果	v14 R3以降の4Dでの結果
String(3216.36 - 3214.89)	"1.470000000000025"	"1.47"
String(0.321636-0.321489)	"0.000146999999999953"	"0.000147"

実数はその性質から正確性に限界があるために、もしあなたの4Dアプリケーションが実数の最初の15桁またはそれ以上を使用する場合(シリアル番号や天文学的数字等)、データフォーマットをテキストまたは倍長整数に変換する必要がでてくる可能性があります。

4Dでは、この比較のデフォルトの正確性は **SET REAL COMPARISON LEVEL** コマンドを使うことによって変更することができます。

## 4Dコードでのポインター表現の再デザイン

メソッドをUnicodeに変更した影響により、4Dコードでのポインター表現を変更する必要性がありました。ポインターは最適化され、二次元配列要素などの特別な機能をサポートするようになりました。以前にコンパイルされたコンポーネントやプラグインとの互換性を保つことは重要であるため、4Dによって等価的に管理される新しいポインターデータ型がランゲージに追加されました。

これにより、二つのコマンドが影響されました:

- **RESOLVE POINTER** はポインターから変数、あるいは一次元配列への際の第四引数には0ではなく-1を返すようになりました。
- **Get pointer** は振る舞いが変更になりました:
  - ポインターから、式を使用しているものも含め、二次元配列へも可能になりました。
  - 不正な変数名はエラー77("不正な変数名です")を返すようになりました。以前のバージョンでは、これらは使用可能でした。
  - 余分な空白はエラーとはならなくなりました。

## **\_o\_CREATE SUBRECORD**

`_o_CREATE SUBRECORD ( subtable )`


引数	型	説明
subtable	サブテーブル	⇒ 新しいサブレコードを作成するためのサブテーブル

### **互換性に関するメモ**

---

バージョン11以降の4Dはサブテーブルをサポートしていません。互換性メカニズムは、変換されたデータベースでコマンドの機能を保護しますが、すべてのサブテーブルは、リレートする標準的なテーブルに取り換えられることが強く推奨されます。

EXECUTE FORMULA ( statement )

引数	型	説明
statement	文字 	実行するコード

## 説明

**EXECUTE FORMULA** は *statement* をコードとして実行します。ステートメントの文字列は必ず1行だけです。*statement* に空の文字列を指定した場合、**EXECUTE FORMULA** コマンドは何も行いません。

ルールは、*statement* が一行のメソッドとして実行されるかぎり、それは正しく実行されます。**EXECUTE FORMULA** は実行速度を低下させるので、代替手段として利用します。コンパイル済みデータベースにおいても、そのコードはコンパイルされていません。つまり *statement* は実行されますが、コンパイル時にコンパイラによるチェックはされません。

*statement* には以下を指定できます：

- プロジェクトメソッドの呼び出し
- 4D コマンドの呼び出し
- 代入

フォーミュラにはプロセス変数とインタープロセス変数を含めることができます。しかし *statement* は1行でなければならないため、(**If**, **While**, などの) フローコントロールを含めることはできません。

## 例題

データベースのすべてのテーブルについて、各テーブルの標準データ入力に使用するための“INPUT FORM”というフォームがあります。そこで、ポインタまたはテーブル名を渡すテーブルのカレント入力フォームとしてこのフォームを設定する、汎用的なプロジェクトメソッドを追加する場合には、次のように記述します：

```

` STANDARD INPUT FORM project method
` STANDARD INPUT FORM ( ポインタ {; 文字 })
` STANDARD INPUT FORM ( ->Table {; TableName })
C_POINTER($1)
C_TEXT($2)

If(Count parameters>=2)
    EXECUTE FORMULA(Command name(55)+"(["+$2+"]";"+Char(Double quote)+"INPUT FORM"+Char(Double
quote)+")")
Else
    If(Count parameters>=1)
        FORM SET INPUT($1->"INPUT FORM")
    End if
End if

```

このプロジェクトメソッドがデータベースに追加された後、以下のように記述します：

```

STANDARD INPUT FORM(->[Employees])
STANDARD INPUT FORM("Employees")

```

**Note:** 通常、汎用ルーチンを記述する場合には、ポインタを使用することをお勧めします。その理由としては、まず、データベースがコンパイルされているとき、そのコードもコンパイルモードで実行されるからです。次に前述の例でもあるようにテーブルの名前を変更すると、コードは正しく動作しなくなるためです。ただし、**EXECUTE FORMULA** コマンドを使用すれば問題が解決する場合があります。

## メソッドエディターでのEnglish-US設定

4D v15より、4Dメソッドエディターは、4Dのバージョンやローカルのシステム設定値に関わらず、デフォルトで国際的な"English-US"言語を使用します。この新しい特徴により、4Dアプリケーション間でのコード解釈を妨げうるリージョン間での差異(例えば日付フォーマットなど)を全て統一します。またフランス語版の4Dであっても、コマンドと定数は常に"English-US"で書かれます。

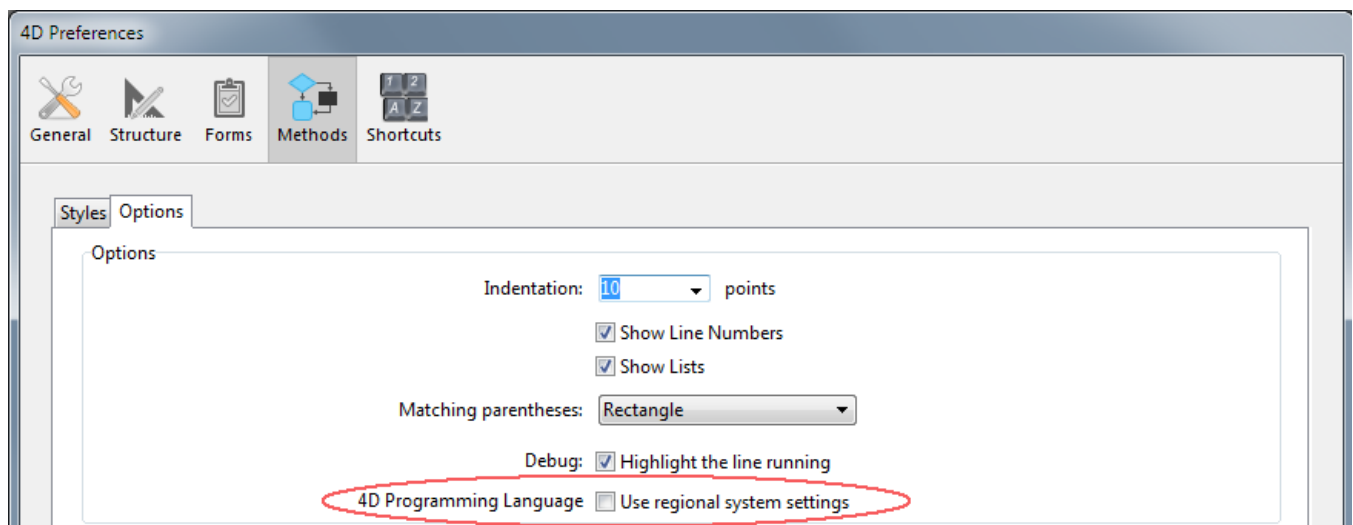
この新しいデフォルト設定により、4Dデベロッパには主に二つの利点があります：

- 国、地域と言語の設定、使用している4Dのバージョンに関わらず、4Dデベロッパー間でのコードの共有が簡単になります。4Dメソッドは単純なコピー/ペースト、またはテキストファイルへの保存だけで、互換性の問題なく交換することが可能になりました。
- また、ソースコントロールツールに4Dメソッドを含めることが可能になりました。これは一般的にリージョン設定や言語とは関係なく書き出される必要があるものです。

この新しい設定は4D設定ダイアログボックス内の新しいオプションにより無効化することもできます。

### 新しい設定オプション

新しい**地域特有のシステム設定を使う**オプションは、4D環境設定ダイアログボックス内のメソッド/オプションタブを有効化/無効化することができ、"国際的な"コーディングを可能にするものです：



このオプションがチェックされていない(4D v15でのデフォルト設定)場合、カレントの4Dアプリケーションの4Dメソッド内にてEnglish-US設定が使用されます。このオプションがチェックされていた場合以前の4Dのバージョン同様、カレントの4Dアプリケーションの4Dメソッド内ではリージョン設定が使用されます。

このオプションを変更した場合、変更が有効になるためには4Dアプリケーションを再起動する必要があります。

### Englis-US設定では何が変わるのか?

新しいEnglish-US設定の採用により、メソッドの書き方に影響がでる可能性があります。これは開発モードで書かれるコードに加え、配付されたアプリケーションでのフォーミュラにも関係します。

この新しいモードにおいては、コードは以下のルールに従う必要があります：

- 実数値の小数点は、全てのバージョンに置いてピリオド(".")が使用されます(例えばフランス語版で一般的に使用されていたカンマ(",")は使用されません)。
- 日付定数は全てのバージョンにおいてISOフォーマットに準拠する必要があります。
- コマンド名と定数名は英語である必要があります(これはフランス語版の4Dにのみ影響します。他の言語ではもともとこのルールが適用されていました)。



注: メソッドエディターには、必要に応じて不正な入力を自動的に修正するメカニズムが搭載されています。

以下の一覧は4D v15と以前のバージョンのコードでの違いをまとめたものです:

	メソッド/フォーミュラでのコード例
4D v15 (デフォルトモードの全てのバージョン)	a:=12.50 b:=!2013-12-31! Current date
4D v14 または 4D v15 (設定がチェックされている、USバージョン等)	a:=12.50 b:=!12/31/2013! Current date
4D v14 または 4D v15 (設定がチェックされている、フランス語版)	a:=12,50 b:=!31/12/2013! Date du jour

注: 4D の以前のバージョンと、設定がチェックされている4D v15においては、実数と日付のフォーマットはシステム設定によって変化します。

## フランス語版における変更点

一部の特定の変更は、フランス語版の4Dにのみ影響します。このバージョンにおいては、ランゲージ(コマンド名と定数名)とオンラインヘルプ、そしてドキュメントにおいて、フランス語の名前を使用してきました。

より詳細な情報は、このマニュアルのフランス語版にまとめてあります。

## 互換性に関する問題

以前のバージョンから変換されたアプリケーションに付いては、4D v15での4DコードのEnglisu-US設定を使用するために変更を加えなければならない場合があります。例えばEXECUTE FORMULA など、コードがオンザフライで解釈され、トークナイズされていない場合には問題が起こる可能性があります。これは4Dの開発モードだけではなく、配付された製品(リモートモードの4Dまたは組み込まれた4Dアプリケーション)にも関係する問題です。

以下の一覧は4D v15での影響を言語ごとにまとめたものです(ここに記載のないその他のバージョンは、US版と同じように影響を受けます):

機能	4Dの言語設定がUSの場合の影響	4Dの言語設定がFRの場合の影響	4Dの言語設定がDEの場合の影響
フォーミュラエディターの適用	フォーミュラ内: 日付フォーマットのみ	フォーミュラ内: コマンド言語(US)と日付/時刻/小数点	フォーミュラ内: 日付フォーマット+小数点
クイックレポート	フォーミュラ内: 日付フォーマットのみ	フォーミュラ内: コマンド言語(US)と日付/時刻/小数点	フォーミュラ内: 日付フォーマット+小数点
4D Write	フォーミュラ内: 日付フォーマットのみ	フォーミュラ内: コマンド言語(US)と日付/時刻/小数点	フォーミュラ内: 日付フォーマット+小数点
4D View	なし	フォーミュラ内(4Dコマンドを使用する場合、4D Viewコマンドは含めない): コマンドランゲージ(US)	なし
PROCESS 4D TAGS	日付フォーマットのみ	コマンド(Cxxxを使用していない場合) 日付/時刻/小数点	日付フォーマット+小数点
EXECUTE FORMULA	日付フォーマットのみ	コマンド言語(US)と日付/時刻/小数点	日付フォーマット+小数点
METHOD GET CODE/METHOD SET CODE	日付フォーマットのみ	コマンド言語(US)と日付/時刻/小数点	日付フォーマット+小数点

必要があれば、新しい地域特有のシステム設定を使うオプションをチェックすることで4D v14の振る舞いへと戻せる、ということに注意して下さい。

## 配付について

---

この設定はマシンごとにローカルに保存されるため、リージョン設定を使用したい場合には、4Dアプリケーションを実行するコンピューターにおいてそれぞれ設定しなければなりません。組み込みアプリのコンテキストに置いては、それぞれのマシンにおいて4D v15設定ファイルを編集した上で、"use\_localized\_language"キーを"true"へと設定しなければなりません。

**注:** リージョン設定を配付されたアプリケーションにおいて使用するためのソリューションは、このマニュアルのフランス語版において提供されています。

## SET REAL COMPARISON LEVEL

SET REAL COMPARISON LEVEL ( epsilon )

引数	型	説明
epsilon	実数 →	実数の同等性を比較するためのイプシロン値

### 説明

**SET REAL COMPARISON LEVEL** コマンドは、実数値と式の同等性を比較するために4Dが使用するイプシロン値を設定します。

コンピュータは常に実数を近似値で計算するため、実数の同等性をテストする時には、この近似値を考慮する必要があります。4Dは、実数を比較する時に2つの実数の差が一定の値より大きいかどうかをテストすることによって、近似値を確認します。この値は**イプシロン値**と呼ばれ、以下のように動作します:

2つの実数aとbがある時、**Abs(a-b)**がイプシロン値より大きい場合、これら2つの数値は等しくないとみなされます。それ以外の場合には等しいとみなされます。

デフォルトで、4Dはイプシロン値を10の-6乗 ( $10^{-6}$ ) に設定しています。イプシロン値は、常に正数を指定してください。例えば:

- $0.00001=0.00002$  はFalseを返します。なぜなら違いは0.00001であり、これは $10^{-6}$ より大きいからです。
- $0.000001=0.000002$  はTrueを返します。なぜなら違いは0.000001であり、これは $10^{-6}$ より大きくないからです。
- $0.000001=0.000003$  はFalseを返します。なぜなら違いは0.000002であり、これは $10^{-6}$ より大きいからです。

**SET REAL COMPARISON LEVEL**を使って、必要に応じて、エプシロン値を増大させるか、減少させることができます。

**Note:**  $10^{-6}$ より小さい値が納められた数値タイプのインデックス付きフィールドに対してクエリや並び替えを実行したい場合、インデックスの構築の前に必ず**SET REAL**

**COMPARISON LEVEL**コマンドを実行してください。

**警告:** 通常、デフォルトのイプシロン値を変更するためにこのコマンドを使用する必要はありません。

**重要:** イプシロン値を変更しても、実数の同等性の比較に影響があるだけで、他の実数計算や実数値の表示には影響はありません。

## 振る舞いの変更(ランゲージ)

### SET EXTERNAL DATA PATH

---

**SET EXTERNAL DATA PATH** は指定したファイルが存在しない場合に、ファイルを作成しないようになりました。代わりにパス名を保存するようになりました。

### MAXIMIZE WINDOW (Windows環境下)

---

Windows での**MAXIMIZE WINDOW** コマンドは、フォーム上で指定されたサイズ制約を考慮するようになりました。そのため結果は、フォーム上(フォームウィンドウの場合)あるいはターゲットウィンドウ(MDI ウィンドウの場合)で指定されたサイズに応じて変化します:

- サイズ制約がターゲットサイズより大きい場合、ウィンドウは以前のバージョンの4Dと同様に"最大化"されます。つまり、ウィンドウは親"Multiple Document Interface" (MDI) ウィンドウのサイズに一致するようにリサイズされるということです。タイトルバーと境界線は非表示となり、コントロールボタン(最小化、復元、閉じる)はアプリケーションメニューバーの右側に配置されます。
- サイズ制約の少なくともどれか一つがターゲットサイズより小さい場合(例えば、MDIウィンドウの幅が100でフォームウィンドウの最大幅が80だった場合)、ウィンドウは"最大化"されず、その代わりに許可された範囲内での最大のサイズへとリサイズされます。

### OBJECT GET COORDINATES / LISTBOX GET CELL COORDINATES

---

已然のバージョンの4Dにおいては、リストボックスに対して使用される**OBJECT GET COORDINATES** コマンドは *object* 引数で指定されるパーツと関係なく、リストボックス自身の座標のみを返していました。例えば、*object* 引数がヘッダーを指定していた場合、**OBJECT GET COORDINATES** コマンドは単にリストボックスの座標を返すのみでした。

4D v15以降、*object* 引数がリストボックスのヘッダー、カラム、フッターを参照している場合、このコマンドは指定されたオブジェクトの座標を返します。

そのため、リストボックス自身の座標を取得したい場合には、コードの修正が必要になる可能性があるという点に注意して下さい。

統一性の観点から、座標の原点は変わりません。つまり、オブジェクトを内包するフォームの左上端が原点となります。リストボックスのサブオブジェクトを取得した場合の結果についての詳細な情報に関しては、**OBJECT GET COORDINATES** コマンドのドキュメントを参照して下さい。

また、4D v15から追加された新しいコマンドも参照して下さい: **LISTBOX GET CELL COORDINATES**

### PROCESS 4D TAGS

---

以前のバージョンの4Dでは、インタープリタモードでは、**PROCESS 4D TAGS** 実行コンテキストにおいて、呼び出し時に定義されたローカル変数はアクセス可能でした。v15ではこれがアクセス不可になりました。

**PROCESS 4D TAGS** コマンドはどんな型(テキスト、データ、倍長整数、実数、他)にもなりうる未定義の数の引数を受け取れるようになりました。配列ポインターを介して配列も使用できるようになりました。4Dメソッドでそうであるように、これらの引数は通常の引数(\$1、\$2、他)を通して使用できます。


(他の新しい変化: 新しい4DEVAL タグが使用可能になり、4DLOOP タグはポインターを受け付けるようになりました。)

### WA GET/SET PREFERENCE (新しいデフォルトの振る舞い)

---

**WA SET PREFERENCE** と **WA GET PREFERENCE** コマンドはWebエリアにURLまたはファイルをドロップすることを許可する新しいセクターを受け入れるようになりました(これはURLがロードされる前、例えばOn Load フォームイベントなどで設定する必要があります)。

セキュリティ上の理由により、Webエリア内にファイルまたはURLをドロップすることによりエリアのコンテンツを変えることは、デフォルトでは不許可となっています。

ユーザーがファイルまたはURLをエリアにドロップしようとした場合には禁止アイコン  がマウスカーソルに表示されます。

(以前のバージョンでは、そのようなアクションを禁止するためには、例えば**WA SET URL FILTERS** のような特定のフィルターをインストールする必要がありました)。

この機能を許可するためには、新しい[WA enable URL drop](#) Webエリア設定を使用する必要があります。

SET EXTERNAL DATA PATH ( aField ; path )

引数	型	説明
aField	テキスト, BLOB, ピクチャー, Object	→ ストレージの場所を設定するフィールド
path	テキスト, 倍長整数	→ 外部ストレージのパス名およびファイル名、または 0 = ストラクチャー定義を使用する 1 = デフォルトフォルダーを使用する

### 説明

**SET EXTERNAL DATA PATH** コマンドは *aField* 引数に渡したフィールドの、カレントレコードの、外部ストレージの場所を設定あるいは変更します。

4Dではテキスト、BLOB、ピクチャーおよびオブジェクト型のフィールドデータをデータファイルの外部に格納することができます。この機能に関する詳細な説明は *Design Reference* マニュアルを参照してください。

このコマンドで指定される設定は、カレントレコードがディスクに保存されるときにのみ適用されます。カレントレコードがキャンセルされると、コマンドはなにも行いません。アプリケーションストラクチャーに設定されたパラメーターは変更されません。

*path* にはカスタムパス名または自動的な場所を指定する定数いずれかを渡すことができます:

#### ● ファイルへのカスタムパス名

この場合外部ストレージをカスタムモードで使用するようになります。特定の4Dデータベース機能はこのモードを自動では利用できません (*Design Reference* マニュアル参照)。特に、ファイルの作成と変更は自分自身で管理する必要があります。

相対パス、または絶対パスを渡す事ができます。絶対パスの場合、パス内にストレージファイルの名前と拡張子を含んでいる必要があります(相対パスを指定する場合、文字列の最初に"./" (Windows) または "../" (OS X) を挿入します)。拡張子は実際のデータ型と一致しなければなりません(保存時に自動で変換されることはありません)。システムシンタックスを使用しなければなりません。データベース外部ファイル (*databaseName.ExternalData*) のデフォルトフォルダーを含むどのフォルダーでも指定できます。この場合、これらのファイルはデータベースが保存されるときに含まれます。

*path* 引数によって指定されたファイルはコマンド実行時に存在しかつアクセス可能である必要があります。フォルダーまたはファイルが存在しない場合、-43エラー("ファイルが見つかりません")が返されます。

外部ファイルをデータファイルと同階層か、そのサブフォルダーに保存する場合、4Dは指定されたパスがデータファイルに対し相対的であるとみなし、データファイルフォルダーが移動されたり名称変更されたりした場合でもそのリンクを保守します。

これにより複数のレコードで同じ外部ファイルを共有することが可能な点に留意してください。この外部ファイルに対して行われた変更はすべてのレコードに対して有効です。この場合、複数のプロセスが同時に同じフィールドを変更できるならば、セマフォを使用して同時アクセスを制限しなければなりません。そうでなければ外部ファイルが損傷するリスクがあります。

#### ● 自動的な場所

**Data File Maintenance** テーマの、以下の2つの定数を指定できます:

定数	型	値	コメント
Use default folder	倍長整数	1	引数として渡されたフィールドのデータは <i>databaseName.ExternalData</i> という名前のデフォルトフォルダーに保存されます。このフォルダーはデータファイルと同階層に作成されます。このモードでは、外部データを、それがデータファイル内にあるときと同様に、4Dが管理します。
Use structure definition	倍長整数	0	4Dはストラクチャーに設定されたフィールドデータの格納設定を使用します。外部ストレージから内部ストレージに変更しても、外部ファイルは削除されません。

一度コマンドが実行されると、4Dは自動でレコードフィールドとディスク上のファイルとのリンクを保守します。 *path* 引数

を変更したい場合を除き、再度コマンドを実行する必要はありません。4Dがフィールドのデータに(ストレージファイルの名前が変更された、ファイルが削除された、パスが変更された、等で)アクセスできなくなると、このフィールドは空になりますがエラーは生成されません。

**注:** **SET EXTERNAL DATA PATH**コマンドは4Dローカルモードまたは4D Serverでのみ実行できます。リモートモードの4Dではなにも行いません。

## 例題

---

ピクチャーフィールド内の既存のファイルを、データベースのデータファイル外に相対パスを使って保存したい場合を考えます:

```
CREATE RECORD ([Photos])
[Photos]Name:="Paris.png"
SET EXTERNAL DATA PATH ([Photos]Thumbnail;Get 4D folder(Database folder)+"custom"+Folder
separator+[Photos]Name)
//"/custom/Paris.png" ファイルはストラクチャーファイルのすぐ隣になければならない
SAVE RECORD ([Photos])
```



OBJECT GET COORDINATES ( { \* ; } object ; left ; top ; right ; bottom )

引数	型	説明
*	演算子	⇒ 指定時, Objectはオブジェクト名 (文字列) 省略時, Objectはフィールドまたは変数
object	フォームオブジェクト	⇒ オブジェクト名 (* 指定時), または フィールドまたは変数 (* 省略時)
left	倍長整数	⇒ オブジェクトの左座標
top	倍長整数	⇒ オブジェクトの上座標
right	倍長整数	⇒ オブジェクトの右座標
bottom	倍長整数	⇒ オブジェクトの下座標

### 説明

**OBJECT GET COORDINATES** コマンドは、引数 \* および *object* によって指定された、現在のフォームのオブジェクトの *left*, *top*, *right* および *bottom* の座標 (ポイント) を返します。

オプションの \* 引数を指定した場合、*object* はオブジェクト名です (文字列)。オプションの \* 引数を省略すると、*object* はフィールドまたは変数です。この場合、文字列ではなくフィールドまたは変数参照 (フィールドまたは変数のみ) を指定します。

*object* にオブジェクト名を渡し、そこでワイルドカード文字 (“@”) を使用して1つ以上のオブジェクトを指定する場合、返される座標は関連するすべてのオブジェクトで構成される四角の座標となります。

**Note:** バージョン6.5からは、文字列に含まれるワイルドカード文字 (@) の取り扱い方を設定することができます。このオプションは、“オブジェクトプロパティ”コマンドに影響を与えます。4D Design Reference マニュアルを参照してください。オブジェクトが存在しない場合やコマンドがフォーム内で呼び出されていない場合、座標(0;0;0;0)が返されます。

リストボックスのコンテキストにおいては、**OBJECT GET COORDINATES** コマンドはリストボックスの親オブジェクトの座標だけではなく、特定のリストボックスのパーツ、つまりカラム、ヘッダー、フッターなどの座標も返すことができます。v14 R5以前のバージョンの4Dにおいては、このコマンドは引数として渡されたエリアに関係なく、常に親リストボックスの座標のみを返してきました。今後は、*object* 引数で参照されたオブジェクトがリストボックスのヘッダー、カラム、フッターなどのサブオブジェクトである場合には、返される座標はそれらの指定されたサブオブジェクトの座標となります。この新機能を使用して例えば、リストボックスヘッダーにマウスオーバーしたときに小さなアイコンを表示し、ユーザーがそれをクリックするとコンテキストメニューが表示される、というような事ができるようになります。

統一性のために、オブジェクトがリストボックスサブオブジェクトまたはリストボックスオブジェクトの場合には、使用される参照フレームは同じになります。つまり原点はオブジェクトを含むフォームの左上端の角になります。リストサブオブジェクトの場合、返される座標は理論値になります。つまり、クリッピングが起こるまではリストボックスのスクロール状態を考慮するという事です(親リストボックスの座標によるカッピングは継続されるという事です)。結果として、サブオブジェクトはその座標において(一部または全体が)非表示になっていることもあり、またそれらの座標はフォームの限界の外側(あるいは負の値)になることもあります。サブオブジェクトが表示されているかどうか(そして表示されているのであればどの部分が表示されているのか)を調べたい場合、返された座標とリストボックスの座標を比較する必要があります。その際、以下のルールが適用されます:

- 全てのサブオブジェクトは、親リストボックスの座標(リストボックスに対して**OBJECT GET COORDINATES** を使用して返された値)に基づいてクリップされます。
- ヘッダーとフッターサブオブジェクトはカラムの中身の上に表示されます。カラムの座標がヘッダーとフッターの線の座標と交差した場合、カラムはその交点では表示されません。
- ロックされたカラムの要素はスクロール可能なカラムの要素の上に表示されます。スクロール可能なカラム内の要素の座標がロックされたカラムの要素の座標と交差した場合、スクロール可能なカラムの要素はその交点では表示されません。

例えば、以下の画像のように、座標が赤い長方形で縁どられた *Capital* カラムの場合を考えます:





## LISTBOX GET CELL COORDINATES

LISTBOX GET CELL COORDINATES ( { \* ; } object ; column ; row ; left ; top ; right ; bottom )

引数	型	説明
*	演算子	➡ If specified = object is the name of the object (string) If omitted = object is a variable
object	フォームオブジェクト	➡ Object name (if * is specified) or variable (if * is omitted)
column	倍長整数	➡ Column number
row	倍長整数	➡ Row number
left	倍長整数	← Left coordinate of the object
top	倍長整数	← Top coordinate of the object
right	倍長整数	← Right coordinate of the object
bottom	倍長整数	← Bottom coordinate of the object

### 説明

**LISTBOX GET CELL COORDINATES** コマンドは引数 \* および *object* によって指定されたリストボックス内の、*column* と *row* 引数で指定したセルの *left*、*top*、*right* および *bottom* にそれぞれ左端、上端、右端、下端の座標を(ポイント単位で)返します。

任意の \* 引数を指定した場合、*object* はオブジェクト名です(文字列)。任意の \* 引数を省略すると、*object* はフィールドまたは変数です。この場合、文字列ではなくフィールドまたは変数参照(フィールドまたは変数のみ)を指定します。

**OBJECT GET COORDINATES** コマンドとの統一性のため、原点はセルを含むフォームの左上端になります。また返される座標は理論値となります。この値は、クリッピングが起こるまではスクロールを考慮に入れます。結果として、そのセルは表示されていない(または一部しか表示されていない)こともあり、座標の位置はフォームの範囲を超えている(負の数値が返される)こともあります。セルが表示されているか(また、表示されているならどの部分が表示されているか)を調べるためには、返された座標と、リストボックスの座標を比較する必要があります。その際、以下の点に注意する必要があります:

- 全てのセルは、親のリストボックスの座標(リストボックスに対しての **OBJECT GET COORDINATES** の値)に沿ってクリップされています。
- ヘッダー・フッターのサブオブジェクトは、列のコンテンツの上に表示されています。セルの座標がヘッダーやフッターの線と交差する場合には、その部分のセルは表示されません。
- ロックされた列の要素は、スクロール可能な列の要素の上に表示されます。スクロール可能な列の要素がロックされた列の要素と交差するとき、スクロール可能な列の要素は表示されません。

より詳細な情報に関しては、**OBJECT GET COORDINATES** コマンドの詳細を参照して下さい。

### 例題

リストボックス内の選択されたセルの周りに赤い長方形を描画する場合を考えます:

```
OBJECT SET VISIBLE (*;"rectangleInfo";False) //赤い長方形を初期化
//長方形はフォーム内のどこかに既に定義済み
LISTBOX GET CELL POSITION (*;"LB1";$col;$row)
LISTBOX GET CELL COORDINATES (*;"LB1";$col;$row;$x1;$y1;$x2;$y2)
OBJECT SET VISIBLE (*;"RedRect";True)
OBJECT SET COORDINATES (*;"RedRect";$x1;$y1;$x2;$y2)
```



PROCESS 4D TAGS ( inputData ; outputData {; param}{; param2 ; ... ; paramN } )

引数	型	説明
inputData	テキスト, BLOB	→ 処理する4Dタグを格納しているデータ
outputData	テキスト, BLOB	← 処理されたデータ
param	テキスト, Number, 日付, 時間, ポインター	→ 実行されるテンプレートへと渡される引数

## 説明

**PROCESS 4D TAGS** コマンドを使用すると、*inputTemplate* 引数に格納されている4D変換タグ(フィールド、若しくはBLOBまたはテキスト型の変数)の処理が開始されます。*param* 引数を使用して値を挿入し(任意)、その結果が*outputResult* に返されます。これらのタグの完全な詳細については、**4D HTMLタグ** の章を参照して下さい。

このコマンドにより、タグや、4D式や変数への参照を含んだ"テンプレート"型のテキストを実行でき、それにより実行コンテキストや引数に渡された値に応じた異なる結果を生成することができます。

例えば、このコマンドにより、4D変換タグを含んだセミダイナミックページに基づいたHTMLページを生成する事ができます(このとき4D Webサーバーを起動する必要はありません)。このコマンドを使用して、データベース内のデータへの参照の処理を(4D Internetコマンド経由で)含んだHTMLフォーマットのEメールを送信する事ができます。テキストに基づいたデータタイプであれば、XML、SVG、マルチスタイルテキストなど、どんなデータタイプでも処理することができます。

処理されるタグを格納しているデータを引数 *inputTemplate* に渡します。この引数はBLOBまたはテキスト型の変数やフィールドです。テキスト型の使用が推奨されます(引数は2GB までのテキストを受け取ることができます)。

**互換性に関する注意:** 4D v12より、BLOB型の引数を使用すると、コマンドは自動でBlobに使用されている文字セットをMacRomanとして扱います。効率化のために、Unicodeモードで処理が実行されるテキスト型の引数を使用することを強く推奨します。

4Dの全ての変換タグがサポートされます (*4DTEXT*、*4DHTML*、*4DSCRIPT*、*4DLOOP*、*4DEVAL*など)。

**注:** Webサーバー (Webプロセス) のフレームワーク以外で *4DINCLUDE* タグを使用する場合:

- 4Dのローカルモードまたは4D Serverの場合、データベースストラクチャーファイルを格納しているフォルダーがデフォルトフォルダーです。
- 4Dのリモートモードの場合、4Dのアプリケーションを格納しているフォルダーがデフォルトフォルダーです。

**PROCESS 4D TAGS** コマンドは、実行されたコードに不定数の*param* 引数を挿入する事をサポートします。プロジェクトメソッド同様、これらの引数は様々なタイプのスカラー値を格納することができます(テキスト、日付、時間、倍長整数、実数、等)。また、配列ポインターによって配列を使用することもできます。4Dタグによって処理されるコードの中では、これらの引数は4D メソッド同様、標準の引数(\$1、\$2等)を通じてアクセス可能です(例題を参照して下さい)。

**PROCESS 4D TAGS** コマンドの実行コンテキストにおいて、専用のローカル変数のセットが定義されます。これらの変数は処理中、読み出し・書き込みともに可能です。

**互換性に関する注意:** 以前のバージョンの4Dでは、インタープリタモードの**PROCESS 4D TAGS** 実行コンテキスト内であれば、呼び出しコンテキストで定義されたローカル変数にアクセス可能でした。4D v14 R4以降、これはできなくなりました。

コマンドの実行後引数 *outputResult* には、*inputTemplate* 引数の結果とともに、そこに含まれる4Dタグが処理された結果が返されます。もし*inputTemplate* 引数が4Dタグを含まない場合、引数 *outputResult* の内容は引数*inputTemplate* の内容と一致します。

引数 *outputResult* はフィールドまたは変数です。ただし引数 *inputTemplate* と同じ型でなくてはなりません。

**注:** このコマンドは**On Web Authenticationデータベースメソッド**を呼び出しません。

## 例題 1

この例題ではテンプレートドキュメントをロードし、そこに含まれるタグを処理し、ファイルとして書き出します:

```
C_BLOB($Blob_x)
C_BLOB($blob_out)
C_TEXT($inputText_t)
```

```
C_TEXT($outputText_t)

DOCUMENT TO BLOB("mytemplate.txt";$Blob_x)
$inputText_t:=BLOB to text($Blob_x;UTF8 text without length)
PROCESS 4D TAGS($inputText_t;$outputText_t)
TEXT TO BLOB($outputText_t;$blob_out;UTF8 text without length)
BLOB TO DOCUMENT($document;$blob_out)
```

## 例題 2

---

以下の例は、配列のデータを使用してテキストを生成します:

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



WA SET PREFERENCE ( { \* ; } object ; selector ; value )

引数	型	説明
*	演算子	→ 指定した場合、オブジェクトがオブジェクトの名前 (文字列) 省略した場合、オブジェクトは変数
object	フォームオブジェクト	→ オブジェクトの名前 ( * を指定した場合) または、変数 ( * を省略した場合)
selector	倍長整数	→ 修正される環境設定
value	ブール	→ 環境設定の値 (True = 許可, False = 不許可)

## 説明

**WA SET PREFERENCE** コマンドを使用して、引数 \* と *object* によって指定されたWebエリアに対して、さまざまな環境設定を行います。

引数 *selector* に修正する環境設定を渡し、引数 *value* にその環境設定に割り当てられる値を渡します。引数 *selector* には、**Web Area** テーマにある以下の定数の1つを渡します。

定数	型	値	コメント
WA enable contextual menu	倍長整数	4	Webエリア内で標準のコンテキストメニューの表示を許可する
WA enable Java applets	倍長整数	1	Webエリア内でJava appletの実行を許可する
WA enable JavaScript	倍長整数	2	Webエリア内でJavaScriptコードの実行を許可する
WA enable plugins	倍長整数	3	Webエリア内でプラグインのインストールを許可する
WA enable URL drop	倍長整数	101	WebエリアへのURLやファイルのドロップを許可する(デフォルト=False)
WA enable Web inspector	倍長整数	100	Web エリア内でインスペクターの表示を許可する

各環境設定を起動するには *value* 引数に **True** を渡し、無効にするには **False** を渡します。

**注:** WebエリアでのWebプラグインとJavaアプレットの使用は**推奨されていません**。何故なら、これらは(特にイベント管理レベルにおいて)4Dのオペレーションを不安定にするおそれがあるからです。

## 例題

'myarea' というWebエリア内でURLドロップを有効化したい場合:

```
WA SET PREFERENCE (*;"myarea";WA enable URL drop;True)
```

WA SET URL FILTERS ( { \* ; } object ; filtersArr ; allowDenyArr )

引数	型	説明
*	演算子	⇒ 指定時, objectはオブジェクト名 (文字列) 省略時, objectは変数
object	フォームオブジェクト	⇒ オブジェクト名 (* 指定時) または 変数 (* 省略時)
filtersArr	文字配列	⇒ フィルタ配列
allowDenyArr	ブール配列	⇒ 許可-拒否配列

### 説明

**WA SET URL FILTERS** コマンドは、\* と *object* 引数で指定したWebエリアで、1 つ以上のフィルタを設定するために使用します。

ユーザからリクエストされたページをロードする前に、4D はフィルタのリストを照会し、ターゲットのURL に接続が許可されているかどうかを調べます。URLの判定は*filtersArr* と*allowDenyArr* 配列の内容に基づき行われます。

リクエストされたURL が許可されない場合、ページはロードされず、[On URL Filtering](#) フォームイベントが生成されます。

*filtersArr* と *allowDenyArr* 配列は同期されていなければなりません。

- *filtersArr* 配列のそれぞれの要素には、フィルタするURL が含まれます。1 つ以上の文字を表すワイルドカードとして \* を使用できます。
- *allowDenyArr* 配列のそれぞれ対応する要素には、URL を許可 (**True**) するか拒否 (**False**) するかを示すブール値が含まれます。

同じURL が許可および拒否されているなど、設定レベルで矛盾がある場合、最後の設定が考慮されます。

フィルタを無効にするには、コマンドを呼び出す際に空の配列を渡すか、配列の最後の要素で、*filtersArr* 配列に "\*" を、*allowDenyArr* 配列に **True** を渡します。

コマンドが実行されると、フィルタはWeb エリアのプロパティとなります。*filtersArr* と*allowDenyArr* が削除されたり初期化されたりしても、コマンドが再実行されるまでフィルタは有効です。エリアで有効になっているフィルタを取得するには、**WA GET URL FILTERS** コマンドを使用しなければなりません。

**重要:** このコマンドによって実行されるフィルタはWeb エリアに関連付けられた"URL" 変数にのみ適用されます ( 変数は通常入力可で、フォームに表示されます)。

フィルタは**WA OPEN URL** コマンドや他のナビゲーションコマンドには適用されません。

### 例題 1

.org, .net そして .fr Web サイトへのアクセスを禁止したい場合:

```
ARRAY TEXT ($filters;0)
ARRAY BOOLEAN ($AllowDeny;0)

APPEND TO ARRAY ($filters;"*.org")
APPEND TO ARRAY ($AllowDeny;False)
APPEND TO ARRAY ($filters;"*.net")
APPEND TO ARRAY ($AllowDeny;False)
APPEND TO ARRAY ($filters;"*.fr")
APPEND TO ARRAY ($AllowDeny;False)
WA SET URL FILTERS (MyWArea;$filters;$AllowDeny)
```

### 例題 2

日本のサイト以外へのアクセスを禁止したい場合(.jp):

```
ARRAY TEXT ($filters;0)
```

```
ARRAY BOOLEAN($AllowDeny;0)

APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"www*.jp") `Select *.jp
APPEND TO ARRAY($AllowDeny;True) `Allow
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)
```

### 例題 3

---

4D のWeb サイトにのみアクセスを許可する場合 (.com, .fr, .es, etc.):

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)

APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"www.4D.*") `Select 4d.fr, 4d.com...
APPEND TO ARRAY($AllowDeny;True) `Allow
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)
```

### 例題 4

---

ローカルのドキュメントにのみアクセスを許可 (C://doc フォルダ内):

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)

APPEND TO ARRAY($filters;"*") `Select all
APPEND TO ARRAY($AllowDeny;False) `Deny all
APPEND TO ARRAY($filters;"file://C:/doc/*")
`Select the path file:// allowed
APPEND TO ARRAY($AllowDeny;True) `Allow
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)
```

### 例題 5

---

特定のキーワードを含むサイトを除いて許可する場合:

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)

APPEND TO ARRAY($filters;"*")
APPEND TO ARRAY($AllowDeny;True) `Allow all
APPEND TO ARRAY($filters;"*elcaro*") `Deny all that contain elcaro
APPEND TO ARRAY($AllowDeny;False)
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)
```

### 例題 6

---

特定のIP アドレスへのアクセスを拒否する場合:

```
ARRAY TEXT($filters;0)
ARRAY BOOLEAN($AllowDeny;0)
APPEND TO ARRAY($filters;"*") `Select all

APPEND TO ARRAY($AllowDeny;True) `Allow all
APPEND TO ARRAY($filters;"86.83.*") `Select IP addresses beginning with 86.83.
APPEND TO ARRAY($AllowDeny;False) `Deny
APPEND TO ARRAY($filters;"86.1*") `Select IP addresses beginning with 86.1 (86.10, 86.135 etc.)
```

```
APPEND TO ARRAY($AllowDeny;False) `Deny
WA SET URL FILTERS(MyWArea;$filters;$AllowDeny)
` (Note that the IP address of a domain may vary).
```

## 📄 名前の変更、テーマの変更

### テーマの変更

---

**SHOW TOOL BAR** と **HIDE TOOL BAR** コマンドは、"ユーザーインターフェース"テーマから"ウィンドウ"テーマへと移動になりました。

### 定数の名前の変更

---

`On Picture Scroll` は `On Scroll` となりました。

4D v15 では、`On Scroll` フォームイベントは以下の二つの"スクロール可能"なオブジェクトに対して使用可能です。

- "トランケート(中央合わせしない)"の表示フォーマットになっているピクチャーフィールドもしくは変数(これは以前の4Dにおいても`On Picture Scroll` という名前で値59で使用可能でした)。
- リストボックス(4D v15からの新機能 - リストボックスのプロパティリスト内にて利用可能)。

#### 互換性に関する注意:

以前のバージョンでの`On Picture Scroll` フォームイベントの実装と、新しい`On Scroll` イベントには二つの小さな違いがあります:

- `On Picture Scroll` はオブジェクトメソッドとフォームメソッドとで生成されていました(フォームプロパティレベルではチェックの付け外しは不可能でした)。統一性の観点から、4D v15では、`On Scroll` イベントはオブジェクトメソッド内においてのみ生成されます。変換されたアプリケーションがピクチャーのスクロールをフォームメソッドで管理していた場合、そのコードを適切なオブジェクトメソッドへと移す必要があります。
- このイベントスタックにおいては、`On Picture Scroll` は他のユーザーイベント(例えば`On Click` 等)の前に呼び出す事が可能でした。それに対し`On Scroll` は常に他のユーザーイベントの後に生成されます。

### OPEN URL

---

**OPEN URL** は、OPEN WEB URL コマンドの新しい名前です。それに加え、このコマンドは新しい任意の`appName` 引数を受け取ります。これにより、`path` 引数で指定されたドキュメントまたはURLを開く際に使用するアプリケーションを指定することができます。

//同じテキストファイルを異なるアプリケーションで開く例:

```
OPEN URL ("C:\\temp\\cookies.txt") //Notepadを使用してファイルを開きます
```

```
OPEN URL ("C:\\temp\\cookies.txt";"winword") //MS Word (インストールされていれば)を使用してファイルを開きます
```

```
OPEN URL ("C:\\temp\\cookies.txt";"excel") //MS Excel (インストールされていれば)を使用してファイルを開きます
```

## SHOW TOOL BAR

### SHOW TOOL BAR

このコマンドは引数を必要としません

#### 説明

---

**SHOW TOOL BAR** コマンドは、カレントプロセスにおいて**Open form window** コマンドで作成されたカスタムのツールバーの表示を管理します。

**Open form window** コマンドに**Toolbar form window** オプションを使用してツールバーウィンドウが作成されている場合、このコマンドはそのウィンドウを表示状態にします。ツールバーウィンドウが既に表示状態であるとき、またはこのタイプのウィンドウが作成されていない場合には、コマンドは何もしません。

#### 例題

---

**HIDE TOOL BAR** コマンドの例題を参照して下さい。

OPEN URL ( path {; appName}{; \*} )

引数	型	説明
path	文字	➡ 開くドキュメントまたはURL
appName	文字	➡ 使用するアプリケーション名
*	演算子	➡ 指定した場合 = URLをエンコードしない, 省略した場合 = URLをエンコードする

## 説明

**OPEN URL** コマンドは、*appName* で指定したアプリケーションを使用して、*path* 引数に渡したファイルやURLを開きます。

*path* 引数には標準のURLまたはファイルのパス名のどちらかを渡す事ができます。コマンドは、OS X環境下ではコロン(':',)、Windows環境下ではスラッシュ('/')、またはfile://で始まるPosix URLを受け取る事ができます。

*appName* 引数が省略されていた場合、4Dはまず引数をファイルパス名として解釈しようとします。この場合4Dはシステムに、もっとも適切なアプリケーションを使用してファイルを開くよう、リクエストします (例えば、.htmlファイルにはブラウザを、.docファイルにはMS Wordを使用します)。この場合 \* 引数は無視されます。

*path* 引数に標準のURL (mailto:, news:, http:などのプロトコル) が渡された場合、4D はデフォルトのWebブラウザを開始し、URLにアクセスします。コンピュータに接続されたボリュームにブラウザがない場合、このコマンドは何も行いません。

*appName* parameter引数が渡された場合、コマンドはまずシステムに問い合わせをします。その名前のアプリケーションがインストールされていた場合、それが起動し、コマンドはそのアプリケーションに指定されたURLまたはドキュメントを開くようにリクエストします。

Windows環境下では、アプリケーション名を認識するメカニズムは、スタートメニューの「ファイル名を指定して実行」で使用されているものと同じです。例えば、以下の様なものを渡す事ができます：

- "iexplore" で Internet Explorer を起動
- "chrome" で Chrome を起動(インストールされていれば)
- "winword" で MS Word を起動(インストールされていれば)

**注:** インストールされているアプリケーションの一覧は、以下のキーのregistry にあります：

HKEY\_LOCAL\_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths

OS X 環境下では、アプリケーションを探すのに、インストールされたアプリケーションを全て自動的にインデックスしてくれるFinderを使用します。パッケージ名を指定することで.app形式のアプリケーションをどれも認識する事ができます：

- "safari"
- "FireFox"
- "TextEdit"

*appName* 引数で指定されたアプリケーションが見つからない場合でも、エラーは返されません。コマンドはその引数が指定されなかったものとして実行されます。

4Dは自動でURLの特別文字をエンコードします。引数に\*を渡すと、4DはURL特別文字のエンコードを行いません。このオプションを使用して、以下のようなURLの送信が可能です: "*http://www.server.net/page.htm?q=something*"

**注:** このコマンドはWebプロセスから呼ばれた時は動作しません。

## 例題 1

以下では、このコマンドがURL引数として受け入れる異なるタイプの文字列を例示します：

```
OPEN URL("http://www.4d.com")
OPEN URL("file:///C:/Users/Laurent/Documents/pending.htm")
OPEN URL("C:\\Users\\Laurent\\Documents\\pending.htm")
OPEN URL("mailto:jean_martin@4d.fr")
```



## 例題 2

---

この例は最適なアプリケーションを起動するために使用できます:

```
$file:=Select document ("";";0)
If (OK=1)
    OPEN URL (Document)
End if
```

## 例題 3

---

*appName* 引数を使用すると同じテキストファイルを異なるアプリケーションを使用して開くことができます:

```
OPEN URL ("C:\\temp\\cookies.txt") //ファイルをメモ帳で開く
OPEN URL ("C:\\temp\\cookies.txt";"winword") //ファイルをMS Wordで開く(インストールされていれば)
OPEN URL ("C:\\temp\\cookies.txt";"excel") //ファイルをMS Excelで開く(インストールされていれば)
```

## 📄 廃止予定の機能

### 廃止予定ではなくなったもの

---

まずは、以前のバージョンのドキュメントでは廃止予定と宣言されていたものの、v15で評価しなおされたのち、廃止予定ではないと見直されたコマンドまたは機能の一覧から始めましょう。これらのコマンドは使用することはできますが、一般的により新しいコマンドで置き換えることが推奨されるものです：

**Before**

**After**

**In break**

**In footer**

**In header**

**Activated**

**Deactivated**

**Document type**

**Modified**

**Outside call**

**SHOW TOOL BAR**

**HIDE TOOL BAR**

これらのうち最後の二つのコマンドはカスタムのツールバーを管理する為に再有効化されたものであり、"ユーザーインターフェイス"テーマから"ウィンドウ"テーマへと移動されました。それに加え、**Open form window** コマンドは新しい `Toolbar form window` 型を受け取るようになりました。

### 廃止予定のコマンドの名前の変更

---

4D v15以降、廃止予定のコマンドは系統的に"\_o\_"の接頭辞がつけられ、4Dコマンド一覧に表示されないようになりました。およそ50ほどの4Dコマンドがこのように改名されました。

4D v15で改名された廃止予定コマンドの完全な一覧については、[廃止予定のコマンドの改名](#) の章を参照して下さい。

### XSLT

---

XSLT処理コマンドは廃止予定と宣言され、それに伴い廃止予定の接頭辞が付けられました。互換性の観点から、XSL変換は4Dではまだサポートされますが、その使用は推奨されません。XSLT処理のサポートは将来の4Dのリリースにおいて削除されます。

**OS X 用4D Server 64-bit に関する注意:** XSLT はOS X用4D Server 64 bit版では使用できません。その結果、このアプリケーションからこれらのコマンドのどれかを呼び出した場合、エラー33"実装されていないコマンドまたはファンクションです"が生成されます。

4Dでは、データベース内のXSLT テクノロジーを置き換えるソリューションを二つ用意しています：

- 4D のv14.2以降実装されている、同等の機能をもつ `PHPlibxslt` モジュールを使用する方法。4Dでは4D XSLTコマンドの代理としてPHP XSLを使用するのを手助けするためのドキュメントをご用意しています。：[Download XSLT with PHP technical document](#) (PDF)
- **PROCESS 4D TAGS** コマンドによってもたらされた新しい選択肢を使用する方法。このコマンドの用途は4D v15において著しく拡張されました。

### 廃止予定の定数

---

- `Has toolbar button Mac` 定数は廃止予定となりました。これに対応するオプションが、OS X 10.6以降Appleによって

廃止予定とされてしまったからです。この定数は"Open Form Window" と "Open Window" の両テーマで使用可能でした。これは4D v15で O\_Has toolbar button Mac と改名されました。

- qr font 定数(1) は O\_qr font と改名されました。この定数は廃止予定となり、今後使用されるべきではありません(互換性は保たれますが、将来のリリースにおいてサポートされなくなります)。新しい O\_qr font 定数(10)が、"**QR Text Properties**"テーマ内に追加されました。今後フォントをするためにはこの定数を使用し、*string* 値を使用する必要があります。その際、**FONT LIST** コマンドによって返される名前を使用することができます。

---

## 旧式ネットワークレイヤー

**Get database parameter** と **SET DATABASE PARAMETER** コマンドに対して新しいセクターが使用可能になりました: Use legacy network layer (倍長整数、87)

この新しいセクターはクライアント/サーバー接続のための旧式ネットワークレイヤーのカレントの状態を設定・取得します。以前のネットワークレイヤーはアプリケーションの中において徐々に**ServerNet** ネットワークへと置き換えられていきます。**ServerNet** は未来のバージョンの4Dで将来のネットワークの進化の恩恵を享受するために必須となります。互換性上の理由から、旧式ネットワークレイヤーは既存のアプリケーションの移行を容易にするために引き続きサポートはされています(v15より前のバージョンから変換されたアプリケーションにおいてはデフォルトで使用されています)。**ServerNet** ネットワークレイヤーを有効化することができます。

- 旧式ネットワークレイヤーを使用するためにはこの引数に1を渡します(これにより**ServerNet** は無効化されます)。
- 旧式ネットワークレイヤーを無効化するためにはこの引数に0を渡します(これにより**ServerNet** を使用します)。

このプロパティは互換性ダイアログ**新しい互換性オプション**の"旧式ネットワークレイヤーを使用"オプションを使用することによって設定することもできます(**新しいServerNetネットワークレイヤー**の章も参照して下さい)。

**実装についての注記:** ServerNet ネットワークレイヤーは 4D v15の "プレビュー" リリースで提供されています。


---

## 4D Write と 4D View プラグインの進化

4D Write と 4D Viewプラグインは引き続きサポートはされますが、将来の4Dにおいてこれ以上開発が進むことはありません。4Dは現在それに代わるソリューションとして"4D Write Pro" と "4D View Pro"を開発中であり、それらの機能は徐々に取り入れられていく予定です。これらの新しいツールの初期バージョンは既に4D v15でご利用いただけます。

## Before

Before -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがbeforeである場合にはTrueを返す

### 説明

---


**Before** はBefore 実行サイクルでTrue を返します。

**Before** 実行サイクルを生成させるには、デザインモードでそのフォームやオブジェクトのOn Loadイベントプロパティを必ず選択してください。

**注:** このコマンドは、**Form event** コマンドを用いてOn Load イベントを返すかどうかをテストするのと同等と言えます。

## In break

In break -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがブレイクエリア内にある場合にはTrueを返す

### 説明

---


**In break** はIn break 実行サイクルでTrueを返します。

**In break** 実行サイクルを生成させるには、デザインモードでそのフォームやオブジェクトでOn Printing Break イベントプロパティを必ず選択してください。

**注:** このコマンドは、**Form event** コマンドを用いてOn Printing Break イベントを返すかどうかをテストするのと同等と言えます。

## In footer

In footer -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがフッター内にある場合にはTrueを返す

### 説明

---


**In footer** はIn footer 実行サイクルに対してTrueを返します。

**In footer** 実行サイクルを生成させるには、デザインモードでそのフォームやオブジェクトでOn Printing footerイベントプロパティを必ず選択してください。

**注:** このコマンドは、**Form event** コマンドを用いてOn Printing footerイベントを返すかどうかをテストするのと同等と言えます。

## In header

In header -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがヘッダー内にある場合にはTrueを返す

### 説明

---

**In header** はIn header実行サイクルに対してTrueを返します。


**In header** 実行サイクルを生成させるには、デザインモードでそのフォームやオブジェクトで On Header イベントプロパティを必ず選択してください。

**注:** このコマンドは、**Form event** コマンドを使用して、On Header イベントを返すかどうかをテストするのと同じです。



## Activated

Activated -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがactivationである場合にTrueを返す

### 説明

---

**Activated** コマンドは、(廃止予定)フォームを含むウインドウがプロセスの最前面のウインドウになると、そのフォームメソッドで**True** を返します。


**注:** このコマンドは、**Form event** コマンドを用いて On Activate イベントを返すかどうかをテストするのと同等と言えます。

**警告:** フォームの**Activated** フェーズに**TRACE** または**ALERT**を置かないでください。入れると無限ループになります。

**Note: Activated** 実行サイクルを生成させるには、デザインモードでそのフォームの On Activate イベントプロパティを必ず選択してください。

## Deactivated

Deactivated -> 戻り値

引数	型	説明
戻り値	ブール	 実行サイクルがdeactivationである場合にTrueを返す

### 説明

---

**Deactivated** コマンドはプロセスの最前面のウィンドウが後ろに移動すると、そのフォームメソッドでTrue を返します。

**Deactivated** 実行サイクルを生成させるには、デザインモードでそのフォームやオブジェクトの On Deactivate イベントプロパティを必ず選択してください。

**注:** このコマンドは、 **Form event** コマンドを用いて On Deactivate イベントを返すかどうかをテストするのと同等と言えます。

Document type ( document ) -> 戻り値

引数	型	説明
document	文字 →	ドキュメント名
戻り値	文字 ↩	Windowsファイル拡張子(3文字以内の文字列) またはMac OSファイルタイプ (4文字の文字列)

### 説明

**Document type** コマンドは、*document*に渡したドキュメントの名前とパス名を持つドキュメントのタイプまたは拡張子を返します。

Windowsでは、**Document type** は、ドキュメントのファイル拡張子(例えば、Microsoft Wordドキュメントを意味する'*DOC*'や実行ファイルを意味する'*EXE*'など)、または対応するMac OSに基づいた4文字のファイルタイプを返します。後者は4Dによりまたは**MAP FILE TYPES**の呼び出しによって、その対等なWindowsのファイル拡張子でマップされた場合です(例えば、'*TXT*'ファイルの拡張子を意味する'*TEXT*')。

Macintoshでは、指定されていると、**Document type** はドキュメントの4文字のファイルタイプを返します(例えば、テキストドキュメントを意味する'*TEXT*'、ダブルクリック可能なアプリケーションを意味する'*APPL*' など)。

### 互換性に関するメモ

Mac OS Xでは、Mac OSのファイルタイプは、もうはやサポートされていません。現在では、Windowsのように、名前の接尾辞に基づいてファイルの識別を行います(を参照)。互換性の理由により、それでも**MAP FILE TYPES**を使用して指定された場合は、ドキュメントのMac OSタイプを読み込みます。

Modified ( aField ) -> 戻り値

引数	型		説明
aField	フィールド	→	テストするフィールド
戻り値	ブール	↩	フィールドに新しい値が代入されていればTrue, そうでなければFalse

## 説明

**Modified** はデータ入力中、プログラムを使用して *field* に値が代入されていたり、データ入力中に値が編集された場合に、**True**を返します。**Modified** コマンドはフォームメソッド（またはフォームメソッドから呼ばれたサブルーチン）で使用されなければなりません。

このコマンドは同じ実行サイクル内でのみ意味のある値を返します。特に以前の **\_o\_During** 実行サイクルに対応するフォームイベント(On Clicked、On After Keystroke、等)では、**False**に設定されます。

データ入力時には、（元の値が変更されたかどうかに関わらず）ユーザがフィールドを編集した後別のフィールドへ移動するか、コントロールをクリックすると、フィールドが更新されたとみなされます。tabキーでフィールドを移動しただけでは、**Modified** はTrueにならない点に注意してください。**Modified** がTrueになるためには、フィールドが編集されなければなりません。

メソッドの実行時には、フィールドに値が割り当てられると（異なる値かどうかに関係なく）、フィールドが編集されたものと解釈されます。

**注: Modified** は、**PUSH RECORD** と **POP RECORD** コマンド実行後は、常に**True**を返します。

いずれの場合でも、フィールドの値が実際に変更されたかどうかを調べるには、**Old** コマンドを使用します。

**注: Modified** はあらゆるタイプのフィールドに対して適用できますが、このコマンドを**Old** コマンドと組み合わせて使用する場合には、**Old** コマンドの制約に注意してください。詳細については**Old** コマンドの説明を参照してください。

データ入力時には、フォームメソッドで**Modified** を使用するよりも、オブジェクトメソッドで**Form event** を使用して処理を実行する方が簡単です。フィールドが修正される度にOn Data Changeイベントがオブジェクトメソッドに送信されるので、オブジェクトメソッドの利用はフォームメソッドで**Modified** を使用したのと同じ意味を持ちます。

**注:** 処理を正しく実行するため、**Modified** コマンドはフォームメソッドまたは、フォームメソッドから呼び出されるメソッド内でのみ使用します。

## 例題 1

次の例は、[Orders]Quantity フィールドや[Orders]Price フィールドが変更されたかどうかを判定します。どちらかが変更されると、[Orders]Total フィールドを再計算します。

```
If ( (Modified ([Orders]Quantity) | (Modified ([Orders]Price) )
    [Orders]Total :=[Orders]Quantity*[Orders]Price
End if
```

2番目の行をサブルーチンにして、[Orders]Quantity フィールドと[Orders]Price フィールドのオブジェクトメソッドのOn Data Changeフォームイベントでそのサブルーチン呼び出ししても、同じ結果となります。

## 例題 2

[anyTable]テーブルのレコードを選択し、次に[anyTable]Important fieldフィールドが修正される可能性がある複数のサブルーチン呼び出ししますが、これらのメソッドはレコードの保存を行いません。メインのメソッドの終わりで、**Modified** コマンドを使用してレコードを保存する必要があるかどうかを調べています:

```
、レコードがカレントレコードとして選択済みです
、サブルーチンを使用して処理を行います
DO SOMETHING
DO SOMETHING ELSE
```

**DO NOT FORGET TO DO THAT**

\ ...

\ レコードを保存する必要があるか検証します


**If** (**Modified** ([anyTable] Important field))

**SAVE RECORD** ([anyTable])

**End if**

## Outside call

Outside call -> 戻り値

引数	型	説明
戻り値	ブール	 True if the execution cycle is an outside call

### 説明

---

**Outside call** は、実行サイクルのあとにTrueを返します。

**Outside call** 実行サイクルが生成されるためには、デザイン環境において [On Outside call](#) イベントプロパティがフォーム・オブジェクトに対して選択されていることを確認して下さい。

**注:** このコマンドは、**Form event** コマンドを使用して、[On Outside call](#) イベントを返すかどうかを試すのと同じであると言えます。

### HIDE TOOL BAR

このコマンドは引数を必要としません

### 説明

---

**HIDE TOOL BAR** コマンドは、カレントプロセスにおいて **Open form window** コマンドで作成されたカスタムのツールバーの表示を管理します。

**Open form window** コマンドに **Toolbar form window** オプションを使用してツールバーウィンドウが作成されている場合、このコマンドはそのウィンドウを非表示にします。ツールバーウィンドウが既に非表示状態であるとき、またはこのタイプのウィンドウが作成されていない場合には、コマンドは何もしません。

### 例題

---

OS X において、カスタムのツールバーと **Has full screen mode Mac** オプションを持つ標準のウィンドウを定義したとします。ツールバーが表示されている状態で標準のウィンドウがユーザーによって最大化された場合、最大化されたウィンドウとツールバーが被ってしまうのは避けたいところです。

これを避けるためには、標準のウィンドウの **On Resize** フォームイベント内にて、このウィンドウがフルスクリーンモード切り替わった瞬間を検知し、**HIDE TOOL BAR** を呼び出す必要があります：

```
Case of
  : (Form event=On Resize)
    GET WINDOW RECT ($left;$top;$right;$bottom)
    If (Screen height=($bottom-$top))
      HIDE TOOL BAR
    Else
      SHOW TOOL BAR
    End if
  End case
```



## 🔧 Open form window

Open form window ( {aTable ;} formName {; type {; hPos {; vPos {; \*}}}} ) -> 戻り値

引数	型	説明
aTable	テーブル	→ フォームが属するテーブル、または省略時デフォルトテーブル
formName	文字	→ フォーム名
type	倍長整数	→ ウィンドウタイプ
hPos	倍長整数	→ ウィンドウの横位置
vPos	倍長整数	→ ウィンドウの縦位置
*	演算子	→ ウィンドウの現在の位置とサイズを保存
戻り値	WinRef	→ ウィンドウ参照番号

### 説明

**Open form window** コマンドはフォーム *formName* のサイズとリサイズプロパティを使用して、新しいウィンドウを開きます。

*formName* フォームはウィンドウに表示されません。フォームを表示するには、フォームをロードするコマンド (**ADD RECORD** 等) を呼び出さなければなりません。

デフォルトで (*type* 引数が渡されないと)、クローズボックス付きの標準ウィンドウが開かれます。 **Open window** コマンドと異なり、クローズボックスにはメソッドは割り当てられません。クローズボックスをクリックすると、On Close Box フォームイベントが有効にされている場合を除き、ウィンドウがキャンセルされ閉じられます。On Close Box フォームイベントが有効であれば、割り当てられたコードが実行されます。

*formName* フォームがリサイズ可能であれば、開かれるウィンドウにはズームボックスとグローボックスが付加されます。

**Note:** フォームの主なプロパティを取得するには **FORM GET PROPERTIES** コマンドを使用します。

任意の *type* 引数は、ウィンドウのタイプを指定するために使用します。以下のいずれかの定数を渡さなければなりません (**Open Form Window** テーマ内):

定数	型	値
Form has full screen mode Mac	倍長整数	65536
Modal form dialog box	倍長整数	1
Movable form dialog box	倍長整数	5
Palette form window	倍長整数	1984
Plain form window	倍長整数	8
Pop up form window	倍長整数	32
Sheet form window	倍長整数	33
Toolbar form window	倍長整数	35

### Notes:

- (グローボックス, クローズボックスなど...) 作成されたウィンドウの属性は、選択された *type* に対する OS のインタフェース仕様に基づきます。ゆえに使用するプラットフォームによって異なる結果が得られる場合があります。
- Has toolbar button Mac OS と Has full screen mode Mac 定数は、他の定数に加算して使用しなければなりません。
- ウィンドウタイプに関する詳細は **ウィンドウタイプ** のセクションを参照してください。ただし **Open Form Window** テーマの定数のみが **Open form window** コマンドで利用可能であることに注意してください。

Toolbar form window 定数が渡された場合、ウィンドウの位置、サイズ、グラフィックプロパティはツールバーのそれに準拠します。具体的には、以下の用になります:

- ウィンドウは常にメニューバーのすぐ下に表示されます。
- ウィンドウの水平方向のサイズはデスクトップ(OS X)または4Dのメインウィンドウ内部(Windows)の利用可能な全ての水平方向のスペースを埋める形で自動的に調整されます。ウィンドウの垂直方向のサイズは、他の全てのフォームウィンドウタイプ同様、フォームプロパティに準じます。
- ウィンドウに境界線はなく、手動で移動・リサイズはできません。また、*hPos*、*vPos* そして \* 引数は渡されていた場

合でも無視されます。

- 二つの異なるツールバーウィンドウを同時に作成することはできません。もしツールバーウィンドウが既に存在している状態で、**Open form window** に `Toolbar form window type` が渡された状態で呼び出された場合、エラー-10613("ツールバー型のフォームウィンドウを二つ作成することはできません")が生成されます。

**OS Xフルスクリーンモードでのツールバーフォームウィンドウ:** お使いのアプリケーションがツールバーウィンドウと、フルスクリーンモードをサポートする(`Has full screen mode Mac` オプション)標準のウィンドウの両方を表示する場合、標準のウィンドウがフルスクリーンモードに切り替わった時にはインターフェースのルールに則り、ツールバーを非表示にしなければなりません。ウィンドウがフルスクリーンモードに切り替わったかどうかを調べるためには、ウィンドウの垂直方向のサイズがスクリーンの高さと同じになったかをテストして下さい(詳細は**HIDE TOOL BAR** コマンドを参照して下さい)。

オプションの引数 `hPos` を使用して、ウィンドウの横位置を指定できます。特定の位置をピクセル単位で指定するか (**Open window** コマンド参照)、**Open Form Window** テーマの以下の定義済み定数を渡します:

定数	型	値
Horizontally centered	倍長整数	65536
On the left	倍長整数	131072
On the right	倍長整数	196608

オプションの引数 `vPos` を使用して、ウィンドウの縦位置を指定できます。特定の位置をピクセル単位で指定するか (**Open window** コマンド参照)、**Open Form Window** テーマの以下の定義済み定数を渡します:

定数	型	値
At the bottom	倍長整数	393216
At the top	倍長整数	327680
Vertically centered	倍長整数	262144

これらの引数はツールバーとメニューバーの表示状態、およびWindowsではアプリケーションウィンドウ現在のサイズを考慮します。

オプションの引数 `*` を渡すと、閉じられるときにその時点での位置とサイズが記憶されます。ウィンドウが再度開かれると、以前の位置とサイズが再現されます。この場合、`vPos` と `hPos` 引数はウィンドウが最初に開かれるときにのみ使用されません。

## 例題 1

以下のコードはクローズボックス付きの標準のウィンドウを開き、自動で"Input"フォームのサイズに調整します。フォームはリサイズ可能に設定されているので、ウィンドウはグローとズームボックスを持ちます:

```
$winRef :=Open form window([Table1];"Enter")
```

## 例題 2

以下のコードはプロジェクトフォーム"Tools"に基づき、スクリーンの左上の位置にフローティングパレットを開きます。このパレットは閉じられた時の位置を記憶し、再度開かれるときにはその位置が使用されます:

```
$winRef :=Open form window("Tools";Palette form window;On the left;At the top;*)
```

## 廃止予定のコマンドの改名

---

4D v15では分かりやすさのために、今までついていなかったものも含めて全ての廃止予定のコマンドに"\_o\_"の接頭辞をつけました。

廃止予定のコマンドは4Dリストには表示されなくなったため([廃止予定のコマンドは非表示に](#)の章を参照して下さい)、それらを選択することはできなくなりました。廃止予定のコマンドは既存のコードの中でのみ改名されます。

以下は、4D v15で改名された廃止予定のコマンドの一覧です(ここにあるものは4Dリストには表示されません):

**以前の名前**

ADD DATA SEGMENT

ADD SUBRECORD

ALL SUBRECORDS

APPLY TO SUBSELECTION

ARRAY STRING

ARRAY TO STRING LIST

Before subselection

C\_STRING

Convert cas

Create resource file

CREATE SUBRECORD

DATA SEGMENT LIST

DELETE RESOURCE

DELETE SUBRECORD

DISABLE BUTTON

ENABLE BUTTON

End subselection

FIRST SUBRECORD

Font name

Font number

Get component resource ID

Get platform interface

INVERT BACKGROUND

ISO to Mac

LAST SUBRECORD

Mac to ISO

Mac to Win

MODIFY SUBRECORD

NEXT SUBRECORD

ORDER SUBRECORDS BY

PICTURE TYPE LIST

PREVIOUS SUBRECORD

QT COMPRESS PICTURE

QT COMPRESS PICTURE FILE

QT LOAD COMPRESS PICTURE FROM FILE

QUERY SUBRECORDS

Records in subselection

REDRAW LIST

SAVE PICTURE TO FILE

SET PICTURE RESOURCE

SET PLATFORM INTERFACE

SET RESOURCE

SET RESOURCE NAME

SET RESOURCE PROPERTIES

SET STRING RESOURCE

SET TEXT RESOURCE

USE EXTERNAL DATABASE

**4D v15での新しい名前**

\_o\_ADD DATA SEGMENT

\_o\_ADD SUBRECORD

\_o\_ALL SUBRECORDS

\_o\_APPLY TO SUBSELECTION

\_o\_ARRAY STRING

\_o\_ARRAY TO STRING LIST

\_o\_Before subselection

\_o\_C\_STRING

\_o\_Convert case

\_o\_Create resource file

\_o\_CREATE SUBRECORD

\_o\_DATA SEGMENT LIST

\_o\_DELETE RESOURCE

\_o\_DELETE SUBRECORD

\_o\_DISABLE BUTTON

\_o\_ENABLE BUTTON

\_o\_End subselection

\_o\_FIRST SUBRECORD

\_o\_Font name

\_o\_Font number

\_o\_Get component resource ID

\_o\_Get platform interface

\_o\_INVERT BACKGROUND

\_o\_ISO to Mac

\_o\_LAST SUBRECORD

\_o\_Mac to ISO

\_o\_Mac to Win

\_o\_MODIFY SUBRECORD

\_o\_NEXT SUBRECORD

\_o\_ORDER SUBRECORDS BY

\_o\_PICTURE TYPE LIST

\_o\_PREVIOUS SUBRECORD

\_o\_QT COMPRESS PICTURE

\_o\_QT COMPRESS PICTURE FILE

\_o\_QT LOAD COMPRESS PICTURE FROM FILE

\_o\_QUERY SUBRECORDS

\_o\_Records in subselection

\_o\_REDRAW LIST

\_o\_SAVE PICTURE TO FILE

\_o\_SET PICTURE RESOURCE

\_o\_SET PLATFORM INTERFACE

\_o\_SET RESOURCE

\_o\_SET RESOURCE NAME

\_o\_SET RESOURCE PROPERTIES

\_o\_SET STRING RESOURCE

\_o\_SET TEXT RESOURCE

\_o\_USE EXTERNAL DATABASE

USE INTERNAL DATABASE

[\\_o\\_USE INTERNAL DATABASE](#)

Win to Mac

[\\_o\\_Win to Mac](#)

## XSLT コマンド

XSLT コマンドは4D v14 R4にて名前が変更になりました:

以前の名前	4D v15での新しい名前
XSLT APPLY TRANSFORMATION	<a href="#">_o_XSLT APPLY TRANSFORMATION</a>
XSLT GET ERROR	<a href="#">_o_XSLT GET ERROR</a>
XSLT SET PARAMETER	<a href="#">_o_XSLT SET PARAMETER</a>

より詳細な情報に関しては、[XSLT コマンドは廃止予定に](#) の章を参照して下さい。

## 4D Pack コマンド

4D v14 R5以降、いくつかの4D Packコマンドも廃止予定となりました:

以前の名前	新しい名前
AP FCLOSE	<a href="#">_o_AP FCLOSE</a>
AP fopen	<a href="#">_o_AP fopen</a>
AP FPRINT	<a href="#">_o_AP FPRINT</a>
AP fread	<a href="#">_o_AP fread</a>
AP Save BMP 8 bits	<a href="#">_o_AP Save BMP 8 bits</a>
AP Add table and fields	<a href="#">_o_AP Add table and fields</a>
AP Create relation	<a href="#">_o_AP Create relation</a>
AP Get file MD5 digest	<a href="#">_o_AP Get file MD5 digest</a>
AP ShellExecute	<a href="#">_o_AP ShellExecute</a>

より詳細な情報については、[廃止予定のコマンドの改名](#) の章を参照して下さい。

## 廃止予定ではなくなったコマンド

---

その一方で、ドキュメントにおいて廃止予定だと以前宣言されていた一部のコマンドのうち、上述の定義に当てはまらないものは、"再採用"されることになりました。

これらのコマンドは、以前のプログラミングモードに対応し、より効率的なコードで置き換えられるために現在では用法は限定的ですが、これらの維持に継続に対する疑問はないとされるものです:

[Activated](#)  
[Outside call](#)  
[After](#)  
[Before](#)  
[Deactivated](#)  
[In header](#)  
[In footer](#)  
[In break](#)  
[Modified](#)  
[Document type](#)

## QR Text Properties

定数	型	値	コメント
_O_qr font	倍長整数	1	4D v14R3 以降廃止予定( <u>qr font name</u> を使用して下さい)
qr alternate background color	倍長整数	9	代替背景色
qr background color	倍長整数	8	背景色番号
qr bold	倍長整数	3	太字スタイル属性 (0 または 1)
qr font name	倍長整数	10	<b>FONT LIST</b> コマンドなどによって返されたフォント名
qr font size	倍長整数	2	ポイント単位のフォントサイズ (9 ~ 255)
qr italic	倍長整数	4	イタリックスタイル属性 (0 または 1)
qr justification	倍長整数	7	テキスト整列属性 (0 = デフォルト, 1 = 左, 2 = 中央, 3 = 右)
qr text color	倍長整数	6	フォントカラー属性 (カラー番号)
qr underline	倍長整数	5	下線スタイル属性 (0 または 1)

## 無効化された機能

### SSL v2 と SSL v3 プロトコルは恒久的に無効化

---

デフォルトでは、TLS v1 プロトコルのみが使用可能となりました。

これらの変更は、以下のものを含む、4Dでのセキュアな通信に全て適用されます：

- HTTP サーバー通信
- SQL サーバー通信
- リモート 4D/4D Server 接続
- HTTP クライアント接続

結果として、以下の接続に関係するアプリについては、必須要件を更新する必要があるかもしれません：

- **TLSをサポートしないブラウザまたはHTTP クライアント(例えばIE6以前等)は、セキュアなプロトコル(ポート443)を使用して4D Web Server/サービスへと接続する事はできなくなりました。**
- **HTTP Get** または **HTTP Request** コマンドは、TLSをサポートしないサーバーにセキュアモードで接続する事はできなくなりました。

ネットワークセキュリティ機能は4D v15で最適化され、お使いの4D アプリケーションの保護を強化するようになりました：

- 脆弱な暗号化方式群は除去されました。
- デフォルトの4D証明書のキーの長さは2048ビットへと増加されました。
- セキュアな通信に、自分の暗号化鍵を使用する事ができるようになりました。

これらの変更は以下のセキュアな通信に関係します：

- クライアント/サーバー
- SQL サーバー
- HTTP クライアント

## 付録: 変換に有用なメソッド

### 不足しているインデックスを作成

---

不足しているインデックスを作成するTech Tip: [#command\\_37](#)

### 重複不可属性でインデックスがついていないフィールドの一覧のディスクファイルの生成

---

重複不可属性でインデックスがついていないフィールドの一覧のディスクファイルを生成するためのTech Tip:  
[#command\\_38](#)

### 組み込みアプリ: リージョン設定の使用

---

メソッドエディターはデフォルトで"English-US"設定に を参照して下さい。

リージョンシステム設定を使用 の設定がマシンに対してローカルではなくなったため、このオプションは組み込みアプリには含まれていません。

組み込みアプリのコンテキストに置いては、それぞれのローカルマシンに置いて4D v15設定ファイルを編集し、"use\_localized\_language"キーを"true"へと設定する必要があります:

```
$UserPreference:=Get 4D folder(Active 4D Folder)+"4D Preferences v15.4DPreferences"  
$ref:=DOM Parse XML source($UserPreference;True)  
$refElem:=DOM Find XML  
element($ref;"preferences/com.4d/method_editor/options";arrElementRefs) //カレント値  
DOM GET XML ATTRIBUTE BY NAME($refElem;"use_localized_language";$attrValue)  
If($attrValue="false") //v14の振る舞いに戻す  
    DOM SET XML ATTRIBUTE($refElem;"use_localized_language";"true")  
End if  
DOM EXPORT TO FILE($ref;$UserPreference)  
DOM CLOSE XML($ref)
```